IN THE UNITED STATES DISTRICT COURT
FOR THE EASTERN DISTRICT OF TEXAS

MYSCRIPT SAS, Plaintiff, v. GOODNOTES LIMITED, Defendant.	CIVIL ACTION NO. ____ **JURY TRIAL DEMANDED**

COMPLAINT FOR PATENT INFRINGEMENT

This is an action for patent infringement in which Plaintiff MyScript SAS ("MyScript") makes the following allegations against Defendant Goodnotes Limited ("Goodnotes") for infringing the Patents asserted in this matter:

THE PARTIES

1. MyScript is a French entity, having a principal place of business located at 3 rue de la Rainière 44339 Nantes Cedex 3 France.

2. Upon information and belief, Goodnotes is a privately held company headquartered in the United Kingdom, having a principal place of business located at 1 Bartholomew Lane, London, United Kingdom, EC2N 2AX, and with an additional office in Hong Kong. Goodnotes engages in business, either directly or through its agents, on an ongoing basis in this judicial district and elsewhere in the United States. Although Goodnotes is engaged in business in Texas, it has not designated an agent for service of process in Texas. The Texas Secretary of State, therefore, is an agent for service of process for Goodnotes under the Texas Civil Practice and Remedies Code § 17.044(b). On information and belief, Goodnotes is responsible for selling the Accused Products in Texas and in the Eastern District of Texas, directly or through intermediaries.

JURISDICTION AND VENUE

3. This Court has subject matter jurisdiction pursuant to 28 U.S.C. §§ 1331 and 1338(a) because this action arises under the patent laws of the United States, 35 U.S.C. §§ 1 *et seq*.

4. This Court has personal jurisdiction over Goodnotes because Goodnotes has (1) committed acts of patent infringement and contributed to and induced acts of patent infringement by others in this District; (2) regularly done or solicited business in this District; (3) engaged in other persistent courses of conduct and derived substantial revenue by its offering of infringing products and services and providing infringing products and services in this District; and (4) purposefully established substantial, systematic, and continuous contacts with this District and should have reasonably expected to be subject to suit here by its offering of infringing products and services and providing infringing products and services in this District.

5. Goodnotes specifically targeted Texas and this judicial district on its Goodnotes USA Tour 2024 to promote, advertise, use, sell, and offer for sale of its infringing products. Between November 2024 and January 2025, Goodnotes held promotional events at the following Texas universities, including the Texas Woman's University, located in the Eastern District of Texas:

a. University of Houston on November 17, 2024 (**Exhibit A**).

b. Texas A&M University on November 20, 2024 (**Exhibit B**).

c. University of Texas, Austin on November 21, 2024 (**Exhibit C**).

d. Texas State University on November 22, 2024 (**Exhibit D**).

e. University of Texas, Arlington on January 13, 2025 (**Exhibit E**).

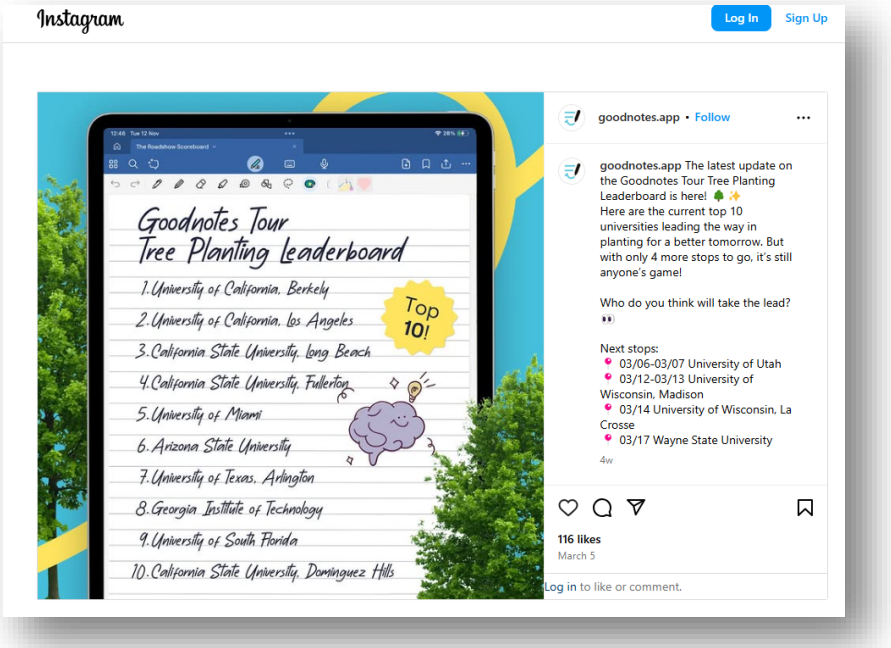
f. Texas Christian University on January 14, 2025 (**Exhibit F**).

g. Texas Woman's University on January 15, 2025 (**Exhibit G**).

6. Goodnotes has sold infringing products in Texas.

a. As part of the Goodnotes USA Tour 2024, Goodnotes stated that "[f]or every ten downloads of Goodnotes at participating universities, we will donate to plant a tree through One Tree Planted. To add a bit of friendly competition, we'll have a leaderboard tracking which colleges plant the most trees." Available at (https://www.goodnotes.com/blog/goodnotes-usa-tour#help-us-plant-trees-along-the-way) (last visited 05/08/2025).

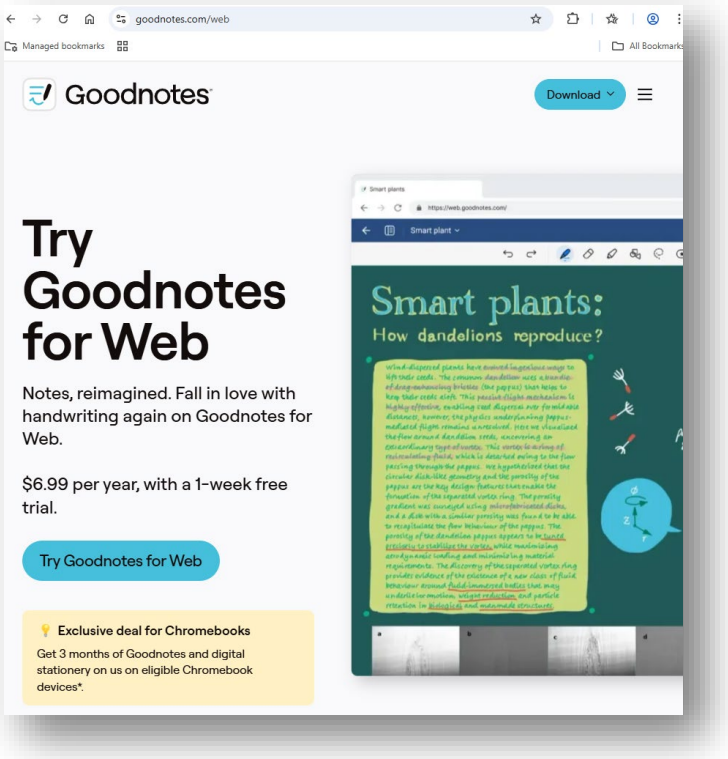
b. On March 5, 2025, Goodnotes' Instagram account (https://www.instagram.com/goodnotes.app/p/DG0zjllug-N/ (last visited 05/08/2025)) posted the leaderboard, indicating that the University of Texas, Arlington was among the top 10 universities with the most downloads of Goodnotes products:



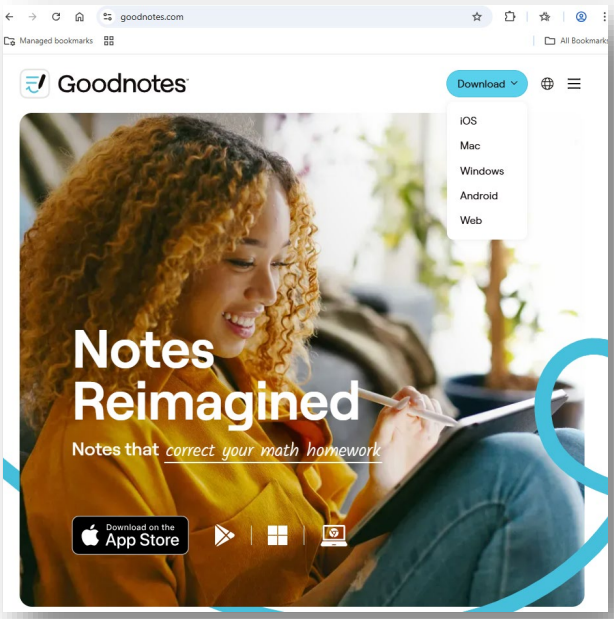
7. Goodnotes has and continues to purposefully place and sell infringing products in Texas and within this judicial district. For example, Goodnotes distributes Goodnotes 6 in this judicial district through Goodnotes website, Apple's App Store, and distributes similar products

in this judicial district through Google's Google Play Store, and Microsoft's Store for Windows devices (collectively the "Accused Products"):

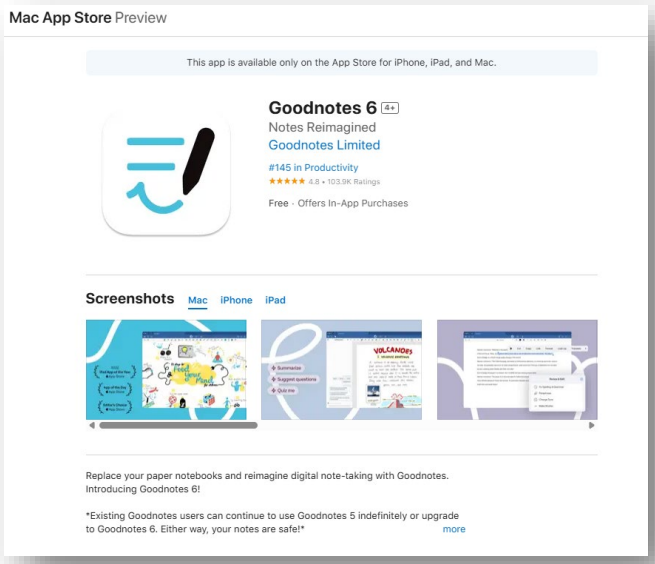
a. Goodnotes' website (goodnotes.com/web) for "Web" use:



b. Goodnotes provides links on its website (goodnotes.com) under Download to connect users to third-party websites to download the Accused Products, including links for Goodnotes 6 available on iOS and MAC, and similar products available on Windows, Android, and Web:

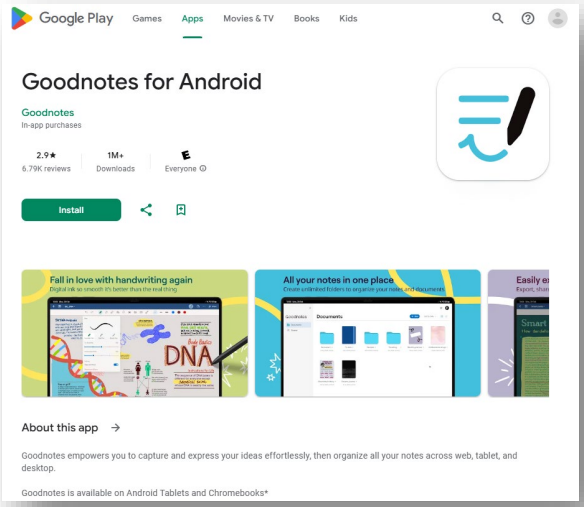


c. Apple App Store (iOS) (https://apps.apple.com/us/app/goodnotes-6/id1444383602) for Goodnotes 6 for use on iPhone, iPad, and Mac devices:

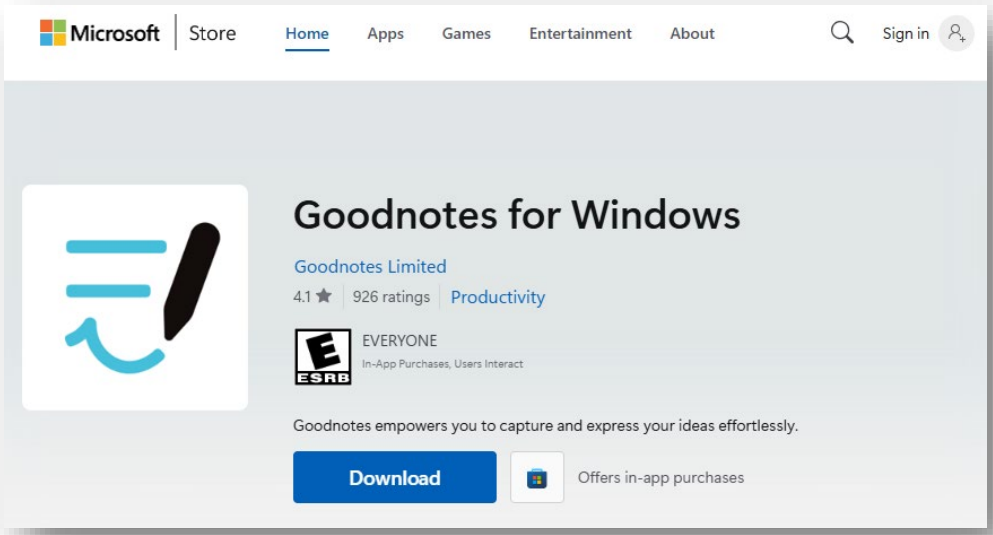


d. Google's Google Play Store (https://play.google.com/store/apps/details?id=com.goodnotes.android.app&utm_source=

web&pcampaignid=pcampaignidMKT-Other-global-all-co-prtnr-py-PartBadge-Mar2515-1) for Goodnotes for Android for use on Android devices:



e. Microsoft Store (https://apps.microsoft.com/detail/9n92mc09db30?hl=en-us&gl=US) for Goodnotes for Windows for use on Windows devices:



8. Goodnotes, directly and through subsidiaries or intermediaries, has purposefully and voluntarily placed the Accused Products into this district and into the stream of commerce with the intention and expectation that the Accused Products will be purchased for use in this district. Goodnotes has imported, offered for sale and sold, and continue to offer for sale and sell, Accused Products for use in this district.

9. Venue is proper in the district under 28 U.S.C. §§ 1391(b), (c), and/or 1400(b). Venue in this district is proper for Goodnotes at least because it is a foreign entity that has committed acts of infringement in this district as detailed throughout this complaint.

BACKGROUND

10. Handwriting recognition technology has advanced significantly in recent years, since its beginnings in the 1990s, providing a more instinctive input method for digital devices. This technology enables users to write on touch-sensitive screens using digital pens or stylus devices, thereby simulating pen-on-paper experience. The innovation of handwriting recognition lies in its ability to convert handwritten input into digital text and shapes, enhancing user interaction with devices through digital ink management. The inclusion of editing capabilities further enriches this technology, allowing for seamless and efficient management of handwritten content

11. Despite its slow initial uptake, the value of handwriting recognition technology has become evident through its recent widespread adoption. Initially, factors such as accuracy, user preferences, and the complexity of the technology contributed to its slower acceptance. However, advancements in machine learning and sensor-based recognition have significantly improved the efficiency and accuracy of handwriting recognition technologies. This has led to a surge in adoption across various industries, with multiple companies integrating handwriting recognition into their products to enhance user experience and productivity.

12. MyScript's innovations are at the forefront of this adoption. The first industrial hybrid system combining artificial neural networks and Markov models was developed by MyScript in the early 2000s. Through consistent research efforts, year after year, MyScript has played a significant role in scientific advancements that have resulted in the development of innovative technology and products by MyScript and others. These innovations have demonstrated that this technology has real-world applications and a viable market. MyScript's innovations have also enabled MyScript to maintain a competitive edge over others in the industry. The widespread licensing of MyScript's technology underscores the industry's recognition of the significant usefulness and value of these innovative features. As more applications and devices incorporate handwriting recognition, its value becomes increasingly apparent.

13. Founded in 1998, MyScript (formally known as "Vision Objects") is a leading company in the field of handwriting recognition technology and renowned for its innovative solutions and extensive patent portfolio. MyScript's handwriting recognition technology enables the conversion of handwritten notes into digital content, enhancing user interaction with digital devices through digital ink management. This technology is particularly valuable for applications requiring instinctive input methods, such as note-taking, diagram creation, and document editing.

14. MyScript's innovative technology solutions are widely recognized as a fundamental aspect of handwriting recognition. MyScript's technology is used by independent software vendors ("ISV") and original equipment manufacturers ("OEM(s)") in a variety of applications and products, serving industries such as consumer electronics, enterprise solutions, automotive and education all over the globe. Notable products include the MyScript Text software development kit ("Text SDK"), MyScript Interactive Ink ("iink") SDK (collectively, "MyScript

SDKs"). MyScript has also developed many business-to-consumer offerings, including two active applications that directly integrate MyScript SDKs, Nebo and MyScript Calculator.

15. MyScript has numerous granted and pending patent applications in different jurisdictions that cover its various technologies. MyScript also employs nearly twenty Ph.D. holders and invests over 40% of its turnover in internal research projects, highlighting its commitment to research and its investment in developing technologies that provide it with a competitive advantage.

16. Goodnotes (formally known as "Time Base Technology Limited") offers digital note-taking applications on various devices, including those running on iOS, Android, and Windows operating systems as set forth above in paragraph 7. One such application is Goodnotes 6 on iOS, which is the latest version of Goodnotes' digital note-taking application. Upon information and belief, Goodnotes launched Goodnotes 6 in August 2023, and it currently has over 25 million monthly active users.

17. Goodnotes 6 includes several features not included in prior versions of Goodnotes' product and which infringe the patents asserted in this action. These features include, but are not limited to, intuitive pen gestures such as scribbling to erase mistakes and circling to select items, a reflow feature that allows users to seamlessly change the size and layout of their handwritten notes while maintaining the visual integrity of the handwritten text, and additional editing features that enhance the app's flexibility and usability.

18. MyScript and Goodnotes had a contractual relationship between 2014 and 2023 during which time MyScript licensed MyScript Text SDK on iOS, Mac, Android and Windows operating systems to Goodnotes for the purposes of development and distribution of the Goodnotes notetaking application. The relationship was governed by a signed licensed agreement and related

amendments ("Agreement"). Throughout this relationship, Goodnotes was aware that MyScript's Text SDK technology is proprietary and that Goodnotes was prohibited under the Agreement from reverse engineering the technology.

19. During the relationship, and upon Goodnotes' request, MyScript provided Goodnotes access to the MyScript iink SDK through MyScript's Partners Portal from May 11, 2021 to August 10, 2021 so Goodnotes could evaluate the suitability and internally test the MyScript iink SDK. Before accessing the MyScript iink SDK through MyScript's Partners Portal, Goodnotes accepted the terms of the MyScript Evaluation Agreement which stated that MyScript retained exclusive ownership of all rights of the MyScript iink SDK and further outlined restrictions on Goodnotes' use of the technology.

20. In September 2022, a Goodnotes employee created an account on the MyScript Developer Portal to test the MyScript iink SDK on device and cloud recognition services. During account creation, the Goodnotes employee accepted the terms and conditions of the MyScript Developer Portal before accessing the technology, which stated that MyScript retained exclusive ownership of all rights of the MyScript iink SDK and further outlined restrictions on the use of the technology.

21. The Agreement's term was fixed until August 5, 2024 and due to renew automatically for periods of one year unless either party gave prior notice. On December 19 2022, the CEO of Goodnotes at that time sent an email to MyScript providing notice that Goodnotes would no longer be integrating MyScript Text SDK in its Goodnotes 5 application because Goodnotes had developed its own handwriting recognition solution. After resolving issues relating to Goodnotes' undeclared and unpaid licenses, the parties mutually terminated the Agreement on May 19, 2023.

22. Upon information and belief, Goodnotes was developing the Goodnotes 6 application during the term of the Agreement.[1]

PATENTS-IN-SUIT

23. This complaint asserts causes of action for infringement of U.S. Patent No. 9,965,175 (the "'175 Patent), U.S. Patent No. 10,082,951 (the "'951 patent"), U.S. Patent No. 11,402,991 (the "'991 patent"), U.S. Patent No. 10,410,316 (the "'316 patent"), U.S. Patent No. 10,181,076 (the "'076 patent"), U.S. Patent No. 10,013,603 (the "'603 patent"), and U.S. Patent No. 9,489,572 (the "'572 patent") (collecting "the Asserted Patents").

24. Each of the Asserted Patents claims patent-eligible subject matter and is a valid and enforceable U.S. patent, the entire right, title, and interest to which MyScript owns by assignment.

U.S. Patent No. 9,965,175

25. On May 8, 2018, after a full and fair examination, the U.S. Patent and Trademark Office ("USPTO") duly and legally issued U.S. Patent No. 9,965,175 (the "'175 patent), entitled SYSTEM AND METHOD OF DIGITAL NOTE TAKING. MyScript is the owner, by assignment, of all rights, title and interest to the '175 patent. Since its issuance, the '175 patent has been in full force and effect. A true and correct copy of the '175 patent is attached as **Exhibit H** hereto.

26. The '175 patent is generally directed to reflowing handwritten text due to resizing of a text block such that the reflow process maintains the visual integrity of the handwritten text, making it easier for users to read and interact with the content on touch-sensitive devices.

27. Goodnotes is not licensed to practice the '175 patent in either an express or implied manner, nor does it enjoy or benefit from any rights in or to the '175 patent whatsoever.

[1] https://www.goodnotes.com/blog/machine-learning-engineer-handwriting-recognition. (June 1, 2023) (last visited April 28, 2025).

U.S. Patent No. 10,082,951

28. On September 25, 2018, after a full and fair examination, the USPTO duly and legally issued U.S. Patent No. 10,082,951 (the "'951 patent"), entitled SYSTEM AND METHOD FOR DIGITAL INK INTERACTIVITY. MyScript is the owner, by assignment, of all rights, title and interest to the '951 patent. A true and correct copy of the '951 patent is attached as **Exhibit I** hereto.

29. The '951 patent is generally directed to mapping digital ink strokes to recognized elements to enhance user experiences by providing dynamic and interactive ways to create and manage hand-drawn text and diagrams on computing devices.

30. Goodnotes is not licensed to practice the '951 patent in either an express or implied manner, nor does it enjoy or benefit from any rights in or to the '951 patent whatsoever.

U.S. Patent No. 11,402,991

31. On August 2, 2022, after a full and fair examination, the USPTO duly and legally U.S. Patent No. 11,402,991 (the "'991 patent"), entitled SYSTEM AND METHOD FOR NOTE TAKING WITH GESTURES. MyScript is the owner, by assignment, of all rights, title and interest to the '991 patent. A true and correct copy of the '991 patent is attached as **Exhibit J** hereto.

32. The '991 patent is generally directed to a system for editing digital documents on computing devices, including methods for processing and displaying digital ink and allowing users to edit handwritten input using gestures on the device's input surface.

33. Goodnotes is not licensed to practice the '991 patent in either an express or implied manner, nor does it enjoy or benefit from any rights in or to the '991 patent whatsoever.

U.S. Patent No. 10,410,316

34. On September 10, 2019, after a full and fair examination, the USPTO duly and legally U.S. Patent No. 10,410,316 (the "'316 patent"), entitled SYSTEM AND METHOD FOR BEAUTIFYING DIGITAL INK. MyScript is the owner, by assignment, of all rights, title and interest to the '316 patent. A true and correct copy of the '316 patent is attached as **Exhibit K** hereto.

35. The '316 patent is generally directed to a system for editing digital documents on computing devices that allows users to refine and enhance the appearance of their handwritten input by selecting and modifying text directly with the cursor, making it visually appealing and easier to read and edit.

36. Goodnotes is not licensed to practice the '316 patent in either an express or implied manner, nor does it enjoy or benefit from any rights in or to the '316 patent whatsoever.

U.S. Patent No. 10,181,076

37. On January 15, 2019, after a full and fair examination, the USPTO duly and legally of U.S. Patent No. 10,181,076 (the "'076 patent"), entitled SYSTEM AND METHOD FOR RECOGNIZING GEOMETRIC SHAPES. MyScript is the owner, by assignment, of all rights, title and interest to the '076 patent. A true and correct copy of the '076 patent is attached as **Exhibit L** hereto.

38. The '076 patent describes a system that can recognize a user's natural drawing of geometric shapes by processing both single-stroke and multi-stroke shapes, applying implicit and user-defined constraints at the vector component level.

39. Goodnotes is not licensed to practice the '076 patent in either an express or implied manner, nor does it enjoy or benefit from any rights in or to the '076 patent whatsoever.

U.S. Patent No. 10,013,603

40. On July 3, 2018, after a full and fair examination, the USPTO duly and legally U.S. Patent No. 10,013,603 (the "'603 patent"), entitled SYSTEM AND METHOD FOR RECOGNIZING MULTIPLE OBJECT STRUCTURE. MyScript is the owner, by assignment, of all rights, title and interest to the '603 patent. A true and correct copy of the '603 patent is attached as **Exhibit M** hereto.

41. The '603 patent describes a system for recognizing an arrangement of multiple objects on computing devices by determining geometric relationships between objects and recognizing handwriting input and related boundaries of that input.

42. Goodnotes is not licensed to practice the '603 patent in either an express or implied manner, nor does it enjoy or benefit from any rights in or to the '603 patent whatsoever.

U.S. Patent No. 9,489,572

43. On November 8, 2016, after a full and fair examination, the USPTO duly and legally of U.S. Patent No. 9,489,572 (the "'572 patent"), entitled SYSTEM AND METHOD FOR RECOGNIZING GEOMETRIC SHAPES. MyScript is the owner, by assignment, of all rights, title and interest to the '572 patent. A true and correct copy of the '572 patent is attached as **Exhibit N** hereto.

44. The '572 patent describes a system that can recognize a user's natural drawing of geometric shapes by processing both single-stroke and multi-stroke shapes, applying implicit and user-defined constraints at the vector component level.

COUNT I
INFRINGEMENT OF U.S. PATENT NO. 9,965,175

45. MyScript incorporates by reference each preceding paragraph as if fully set forth herein.

46. Goodnotes and/or users of the Accused Products have directly infringed (either literally or under the doctrine of equivalents) and continue to directly infringe one or more of the claims of the ’175 patent, in violation of 35 U.S.C. § 271(a), by making, using, offering for sale, and/or selling the Accused Products within the United States and/or importing the Accused Products into the United States without authority or license. A non-limiting example of this infringement is set forth in **Exhibit O**.

47. Additionally, and/or alternatively, Goodnotes has indirectly infringed and continues to indirectly infringe one or more of the claims of the ’175 patent, in violation of 35 U.S.C. § 271(b), by actively inducing users of the Accused Products to directly infringe the one or more claims of the ’175 patent. For example, (a) Goodnotes has had actual knowledge of the ’175 patent or was willfully blind to its existence prior to (at least as early as December 10, 2024), and no later than, the filing of this action; (b) Goodnotes intentionally causes, urges, or encourages users of the Accused Products to directly infringe one or more claims of the ’175 patent by promoting, advertising, and instructing customers and potential customers about the Accused Products and uses of the application, including infringing uses; (c) Goodnotes knows (or should know) that its actions will induce users of the Accused Products directly infringe one or more claims the ’175 patent; and (d) users of the Accused Products directly infringe one or more claims of the ’175 patent. For instance, at a minimum, Goodnotes has supplied and continues to supply the Accused Products to customers while knowing that use of the app will infringe one or more claims of the ’175 patent, and that Goodnotes’ customers then directly infringe one or more claims of the ’175 patent by installing and using the Accused Products in accordance with Goodnotes’ product literature. A non-limiting example of this infringement is set forth in **Exhibit O**.

48. Goodnotes' infringement of the '175 patent is also willful as outlined above because Goodnotes (a) had actual knowledge of the '175 patent or was willfully blind to its existence prior to (at least as early as December 10, 2024), and no later than, the filing of this action; (b) engaged in the aforementioned activity despite an objectively high likelihood that Goodnotes' actions constituted infringement of the '175 patent; and (c) this objectively-defined risk was either known or so obvious that it should have been known to Goodnotes.

49. MyScript is entitled to recover from Goodnotes all damages that MyScript has sustained as a result of Goodnotes' infringement of the '175 patent, including, without limitation, a reasonable royalty and lost profits.

50. Goodnotes' infringement of the '175 patent was and continues to be willful and deliberate, entitling MyScript to enhanced damages.

51. Goodnotes' infringement of the '175 patent is exceptional and entitles MyScript to attorneys' fees and costs incurred in prosecuting this action under 35 U.S.C. § 285.

52. Goodnotes' infringement of the '175 patent has caused irreparable harm (including the loss of market share) to MyScript and will continue to do so unless enjoined by this Court.

COUNT II
INFRINGEMENT OF U.S. PATENT NO. 10,082,951

53. MyScript incorporates by reference each preceding paragraph as if fully set forth herein.

54. Goodnotes and/or users of the Accused Products have directly infringed (either literally or under the doctrine of equivalents) and continue to directly infringe one or more of the claims of the '951 patent, in violation of 35 U.S.C. § 271(a), by making, using, offering for sale, and/or selling the Accused Products within the United States and/or importing the Accused

Products into the United States without authority or license. A non-limiting example of this infringement is set forth in **Exhibit P**.

55. Additionally, and/or alternatively, Goodnotes has indirectly infringed and continues to indirectly infringe one or more of the claims of the ’951 patent, in violation of 35 U.S.C. § 271(b), by actively inducing users of the Accused Products to directly infringe the one or more claims of the ’951 patent. For example, (a) Goodnotes has had actual knowledge of the ’951 patent or was willfully blind to its existence prior to (at least as early as December 10, 2024), and no later than, the filing of this action; (b) Goodnotes intentionally causes, urges, or encourages users of the Accused Products to directly infringe one or more claims of the ’951 patent by promoting, advertising, and instructing customers and potential customers about the Accused Products and uses of the application, including infringing uses; (c) Goodnotes knows (or should know) that its actions will induce users of the Accused Products directly infringe one or more claims the ’951 patent; and (d) users of the Accused Products directly infringe one or more claims of the ’951 patent. For instance, at a minimum, Goodnotes has supplied and continues to supply the Accused Products to customers while knowing that use of the app will infringe one or more claims of the ’951 patent, and that Goodnotes’ customers then directly infringe one or more claims of the ’951 patent by installing and using the Accused Products in accordance with Goodnotes’ product literature. A non-limiting example of this infringement is set forth in **Exhibit P**.

56. Goodnotes’ infringement of the ’951 patent is also willful as outlined above because Goodnotes (a) had actual knowledge of the ’951 patent or was willfully blind to its existence prior to (at least as early as December 10, 2024), and no later than, the filing of this action; (b) engaged in the aforementioned activity despite an objectively high likelihood that

Goodnotes' actions constituted infringement of the '951 patent; and (c) this objectively-defined risk was either known or so obvious that it should have been known to Goodnotes.

57. MyScript is entitled to recover from Goodnotes all damages that MyScript has sustained as a result of Goodnotes' infringement of the '951 patent, including, without limitation, a reasonable royalty and lost profits.

58. Goodnotes' infringement of the '951 patent was and continues to be willful and deliberate, entitling MyScript to enhanced damages.

59. Goodnotes' infringement of the '951 patent is exceptional and entitles MyScript to attorneys' fees and costs incurred in prosecuting this action under 35 U.S.C. § 285.

60. Goodnotes' infringement of the '951 patent has caused irreparable harm (including the loss of market share) to MyScript and will continue to do so unless enjoined by this Court.

COUNT III
INFRINGEMENT OF U.S. PATENT NO. 11,402,991

61. MyScript incorporates by reference each preceding paragraph as if fully set forth herein.

62. Goodnotes and/or users of the Accused Products have directly infringed (either literally or under the doctrine of equivalents) and continue to directly infringe one or more of the claims of the '991 patent, in violation of 35 U.S.C. § 271(a), by making, using, offering for sale, and/or selling the Accused Products within the United States and/or importing the Accused Products into the United States without authority or license. A non-limiting example of this infringement is set forth in **Exhibit Q**.

63. Additionally, and/or alternatively, Goodnotes has indirectly infringed and continues to indirectly infringe one or more of the claims of the '991 patent, in violation of 35 U.S.C. § 271(b), by actively inducing users of the Accused Products to directly infringe the one or

more claims of the ’991 patent. For example, (a) Goodnotes has had actual knowledge of the ’991 patent or was willfully blind to its existence prior to (at least as early as December 10, 2024), and no later than, the filing of this action; (b) Goodnotes intentionally causes, urges, or encourages users of the Accused Products to directly infringe one or more claims of the ’991 patent by promoting, advertising, and instructing customers and potential customers about the Accused Products and uses of the application, including infringing uses; (c) Goodnotes knows (or should know) that its actions will induce users of the Accused Products directly infringe one or more claims the ’991 patent; and (d) users of the Accused Products directly infringe one or more claims of the ’991 patent. For instance, at a minimum, Goodnotes has supplied and continues to supply the Accused Products to customers while knowing that use of the app will infringe one or more claims of the ’991 patent, and that Goodnotes’ customers then directly infringe one or more claims of the ’991 patent by installing and using the Accused Products in accordance with Goodnotes’ product literature. A non-limiting example of this infringement is set forth in **Exhibit Q**.

64. Goodnotes’ infringement of the ’991 patent is also willful as outlined above because Goodnotes (a) had actual knowledge of the ’991 patent or was willfully blind to its existence prior to (at least as early as December 10, 2024), and no later than, the filing of this action; (b) engaged in the aforementioned activity despite an objectively high likelihood that Goodnotes’ actions constituted infringement of the ’991 patent; and (c) this objectively-defined risk was either known or so obvious that it should have been known to Goodnotes.

65. MyScript is entitled to recover from Goodnotes all damages that MyScript has sustained as a result of Goodnotes’ infringement of the ’991 patent, including, without limitation, a reasonable royalty and lost profits.

66. Goodnotes' infringement of the '991 patent was and continues to be willful and deliberate, entitling MyScript to enhanced damages.

67. Goodnotes' infringement of the '991 patent is exceptional and entitles MyScript to attorneys' fees and costs incurred in prosecuting this action under 35 U.S.C. § 285.

68. Goodnotes' infringement of the '991 patent has caused irreparable harm (including the loss of market share) to MyScript and will continue to do so unless enjoined by this Court.

COUNT IV
INFRINGEMENT OF U.S. PATENT NO. 10,410,316

69. MyScript incorporates by reference each preceding paragraph as if fully set forth herein.

70. Goodnotes and/or users of the Accused Products have directly infringed (either literally or under the doctrine of equivalents) and continue to directly infringe one or more of the claims of the '316 patent, in violation of 35 U.S.C. § 271(a), by making, using, offering for sale, and/or selling the Accused Products within the United States and/or importing the Accused Products into the United States without authority or license. A non-limiting example of this infringement is set forth in **Exhibit R**.

71. Additionally, and/or alternatively, Goodnotes has indirectly infringed and continues to indirectly infringe one or more of the claims of the '316 patent, in violation of 35 U.S.C. § 271(b), by actively inducing users of the Accused Products to directly infringe the one or more claims of the '316 patent. For example, (a) GoodNotes has had actual knowledge of the '316 patent or was willfully blind to its existence prior to (at least as early as December 10, 2024), and no later than, the filing of this action; (b) Goodnotes intentionally causes, urges, or encourages users of the Accused Products to directly infringe one or more claims of the '316 patent by promoting, advertising, and instructing customers and potential customers about the Accused

Products and uses of the application, including infringing uses; (c) Goodnotes knows (or should know) that its actions will induce users of the Accused Products directly infringe one or more claims the '316 patent; and (d) users of the Accused Products directly infringe one or more claims of the '316 patent. For instance, at a minimum, Goodnotes has supplied and continues to supply the Accused Products to customers while knowing that use of the app will infringe one or more claims of the '316 patent, and that Goodnotes' customers then directly infringe one or more claims of the '316 patent by installing and using the Accused Products in accordance with Goodnotes' product literature. A non-limiting example of this infringement is set forth in **Exhibit R**.

72. Goodnotes' infringement of the '316 patent is also willful as outlined above because Goodnotes (a) had actual knowledge of the '316 patent or was willfully blind to its existence prior to (at least as early as December 10, 2024), and no later than, the filing of this action; (b) engaged in the aforementioned activity despite an objectively high likelihood that Goodnotes' actions constituted infringement of the '316 patent; and (c) this objectively-defined risk was either known or so obvious that it should have been known to Goodnotes.

73. MyScript is entitled to recover from Goodnotes all damages that MyScript has sustained as a result of Goodnotes' infringement of the '316 patent, including, without limitation, a reasonable royalty and lost profits.

74. Goodnotes' infringement of the '316 patent was and continues to be willful and deliberate, entitling MyScript to enhanced damages.

75. Goodnotes' infringement of the '316 patent is exceptional and entitles MyScript to attorneys' fees and costs incurred in prosecuting this action under 35 U.S.C. § 285.

76. Goodnotes' infringement of the '316 patent has caused irreparable harm (including the loss of market share) to MyScript and will continue to do so unless enjoined by this Court.

COUNT V
INFRINGEMENT OF U.S. PATENT NO. 10,181,076

77. MyScript incorporates by reference each preceding paragraph as if fully set forth herein.

78. Goodnotes and/or users of the Accused Products have directly infringed (either literally or under the doctrine of equivalents) and continue to directly infringe one or more of the claims of the '076 patent, in violation of 35 U.S.C. § 271(a), by making, using, offering for sale, and/or selling the Accused Products within the United States and/or importing the Accused Products into the United States without authority or license. A non-limiting example of this infringement is set forth in **Exhibit S**.

79. Additionally, and/or alternatively, Goodnotes has indirectly infringed and continues to indirectly infringe one or more of the claims of the '076 patent, in violation of 35 U.S.C. § 271(b), by actively inducing users of the Accused Products to directly infringe the one or more claims of the '076 patent. For example, (a) Goodnotes has had actual knowledge of the '076 patent or was willfully blind to its existence prior to (at least as early as December 10, 2024), and no later than, the filing of this action; (b) Goodnotes intentionally causes, urges, or encourages users of the Accused Products to directly infringe one or more claims of the '076 patent by promoting, advertising, and instructing customers and potential customers about the Accused Products and uses of the application, including infringing uses; (c) Goodnotes knows (or should know) that its actions will induce users of the Accused Products directly infringe one or more claims the '076 patent; and (d) users of the Accused Products directly infringe one or more claims of the '076 patent. For instance, at a minimum, Goodnotes has supplied and continues to supply the Accused Products to customers while knowing that use of the app will infringe one or more claims of the '076 patent, and that Goodnotes' customers then directly infringe one or more claims

of the ’076 patent by installing and using the Accused Products in accordance with Goodnotes’ product literature. A non-limiting example of this infringement is set forth in **Exhibit S**.

80. Goodnotes’ infringement of the ’076 patent is also willful as outlined above because Goodnotes (a) had actual knowledge of the ’076 patent or was willfully blind to its existence prior to (at least as early as December 10, 2024), and no later than, the filing of this action; (b) engaged in the aforementioned activity despite an objectively high likelihood that Goodnotes’ actions constituted infringement of the ’076 patent; and (c) this objectively-defined risk was either known or so obvious that it should have been known to Goodnotes.

81. MyScript is entitled to recover from Goodnotes all damages that MyScript has sustained as a result of Goodnotes’ infringement of the ’076 patent, including, without limitation, a reasonable royalty and lost profits.

82. Goodnotes’ infringement of the ’076 patent was and continues to be willful and deliberate, entitling MyScript to enhanced damages.

83. Goodnotes’ infringement of the ’076 patent is exceptional and entitles MyScript to attorneys’ fees and costs incurred in prosecuting this action under 35 U.S.C. § 285.

84. Goodnotes’ infringement of the ’076 patent has caused irreparable harm (including the loss of market share) to MyScript and will continue to do so unless enjoined by this Court.

COUNT VI
INFRINGEMENT OF U.S. PATENT NO. 10,013,603

85. MyScript incorporates by reference each preceding paragraph as if fully set forth herein.

86. Goodnotes and/or users of the Accused Products have directly infringed (either literally or under the doctrine of equivalents) and continue to directly infringe one or more of the claims of the ’603 patent, in violation of 35 U.S.C. § 271(a), by making, using, offering for sale,

and/or selling the Accused Products within the United States and/or importing the Accused Products into the United States without authority or license. A non-limiting example of this infringement is set forth in **Exhibit T**.

87. Additionally, and/or alternatively, Goodnotes has indirectly infringed and continues to indirectly infringe one or more of the claims of the ’603 patent, in violation of 35 U.S.C. § 271(b), by actively inducing users of the Accused Products to directly infringe the one or more claims of the ’603 patent. For example, (a) Goodnotes has had actual knowledge of the ’603 patent or was willfully blind to its existence prior to (at least as early as December 10, 2024), and no later than, the filing of this action; (b) Goodnotes intentionally causes, urges, or encourages users of the Accused Products to directly infringe one or more claims of the ’603 patent by promoting, advertising, and instructing customers and potential customers about the Accused Products and uses of the application, including infringing uses; (c) Goodnotes knows (or should know) that its actions will induce users of the Accused Products directly infringe one or more claims the ’603 patent; and (d) users of the Accused Products directly infringe one or more claims of the ’603 patent. For instance, at a minimum, Goodnotes has supplied and continues to supply the Accused Products to customers while knowing that use of the app will infringe one or more claims of the ’603 patent, and that Goodnotes’ customers then directly infringe one or more claims of the ’603 patent by installing and using the Accused Products in accordance with Goodnotes’ product literature. A non-limiting example of this infringement is set forth in **Exhibit T**.

88. Goodnotes’ infringement of the ’603 patent is also willful as outlined above because Goodnotes (a) had actual knowledge of the ’603 patent or was willfully blind to its existence prior to (at least as early as December 10, 2024), and no later than, the filing of this action; (b) engaged in the aforementioned activity despite an objectively high likelihood that

Goodnotes' actions constituted infringement of the '603 patent; and (c) this objectively-defined risk was either known or so obvious that it should have been known to Goodnotes.

89. MyScript is entitled to recover from Goodnotes all damages that MyScript has sustained as a result of Goodnotes' infringement of the '603 patent, including, without limitation, a reasonable royalty and lost profits.

90. Goodnotes' infringement of the '603 patent was and continues to be willful and deliberate, entitling MyScript to enhanced damages.

91. Goodnotes' infringement of the '603 patent is exceptional and entitles MyScript to attorneys' fees and costs incurred in prosecuting this action under 35 U.S.C. § 285.

92. Goodnotes' infringement of the '603 patent has caused irreparable harm (including the loss of market share) to MyScript and will continue to do so unless enjoined by this Court.

COUNT VII
INFRINGEMENT OF U.S. PATENT NO. 9,489,572

93. MyScript incorporates by reference each preceding paragraph as if fully set forth herein.

94. Goodnotes and/or users of the Accused Products have directly infringed (either literally or under the doctrine of equivalents) and continue to directly infringe one or more of the claims of the '572 patent, in violation of 35 U.S.C. § 271(a), by making, using, offering for sale, and/or selling the Accused Products within the United States and/or importing the Accused Products into the United States without authority or license. A non-limiting example of this infringement is set forth in **Exhibit U**.

95. Additionally, and/or alternatively, Goodnotes has indirectly infringed and continues to indirectly infringe one or more of the claims of the '572 patent, in violation of 35 U.S.C. § 271(b), by actively inducing users of the Accused Products to directly infringe the one or

more claims of the ’572 patent. For example, (a) Goodnotes has had actual knowledge of the ’572 patent or was willfully blind to its existence prior to (at least as early as December 10, 2024), and no later than, the filing of this action; (b) Goodnotes intentionally causes, urges, or encourages users of the Accused Products to directly infringe one or more claims of the ’572 patent by promoting, advertising, and instructing customers and potential customers about the Accused Products and uses of the application, including infringing uses; (c) Goodnotes knows (or should know) that its actions will induce users of the Accused Products directly infringe one or more claims the ’572 patent; and (d) users of the Accused Products directly infringe one or more claims of the ’572 patent. For instance, at a minimum, Goodnotes has supplied and continues to supply the Accused Products to customers while knowing that use of the app will infringe one or more claims of the ’572 patent, and that Goodnotes’ customers then directly infringe one or more claims of the ’572 patent by installing and using the Accused Products in accordance with Goodnotes’ product literature. A non-limiting example of this infringement is set forth in **Exhibit U**.

96. Goodnotes’ infringement of the ’572 patent is also willful as outlined above because Goodnotes (a) had actual knowledge of the ’572 patent or was willfully blind to its existence prior to (at least as early as December 10, 2024), and no later than, the filing of this action; (b) engaged in the aforementioned activity despite an objectively high likelihood that Goodnotes’ actions constituted infringement of the ’572 patent; and (c) this objectively-defined risk was either known or so obvious that it should have been known to Goodnotes.

97. MyScript is entitled to recover from Goodnotes all damages that MyScript has sustained as a result of Goodnotes’ infringement of the ’572 patent, including, without limitation, a reasonable royalty and lost profits.

98. Goodnotes' infringement of the '572 patent was and continues to be willful and deliberate, entitling MyScript to enhanced damages.

99. Goodnotes' infringement of the '572 patent is exceptional and entitles MyScript to attorneys' fees and costs incurred in prosecuting this action under 35 U.S.C. § 285.

100. Goodnotes' infringement of the '572 patent has caused irreparable harm (including the loss of market share) to MyScript and will continue to do so unless enjoined by this Court.

PRAYER FOR RELIEF

WHEREFORE, MyScript respectfully requests the following relief from this Court:

A. That Judgment be entered that Defendant has infringed and continues to infringe at least one or more claims of each of the Asserted Patents, directly and/or indirectly, literally and/or under the doctrine of equivalents, and that such infringement has been and is willful;

B. An injunction enjoining Defendant, its officers, agents, servants, employees and attorneys, and other persons in active concert or participation with Defendant, and its parents, subsidiaries, divisions, successors and assigns, from further infringement of the Asserted Patents;

C. An award of damages sufficient to compensate MyScript for Defendant's infringement under 35 U.S.C. § 284, including an enhancement of damages on account of Defendant's willful infringement;

D. That the case be found exceptional under 35 U.S.C. § 285 and that MyScript be awarded its reasonable attorneys' fees;

E. Costs and expenses in this action;

F. An award of prejudgment and post-judgment interest; and

G. Such other and further relief as the Court may deem just and proper.

REQUEST FOR JURY TRIAL

Pursuant to Rule 38 of the Federal Rules of Civil Procedure, MyScript requests a trial by jury of any issues so triable by right.

Dated: May 15, 2025

Respectfully Submitted,

/s/ Deron R. Dacus

Deron R. Dacus
State Bar No. 00790553
THE DACUS FIRM, P.C.
821 ESE Loop 323, Suite 430
Tyler, TX 75701
(903) 705-7233
ddacus@dacusfirm.com
Counsel for Plaintiff
MyScript SAS

OF COUNSEL:

Amol A. Parikh
Ashley K. Justice
MCDERMOTT WILL & EMERY LLP
444 W Lake Street
Chicago, Illinois 60606
(312) 984-6477
amparikh@mwe.com
akjustice@mwe.com

EXHIBIT A

luma+ Sign In



Goodnotes at University of Houston

NOV 17 **Sunday, November 17, 2024**
11:00 AM - 4:00 PM

University of Houston
Houston, Texas

Registration

Past Event
This event ended 170 days ago.

Welcome! To join the event, please register below.

Register

Presented by
GoodEvents by Goo... Subscribe

Welcome to the ultimate hub for all things events! 🎉 Whether hosted by our amazing ambassadors or directly by Goodnotes, this is your go-to spot

Hosted By

B

24 Went

Yvette Djopong, Aiman Shekoh and 22 others

Contact the Host

Report Event

About Event

Goodnotes is coming to University of Houston

Sign up for double entries in the Goodnotes Lucky Draw and win big prizes!

Play & Win 🏆

Play games to get a feel for how natural and flexible digital ink in Goodnotes is—and to earn a chance to win prizes!

Win an iPad Air 🎁

Win Goodnotes merch or even an iPad! At least one student from each school will win a brand-new iPad Air.

Win a free subscription 📲

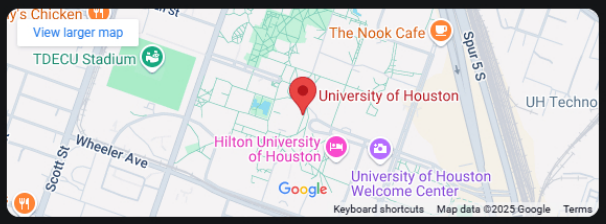
Play our lucky draw to win a free yearly Goodnotes 6 subscription. There are 5 subscriptions that are available for students to win at each university.

Plant trees! 🌳

For every 10 downloads of Goodnotes at your school, we'll plant a tree! We'll have a leaderboard on-site to see which college has planted the most trees.

Location

University of Houston
4302 University Dr, Houston, TX 77004, USA



What's New Discover Pricing Help

Host your event with Luma ↗

EXHIBIT B

luma

Sign In



Goodnotes at Texas A&M University

NOV 20 Wednesday, November 20, 2024
4:00 PM - 8:15 PM

Texas A&M University
College Station, Texas

Registration

Past Event
This event ended 167 days ago.

Welcome! To join the event, please register below.

Register

Presented by
GoodEvents by Goo... Subscribe

Welcome to the ultimate hub for all things events! 🎉 Whether hosted by our amazing ambassadors or directly by Goodnotes, this is your go-to spot

Hosted By

B

18 Went

Ryley Priddy, Madison Ngo and 16 others

Contact the Host

Report Event

About Event

Goodnotes is coming to Texas A&M University

Sign up for double entries in the Goodnotes Lucky Draw and win big prizes

Play & Win 🏆

Play games to get a feel for how natural and flexible digital ink in Goodnotes is—and to earn a chance to win prizes!

Win an iPad Air 🎁

Win Goodnotes merch or even an iPad! At least one student from each school will win a brand-new iPad Air.

Win a free subscription 📲

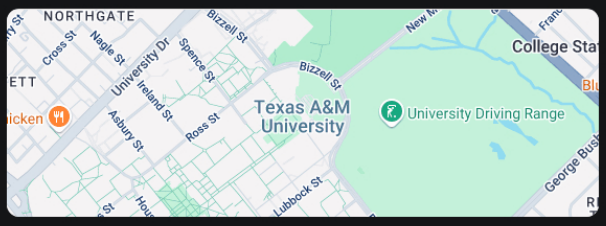
Play our lucky draw to win a free yearly Goodnotes 6 subscription. There are 5 subscriptions that are available for students to win at each university.

Plant trees! 🌳

For every 10 downloads of Goodnotes at your school, we'll plant a tree! We'll have a leaderboard on-site to see which college has planted the most trees.

Location

Texas A&M University
400 Bizzell St, College Station, TX 77840, USA



What's New Discover Pricing Help

Host your event with Luma

EXHIBIT C

luma

Sign In



Goodnotes at University of Texas, Austin

NOV 21 **Thursday, November 21, 2024**
9:00 AM - 7:00 PM

309 E 21st St
Austin, Texas

Registration

Past Event
This event ended 166 days ago.

Welcome! To join the event, please register below.

Register

Presented by
GoodEvents by Goo... Subscribe

Welcome to the ultimate hub for all things events! 🎉 Whether hosted by our amazing ambassadors or directly by Goodnotes, this is your go-to spot

Hosted By

B

184 Went

Darsh Pathak, Chris Guinn and 182 others

Contact the Host
Report Event

About Event

Goodnotes is coming to University of Texas, Austin

Sign up for double entries in the Goodnotes Lucky Draw and win big prizes

Play & Win 🏆

Play games to get a feel for how natural and flexible digital ink in Goodnotes is—and to earn a chance to win prizes!

Win an iPad Air 🎁

Win Goodnotes merch or even an iPad! At least one student from each school will win a brand-new iPad Air.

Win a free subscription 🏴

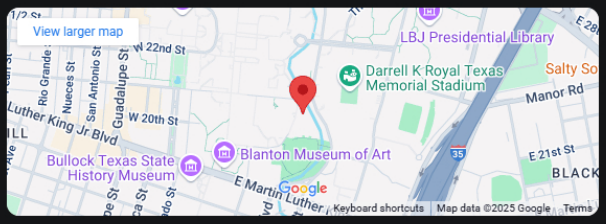
Play our lucky draw to win a free yearly Goodnotes 6 subscription. There are 5 subscriptions that are available for students to win at each university.

Plant trees! 🌱

For every 10 downloads of Goodnotes at your school, we'll plant a tree! We'll have a leaderboard on-site to see which college has planted the most trees.

Location

309 E 21st St
Austin, TX 78712, USA



What's New Discover Pricing Help

Host your event with Luma

EXHIBIT D

luma

Sign In



Goodnotes at Texas State University

NOV 22 Friday, November 22, 2024
5:00 PM - 9:00 PM

3875 Holman St
Houston, Texas

Registration

Past Event
This event ended 165 days ago.

Welcome! To join the event, please register below.

Register

Presented by
GoodEvents by Goo... Subscribe

Welcome to the ultimate hub for all things events! 🎉 Whether hosted by our amazing ambassadors or directly by Goodnotes, this is your go-to spot

Hosted By

B

7 Went

Mustafa Abed, Gladis Santos and 5 others

Contact the Host

Report Event

About Event

Goodnotes is coming to Texas State University

Sign up for double entries in the Goodnotes Lucky Draw and win big prizes

Play & Win 🏆

Play games to get a feel for how natural and flexible digital ink in Goodnotes is—and to earn a chance to win prizes!

Win an iPad Air 🎁

Win Goodnotes merch or even an iPad! At least one student from each school will win a brand-new iPad Air.

Win a free subscription 🗓️

Play our lucky draw to win a free yearly Goodnotes 6 subscription. There are 5 subscriptions that are available for students to win at each university.

Plant trees! 🌳

For every 10 downloads of Goodnotes at your school, we'll plant a tree! We'll have a leaderboard on-site to see which college has planted the most trees.

Location

3875 Holman St
Houston, TX 77004, USA

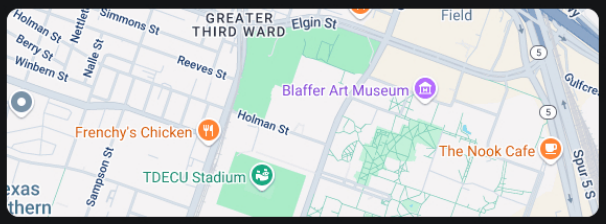


EXHIBIT E

luma+ Sign In



Goodnotes at University of Texas, Arlington

JAN 13 **Monday, January 13**
9:00 AM - 6:00 PM

300 W 1st St
Arlington, Texas

Registration

Past Event
This event ended 113 days ago.

Welcome! To join the event, please register below.

Register

Presented by
GoodEvents by Goo... Subscribe

Welcome to the ultimate hub for all things events! 🎉 Whether hosted by our amazing ambassadors or directly by Goodnotes, this is your go-to spot

Hosted By

B

26 Went

Oyindamola Ogunhibe, Morgan Westbrook and 24 others

Contact the Host
Report Event

About Event

Goodnotes is coming to University of Texas, Arlington

Sign up for double entries in the Goodnotes Lucky Draw and win big prizes

Play & Win 🏆

Play games to get a feel for how natural and flexible digital ink in Goodnotes is—and to earn a chance to win prizes!

Win an iPad Air 🎁

Win Goodnotes merch or even an iPad! At least one student from each school will win a brand-new iPad Air.

Win a free subscription 🗹

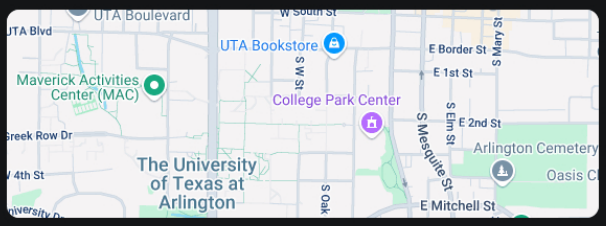
Play our lucky draw to win a free yearly Goodnotes 6 subscription. There are 5 subscriptions that are available for students to win at each university.

Plant trees! 🌳

For every 10 downloads of Goodnotes at your school, we'll plant a tree! We'll have a leaderboard on-site to see which college has planted the most trees.

Location

300 W 1st St
Arlington, TX 76019, USA



What's New Discover Pricing Help

Host your event with Luma

EXHIBIT F

luma

Sign In



Goodnotes at Texas Christian University

JAN 14 **Tuesday, January 14**
9:00 AM - 6:00 PM

2800 S University Dr
Fort Worth, Texas

Registration

Past Event
This event ended 112 days ago.

Welcome! To join the event, please register below.

Register

Presented by
GoodEvents by Goo... Subscribe

Welcome to the ultimate hub for all things events! 🎉 Whether hosted by our amazing ambassadors or directly by Goodnotes, this is your go-to spot

Hosted By

B

9 Went

Oyindamola Ogunhibe, Wakati and 7 others

Contact the Host
Report Event

About Event

Goodnotes is coming to Texas Christian University

Sign up for double entries in the Goodnotes Lucky Draw and win big prizes

Play & Win 🏆

Play games to get a feel for how natural and flexible digital ink in Goodnotes is—and to earn a chance to win prizes!

Win an iPad Air 🎁

Win Goodnotes merch or even an iPad! At least one student from each school will win a brand-new iPad Air.

Win a free subscription

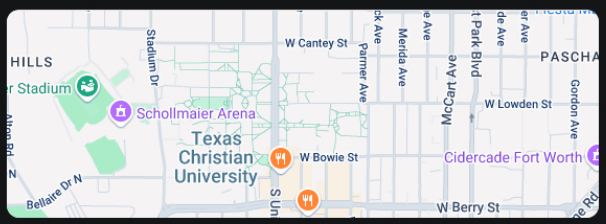
Play our lucky draw to win a free yearly Goodnotes 6 subscription. There are 5 subscriptions that are available for students to win at each university.

Plant trees! 🌳

For every 10 downloads of Goodnotes at your school, we'll plant a tree! We'll have a leaderboard on-site to see which college has planted the most trees.

Location

2800 S University Dr
Fort Worth, TX 76109, USA



What's New Discover Pricing Help

Host your event with Luma

EXHIBIT G

luma

Sign In



Presented by
GoodEvents by Goo...

Subscribe

Welcome to the ultimate hub for all things events! 🎉 Whether hosted by our amazing ambassadors or directly by Goodnotes, this is your go-to spot

Hosted By

B

12 Went

Kai Hall, Neida and 10 others

Contact the Host

Report Event

Goodnotes at Texas Woman's University

JAN 15 **Wednesday, January 15**
9:00 AM - 4:00 PM

Hubbard Hall
Denton, Texas

Registration

Past Event
This event ended 111 days ago.

Welcome! To join the event, please register below.

Register

About Event

Goodnotes is coming to Texas Woman's University.

Sign up for double entries in the Goodnotes Lucky Draw and win big prizes

Play & Win 🏆

Play games to get a feel for how natural and flexible digital ink in Goodnotes is—and to earn a chance to win prizes!

Win an iPad Air 🎁

Win Goodnotes merch or even an iPad! At least one student from each school will win a brand-new iPad Air.

Win a free subscription 📲

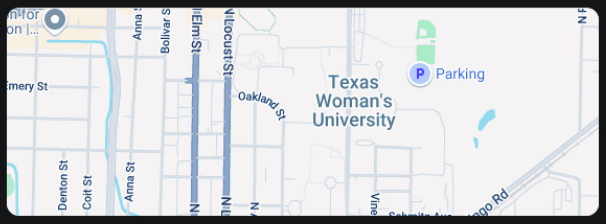
Play our lucky draw to win a free yearly Goodnotes 6 subscription. There are 5 subscriptions that are available for students to win at each university.

Plant trees! 🌳

For every 10 downloads of Goodnotes at your school, we'll plant a tree! We'll have a leaderboard on-site to see which college has planted the most trees.

Location

Hubbard Hall
301 Administration Dr, Denton, TX 76201, USA



What's New Discover Pricing Help

Host your event with Luma

EXHIBIT H

US009965175B2

(12) **United States Patent**
Rucine et al.

(10) **Patent No.: US 9,965,175 B2**
(45) **Date of Patent: May 8, 2018**

(54) **SYSTEM AND METHOD OF DIGITAL NOTE TAKING**

(71) Applicant: **MyScript**, Nantes (FR)

(72) Inventors: **Nicolas Rucine**, Nantes (FR); **Nathalie Delbecque**, Nantes (FR); **Robin Mélinand**, Nantes (FR); **Arnoud Boekhoorn**, Nantes (FR); **Cédric Coulon**, Nantes (FR); **François Bourlieux**, Nantes (FR); **Thomas Penin**, Nantes (FR); **Aristote Laval**, Nantes (FR)

(73) Assignee: **MyScript**, Nantes (FR)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 22 days.

(21) Appl. No.: **14/886,195**

(22) Filed: **Oct. 19, 2015**

(65) **Prior Publication Data**

US 2017/0060819 A1 Mar. 2, 2017

(30) **Foreign Application Priority Data**

Aug. 25, 2015 (EP) 15290219

(51) **Int. Cl.**
G06F 17/00 (2006.01)
G06F 3/0488 (2013.01)
(Continued)

(52) **U.S. Cl.**
CPC ***G06F 3/04883*** (2013.01); ***G06F 3/0482*** (2013.01); ***G06F 3/0485*** (2013.01);
(Continued)

(58) **Field of Classification Search**
CPC G06F 17/241; G06F 17/24; G06F 3/03545; G06F 3/04883; G06F 3/0481; G06K 9/00865; G06K 9/222
(Continued)

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,590,257 A 12/1996 Forcier
5,596,350 A 1/1997 Capps et al.
(Continued)

FOREIGN PATENT DOCUMENTS

EP 0720084 7/1996

OTHER PUBLICATIONS

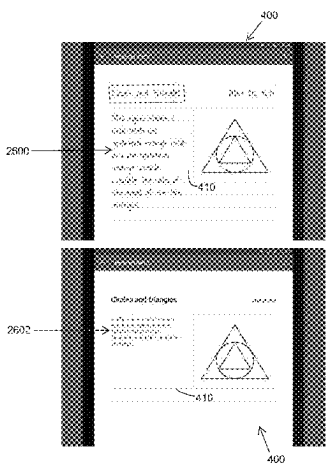
International Search Report for Related Application No. PCT/EP216/001446, dated Nov. 29, 2016, EPO—International, WPI Data.
(Continued)

Primary Examiner — Quoc A Tran
(74) *Attorney, Agent, or Firm* — McDermott Will & Emery LLP

(57) **ABSTRACT**

A system, method and computer program product for use in digital note taking with handwriting input to a computing device are provided. The computing device is connected to an input device in the form of an input surface. A user is able to provide input by applying pressure to or gesturing above the input surface using either his or her finger or an instrument such as a stylus or pen. The present system and method monitors the input strokes. The computing device further has a processor and at least one application for recognizing the handwriting input under control of the processor. The at least one system application is configured to cause display of, on a display interface of a computing device, digital ink in a block layout in accordance with a layout of blocks of the handwriting input and a configuration of the computing device display interface.

18 Claims, 43 Drawing Sheets



(51) **Int. Cl.**

G06K 9/00	(2006.01)
G06F 3/0482	(2013.01)
G06F 3/0484	(2013.01)
G06F 3/0485	(2013.01)
G06F 17/21	(2006.01)
G06F 17/24	(2006.01)
G06K 9/22	(2006.01)
G06F 3/0481	(2013.01)
G06F 3/0354	(2013.01)

(52) **U.S. Cl.**
CPC ***G06F 3/04845*** (2013.01); ***G06F 17/211*** (2013.01); ***G06F 17/242*** (2013.01); ***G06K 9/00402*** (2013.01); *G06F 3/03545* (2013.01); *G06F 3/0481* (2013.01); *G06F 17/24* (2013.01); *G06F 17/241* (2013.01); *G06K 9/00865* (2013.01); *G06K 9/222* (2013.01)

(58) **Field of Classification Search**
USPC 715/268, 762, 763, 784–786
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,613,019	A	3/1997	Altman et al.	
7,266,769	B2	9/2007	Thacker	
7,616,333	B2	11/2009	Wakeam et al.	
7,751,623	B1	7/2010	Simmons et al.	
2014/0363074	A1*	12/2014	Dolfing	G06K 9/00979 382/156
2017/0213069	A1*	7/2017	Giron Espon	B42D 25/318

OTHER PUBLICATIONS

"MyScript—The Power of Handwriting", Dec. 11, 2014, pp. 1-3. URL: https://www.youtube.com/watch?v=yojFiBhPMUo.
aha!™ InkWriter™ Handbook, aha! Software Corporation, 1993.

* cited by examiner

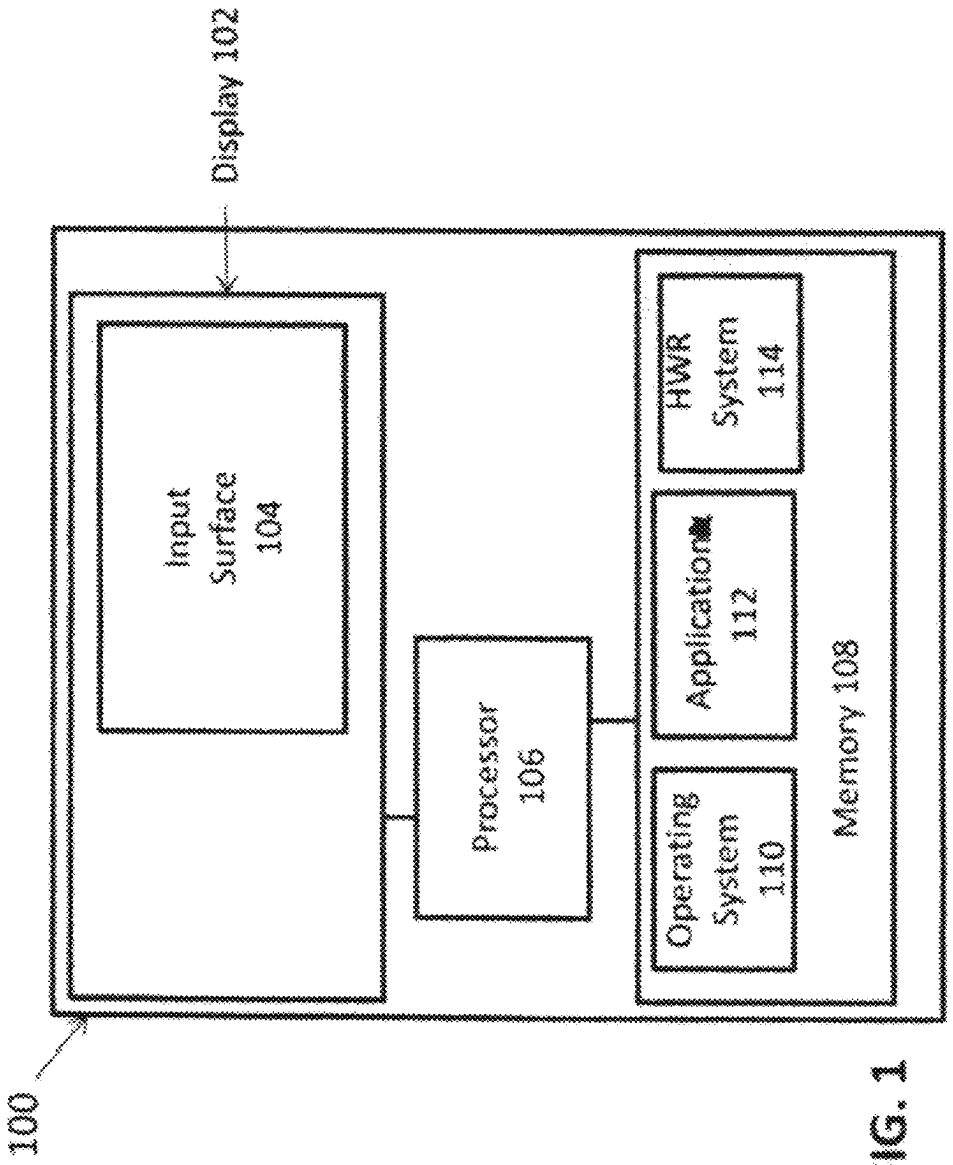

Display 102
Input Surface 104
Processor 106
HWR System 114
Application 112
Operating System 110
Memory 108
100

FIG. 1

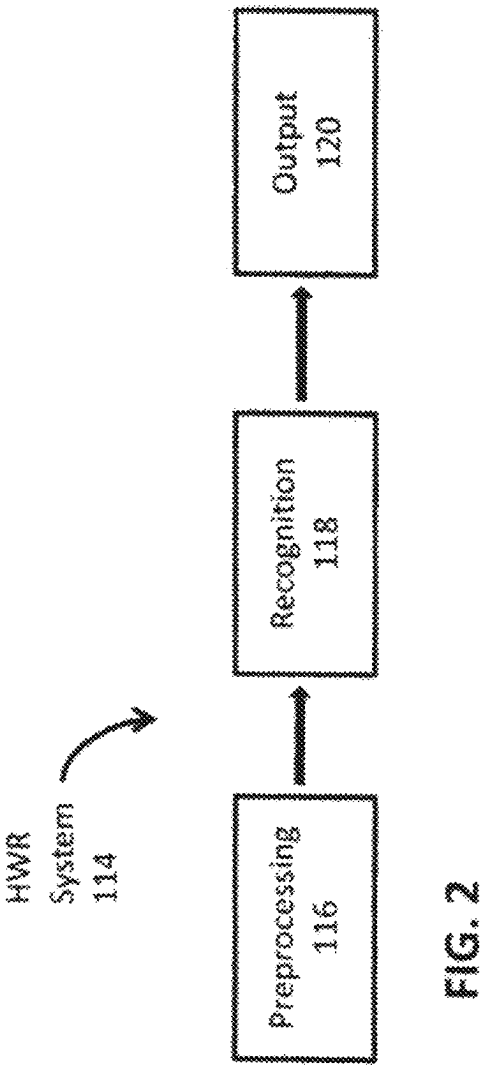

HWR System 114
Preprocessing 116
Recognition 118
Output 120

FIG. 2

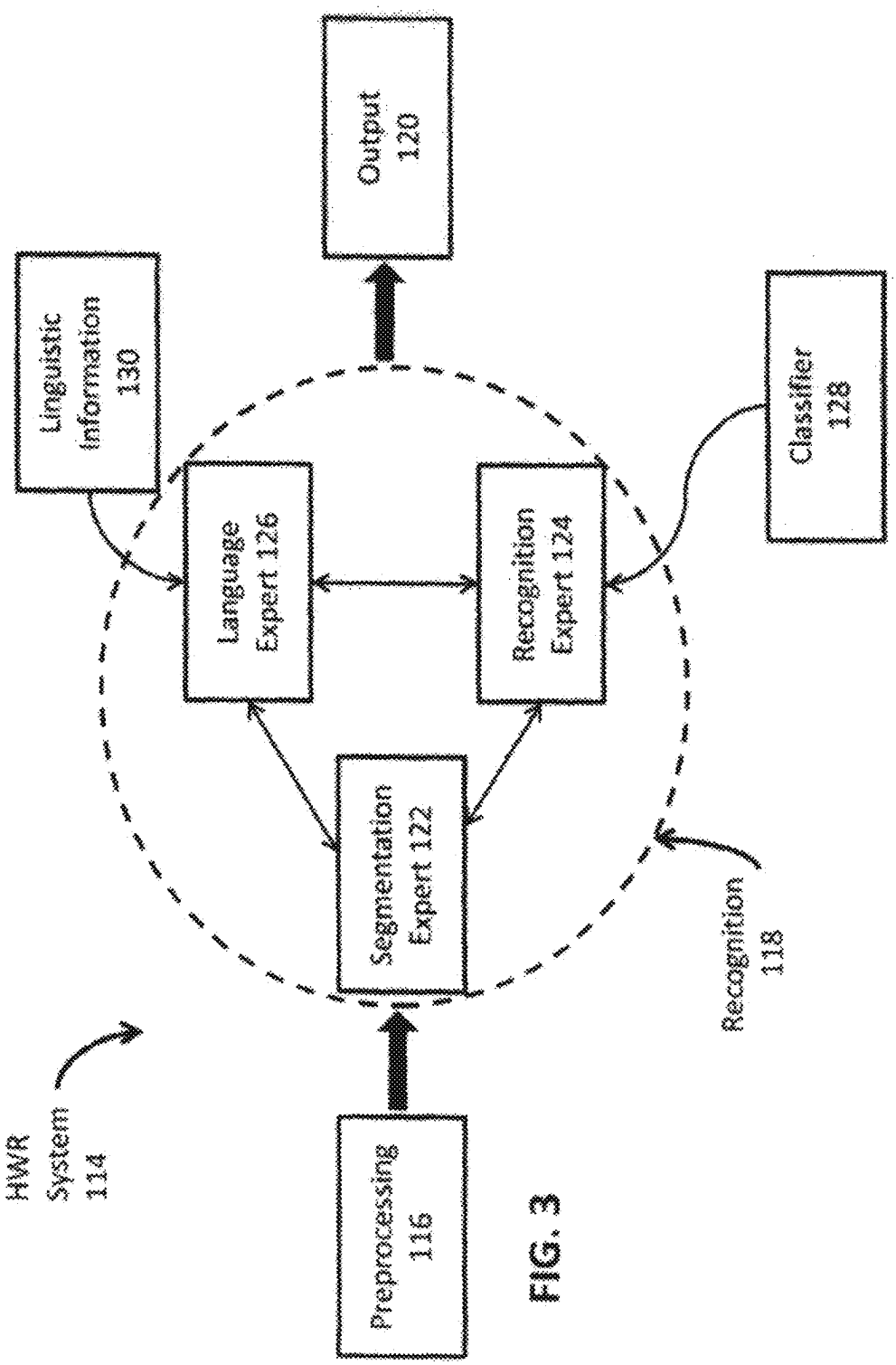

Output
120
Linguistic
Information
130
Classifier
128
Language
Expert 126
Recognition
Expert 124
Segmentation
Expert 122
Recognition
118
HWR
System
114
Preprocessing
116

FIG. 3

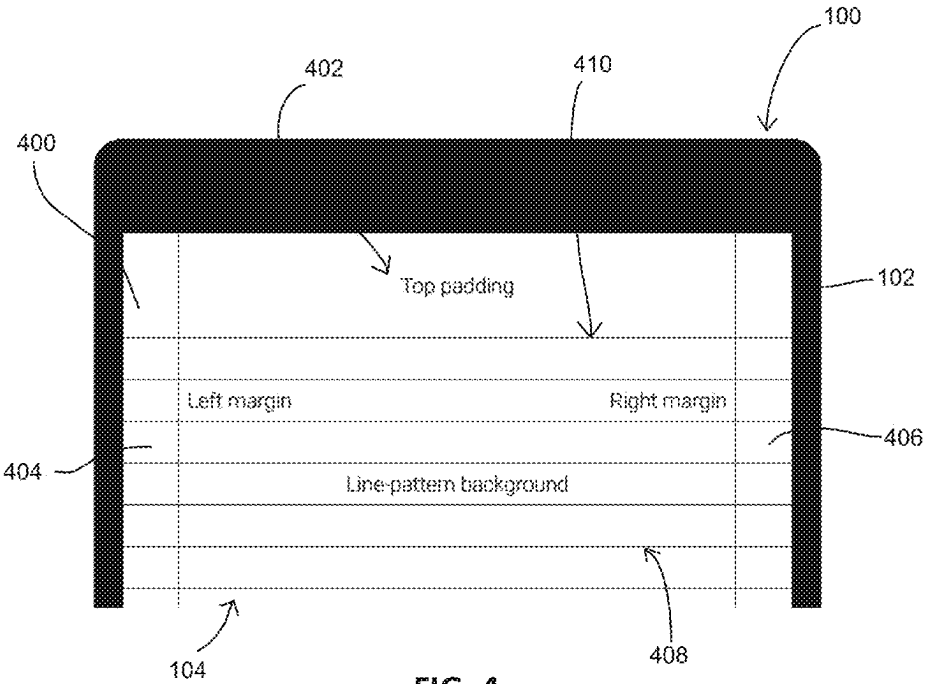

100
402
410
400
Top padding
102
Left margin
Right margin
406
404
Line-pattern background
104
408

FIG. 4

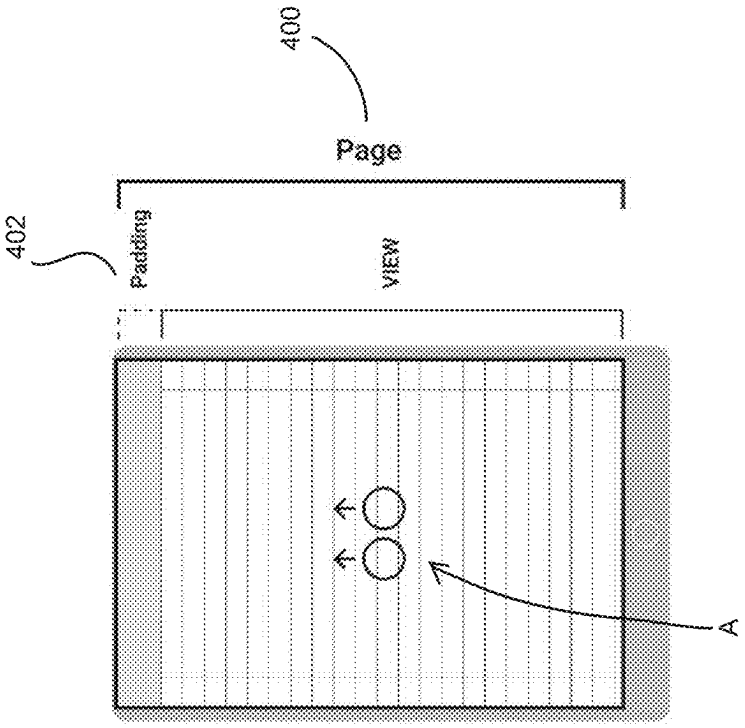

400
Page
402
Padding
VIEW
A

FIG. 5B

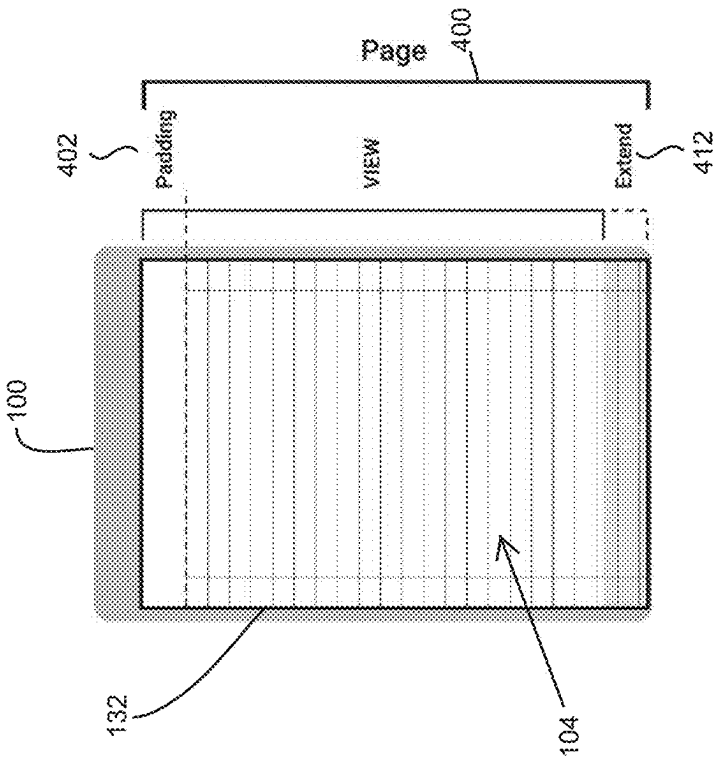

Page
400
402
Padding
VIEW
Extend
412
100
132
104

FIG. 5A

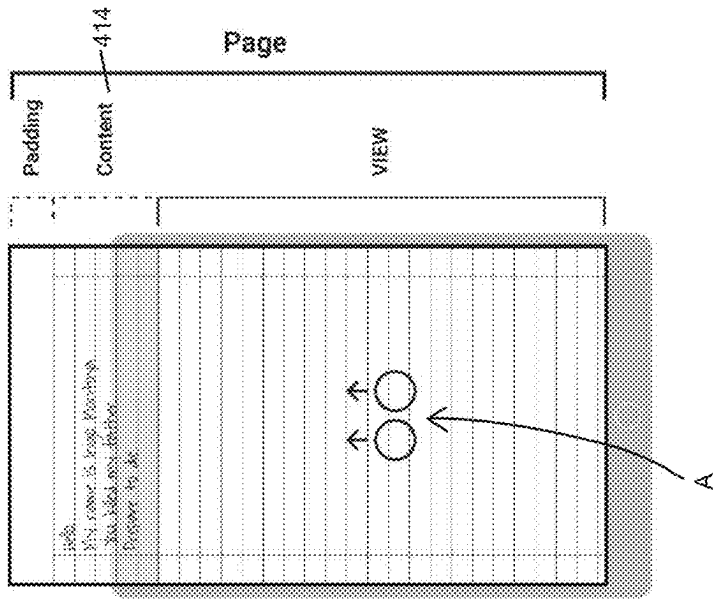

Page
414
Padding
Content
VIEW
A

FIG. 6B

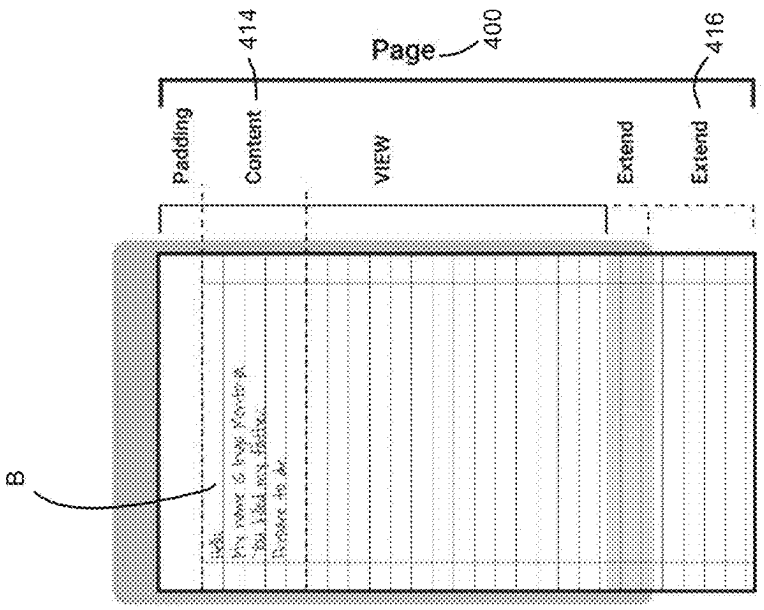

Page
400
414
416
Padding
Content
VIEW
Extend
Extend
B

FIG. 6A

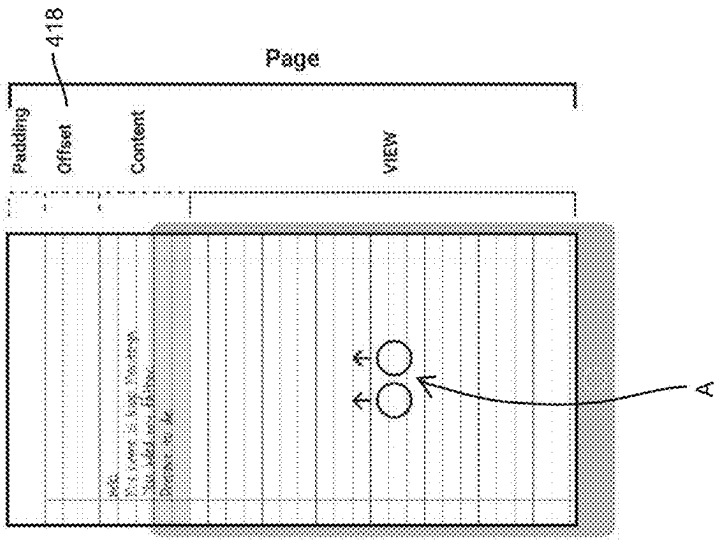

418
Page
Padding
Offset
Content
VIEW
A

FIG. 7B

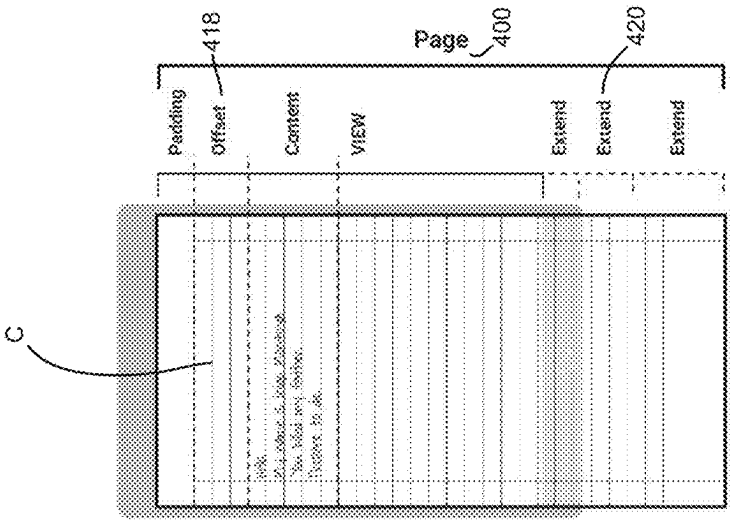

418
Page
400
420
Padding
Offset
Content
VIEW
Extend
Extend
Extend
C

FIG. 7A

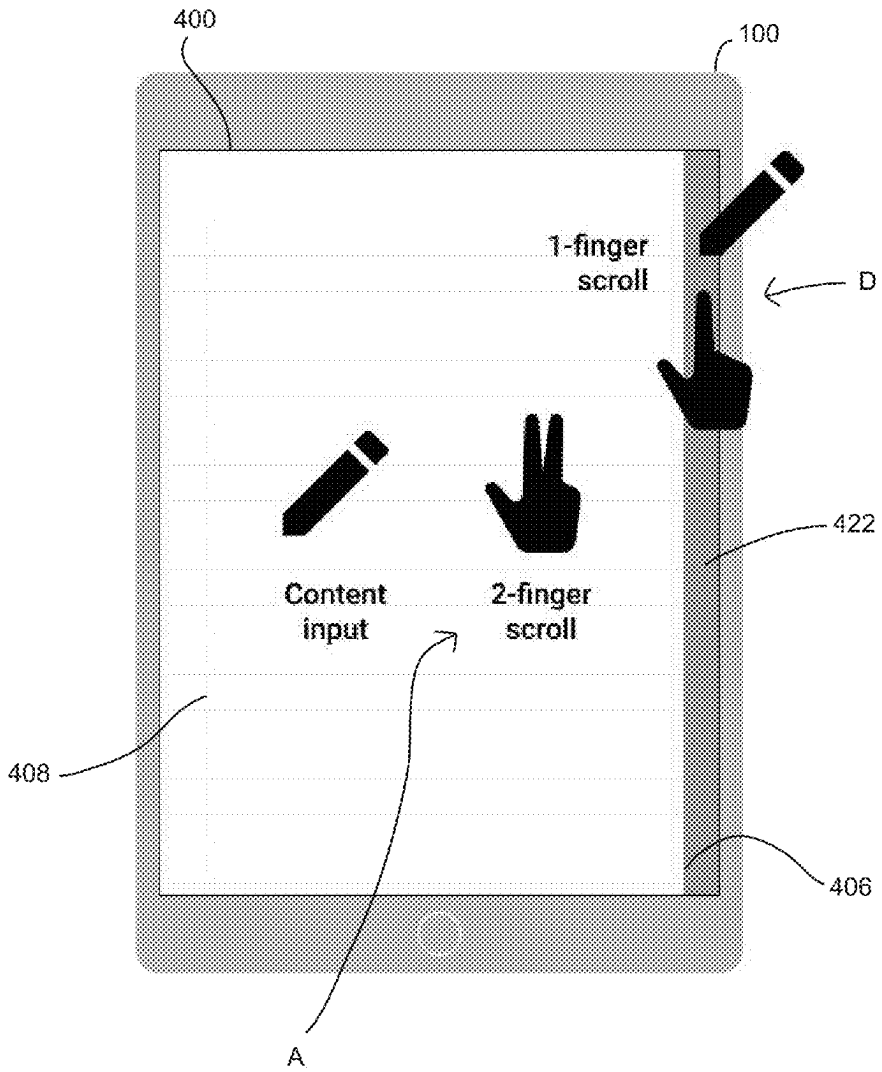

400
100
1-finger
scroll
D
Content
input
2-finger
scroll
422
408
406
A

FIG. 8

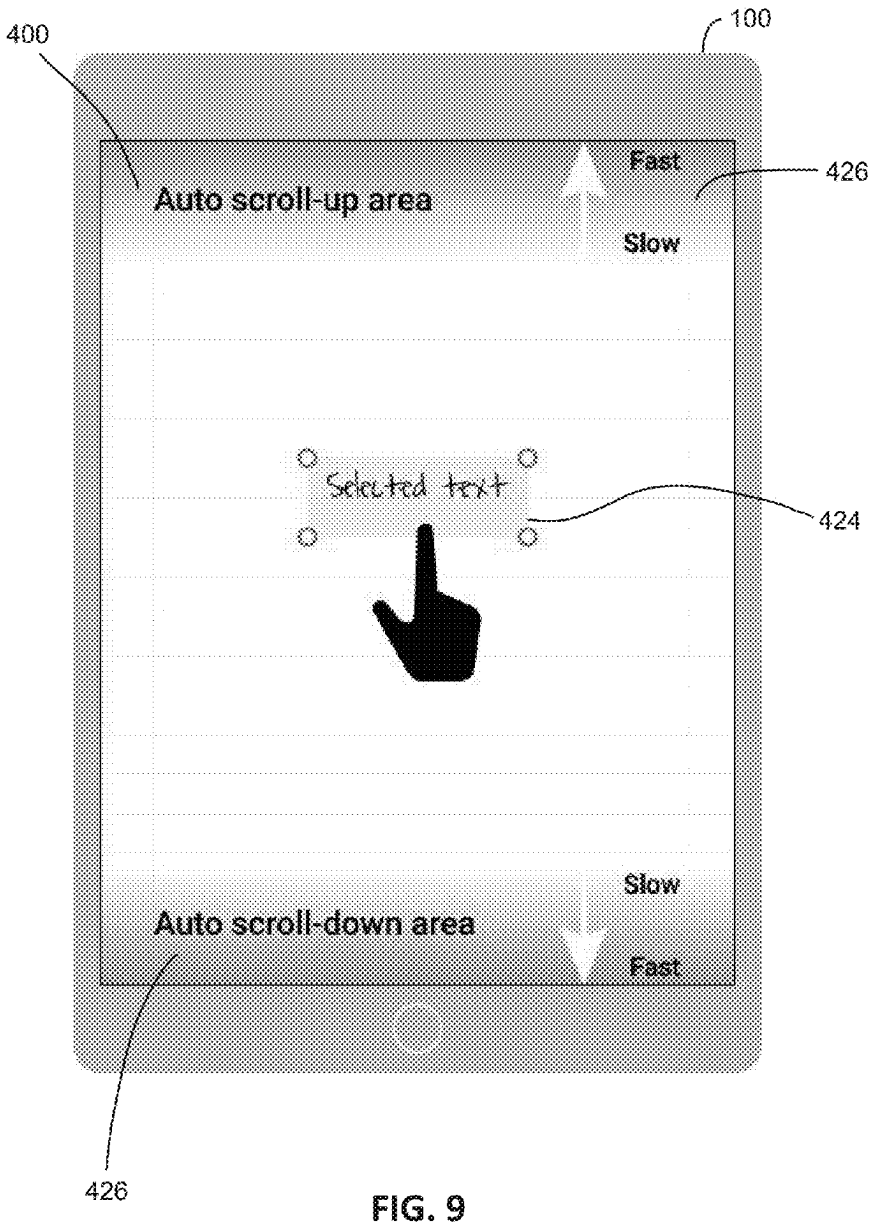

400
100
Auto scroll-up area
Fast
Slow
426
Selected text
424
Slow
Auto scroll-down area
Fast
426

FIG. 9

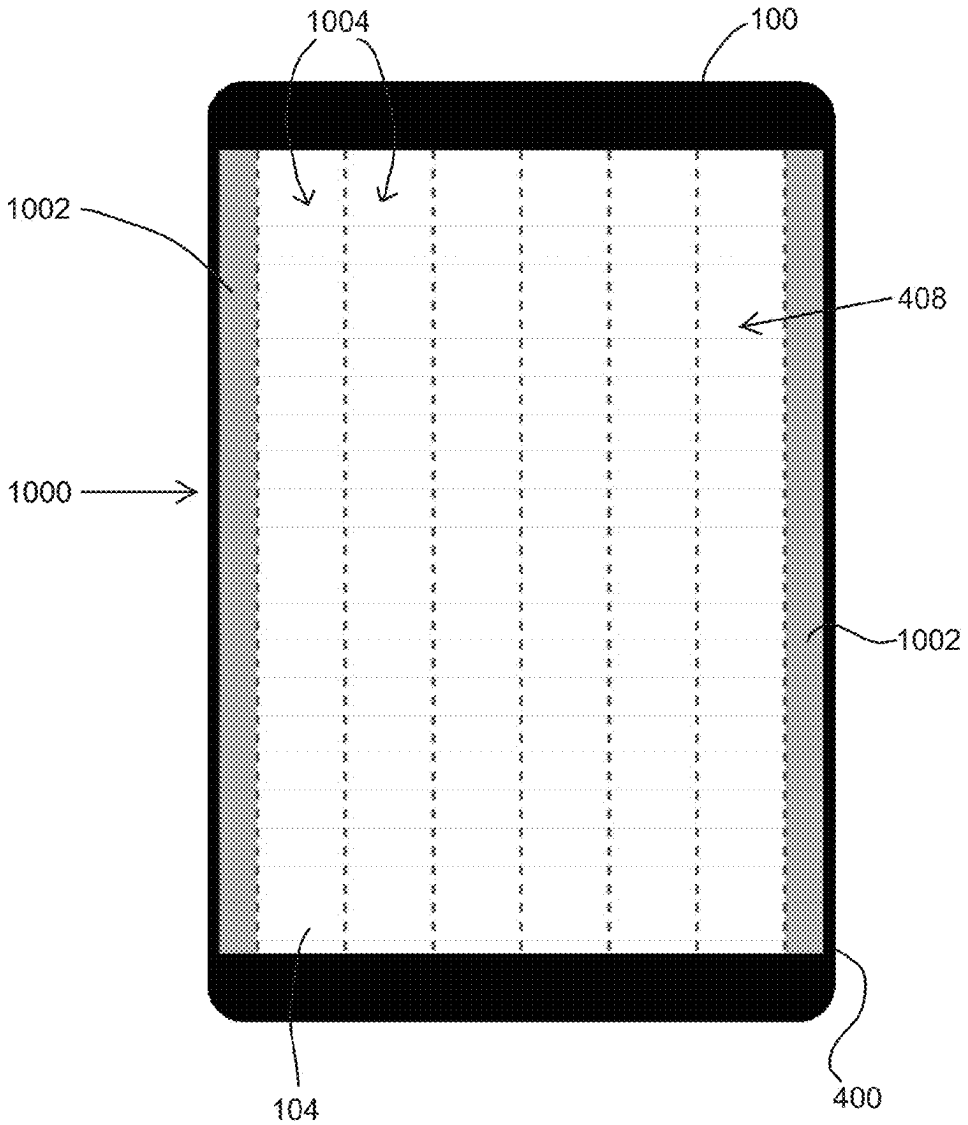


FIG. 10

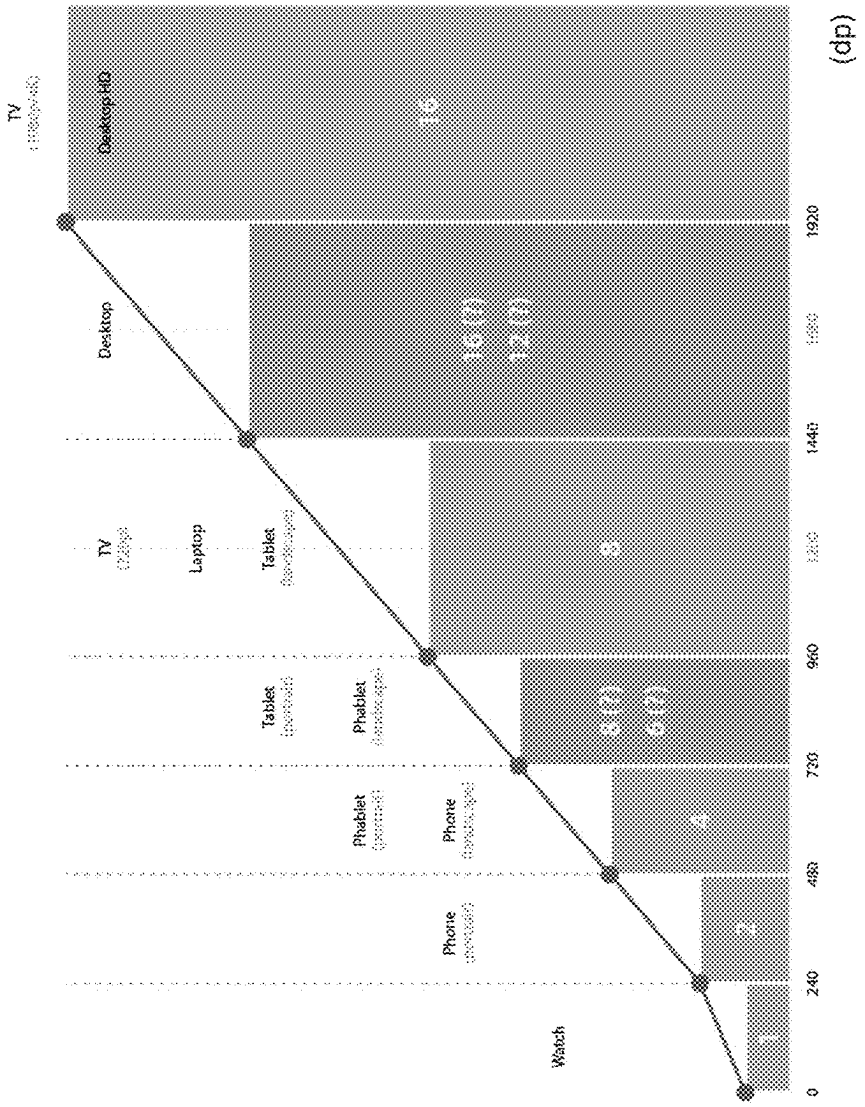

TV
Desktop HD
Desktop
TV
Laptop
Tablet
Tablet
Phablet
Phablet
Phone
Phone
Watch
16
16 (1)
12 (1)
9
6 (1)
5 (1)
4
3
1
0
240
480
720
960
1440
1920
(dp)

FIG. 11

Total	Column Count															
	1/x	2/x	3/x	4/x	5/x	6/x	7/x	8/x	9/x	10/x	11/x	12/x	13/x	14/x	15/x	16/x
x/1	1/1															
x/2	1/2							2/2								
x/4	1/4			2/4				3/4				4/4				
x/6	1/6		2/6			3/6			4/6			5/6			6/6	
x/8	1/8		2/8		3/8		4/8		5/8		6/8		7/8		8/8	
x/12	1/12	2/12	3/12		4/12	5/12	6/12		7/12	8/12	9/12		10/12	11/12	12/12	
x/16	1/16	2/16	3/16	4/16	5/16	6/16	7/16	8/16	9/16	10/16	11/16	12/16	13/16	14/16	15/16	16/16

FIG. 12A

Total	Column Count															
	1/x	2/x	3/x	4/x	5/x	6/x	7/x	8/x	9/x	10/x	11/x	12/x	13/x	14/x	15/x	16/x
x/1	1/1															
x/2	1/2							2/2								
x/4	1/4			2/4				3/4				4/4				
x/8	1/8		2/8		3/8		4/8		5/8		6/8		7/8		8/8	
x/16	1/16	2/16	3/16	4/16	5/16	6/16	7/16	8/16	9/16	10/16	11/16	12/16	13/16	14/16	15/16	16/16

FIG. 12B

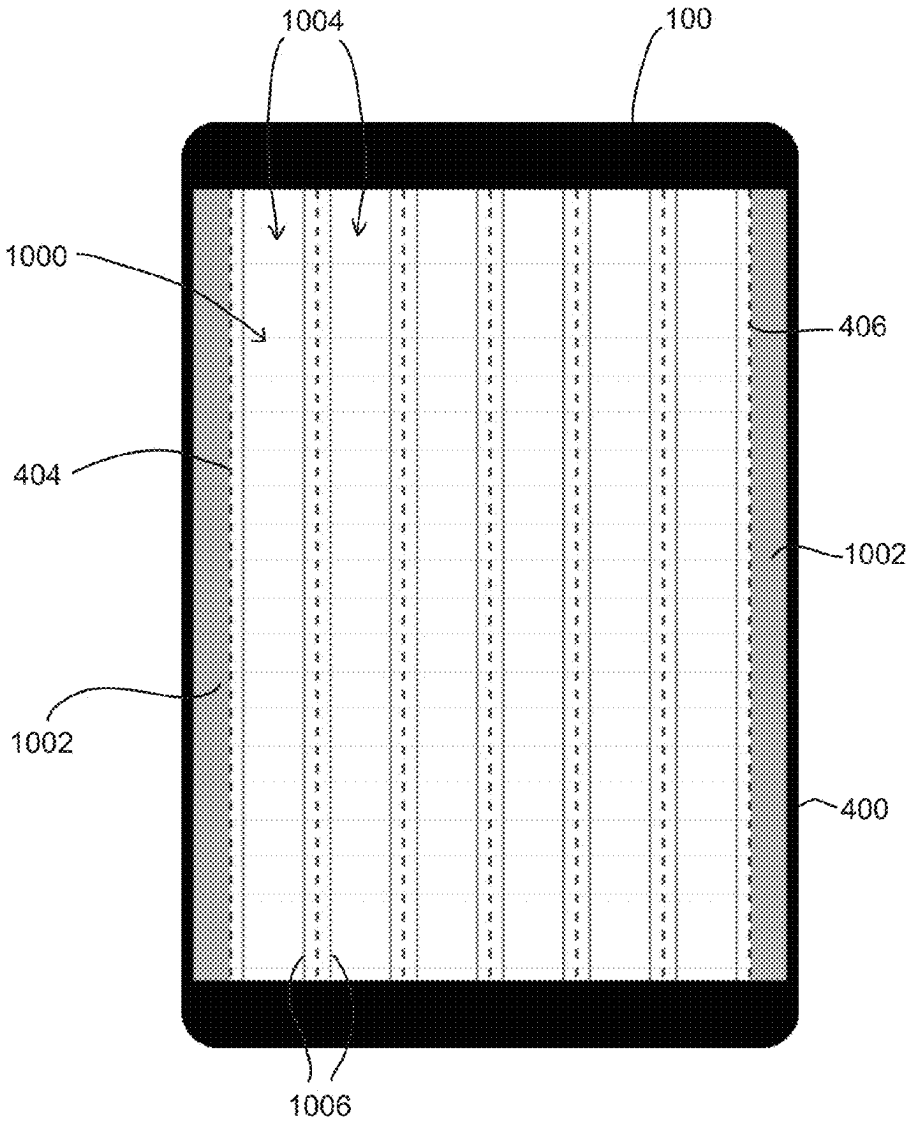


FIG. 13

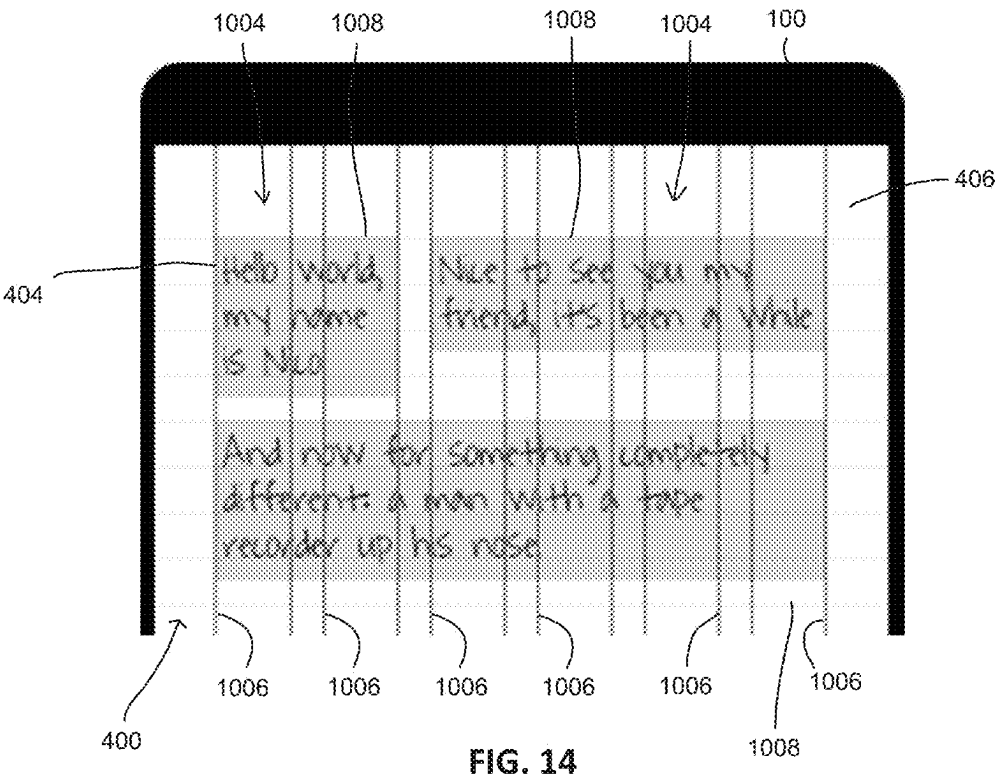

1004
1008
1008
1004
100
406
404
Hello world, my name is Nic
Nice to see you my friend, it's been a while
And now for something completely different a man with a nice
1006
1006
1006
1006
1006
1006
400
1008

FIG. 14

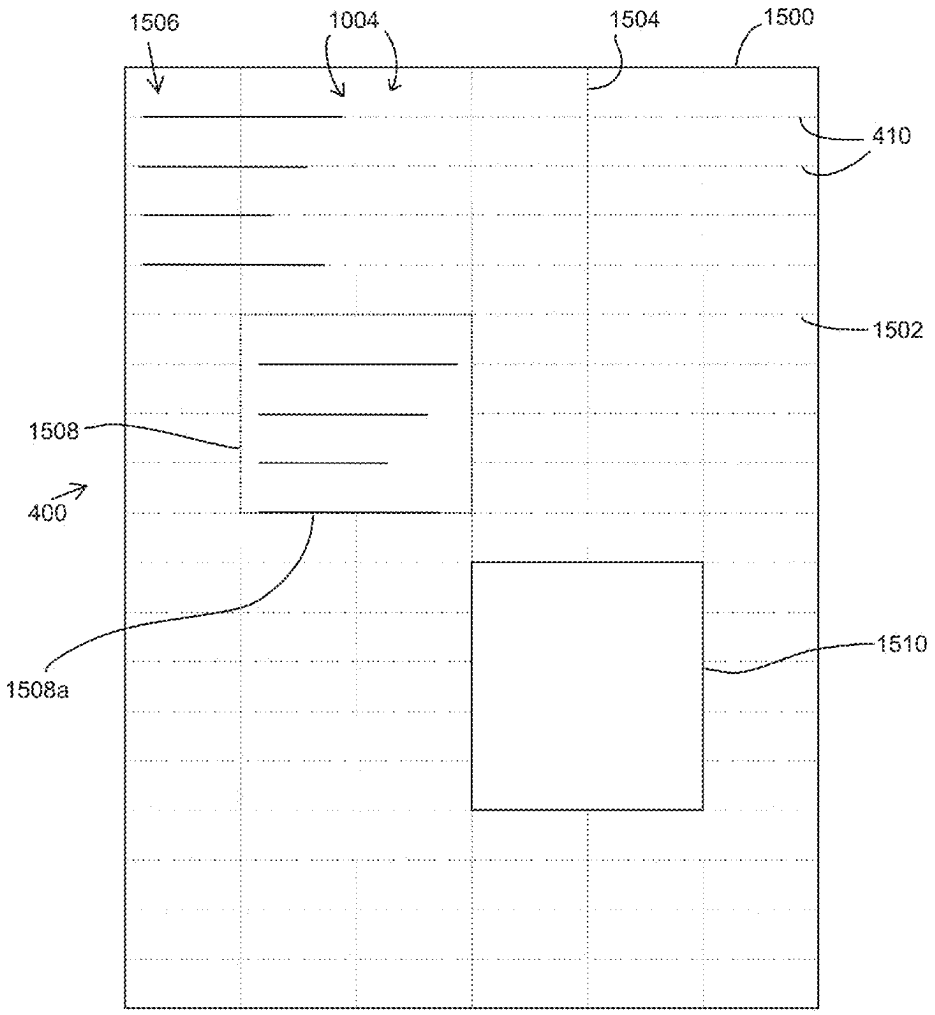


FIG. 15

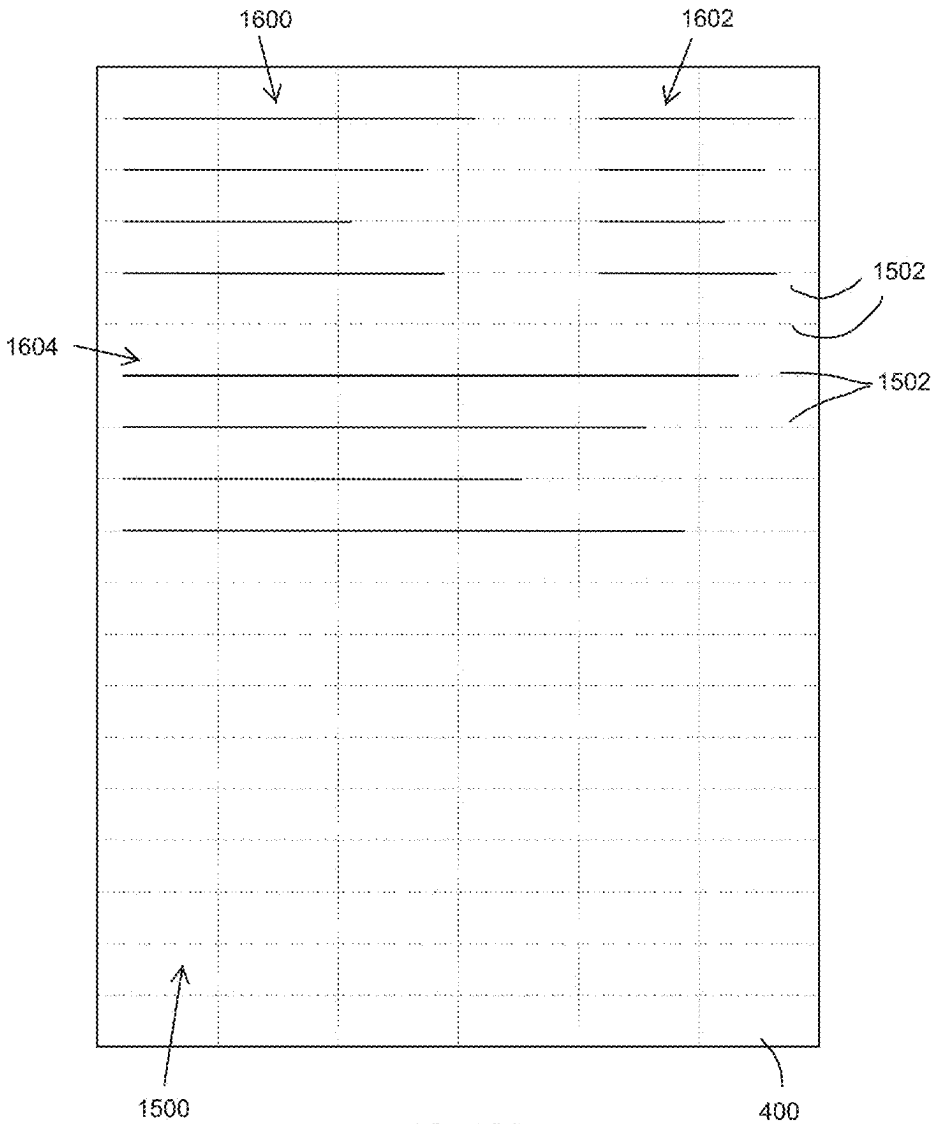


FIG. 16A

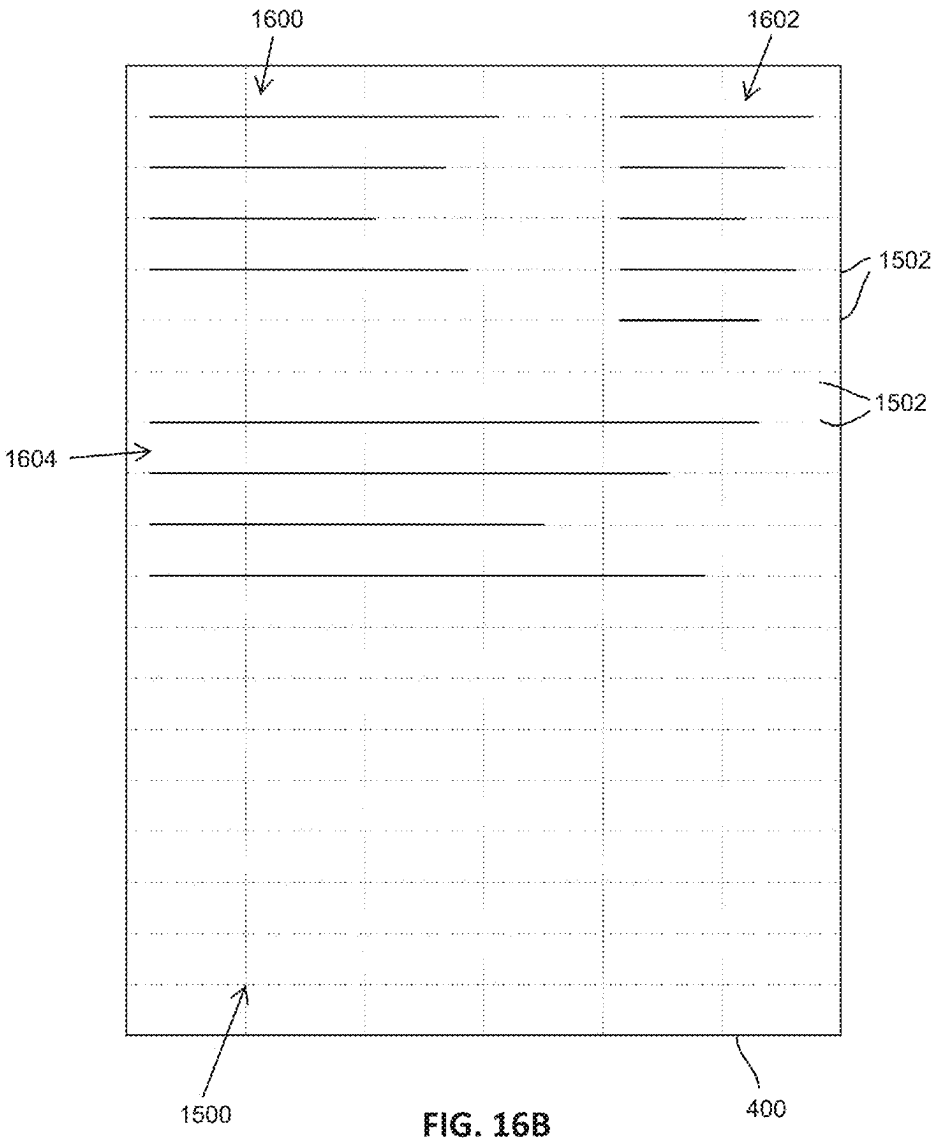


FIG. 16B

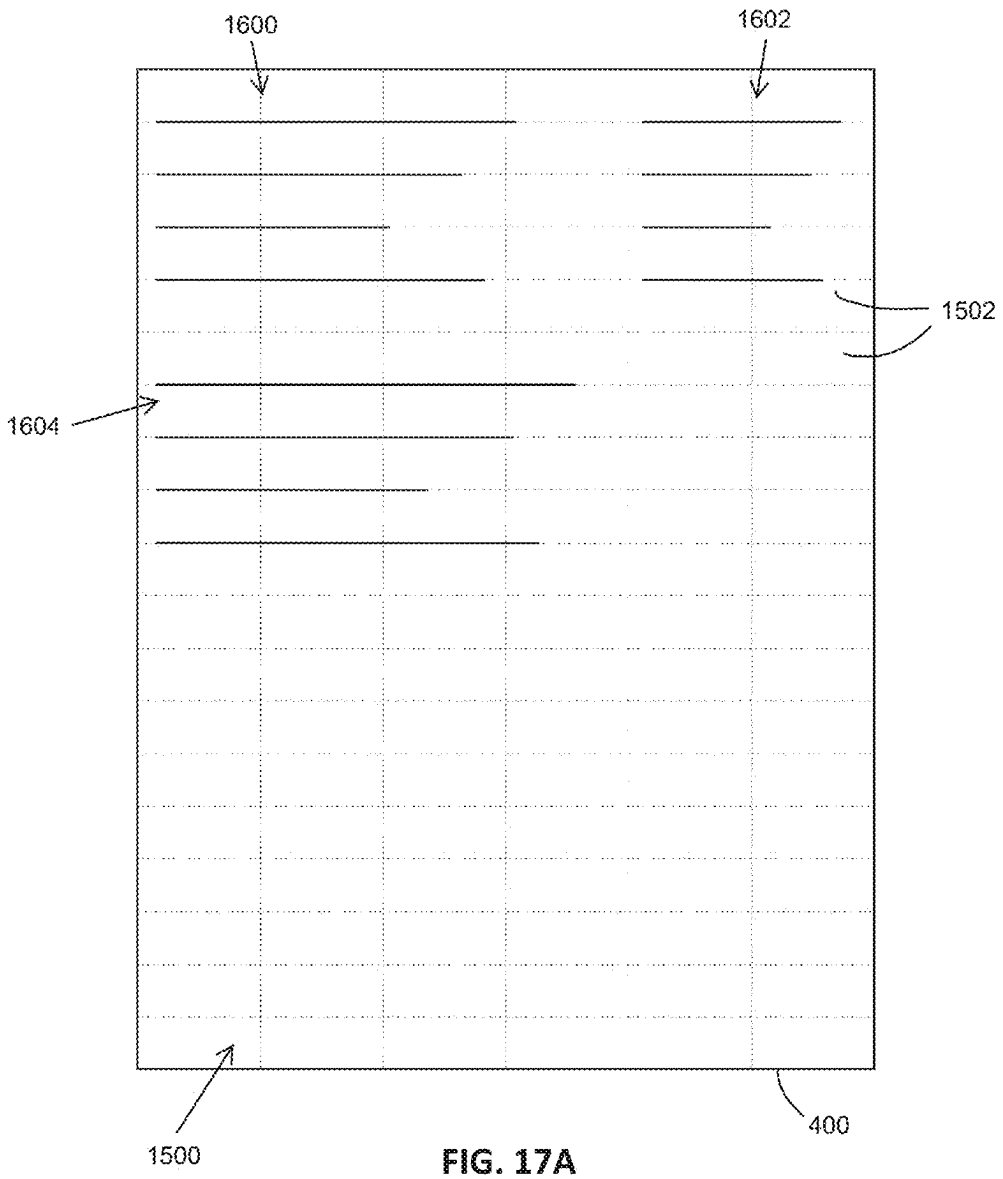


FIG. 17A

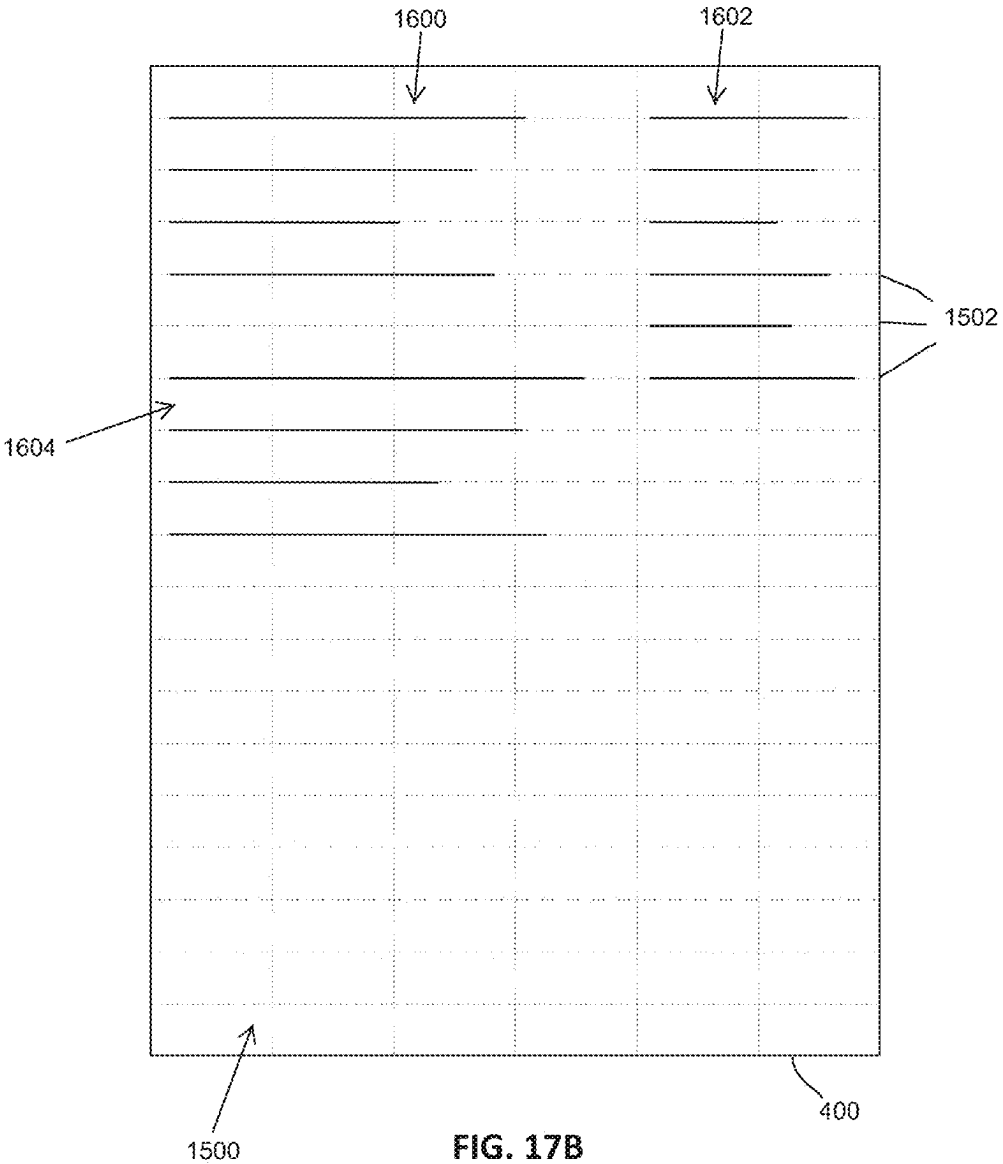


FIG. 17B

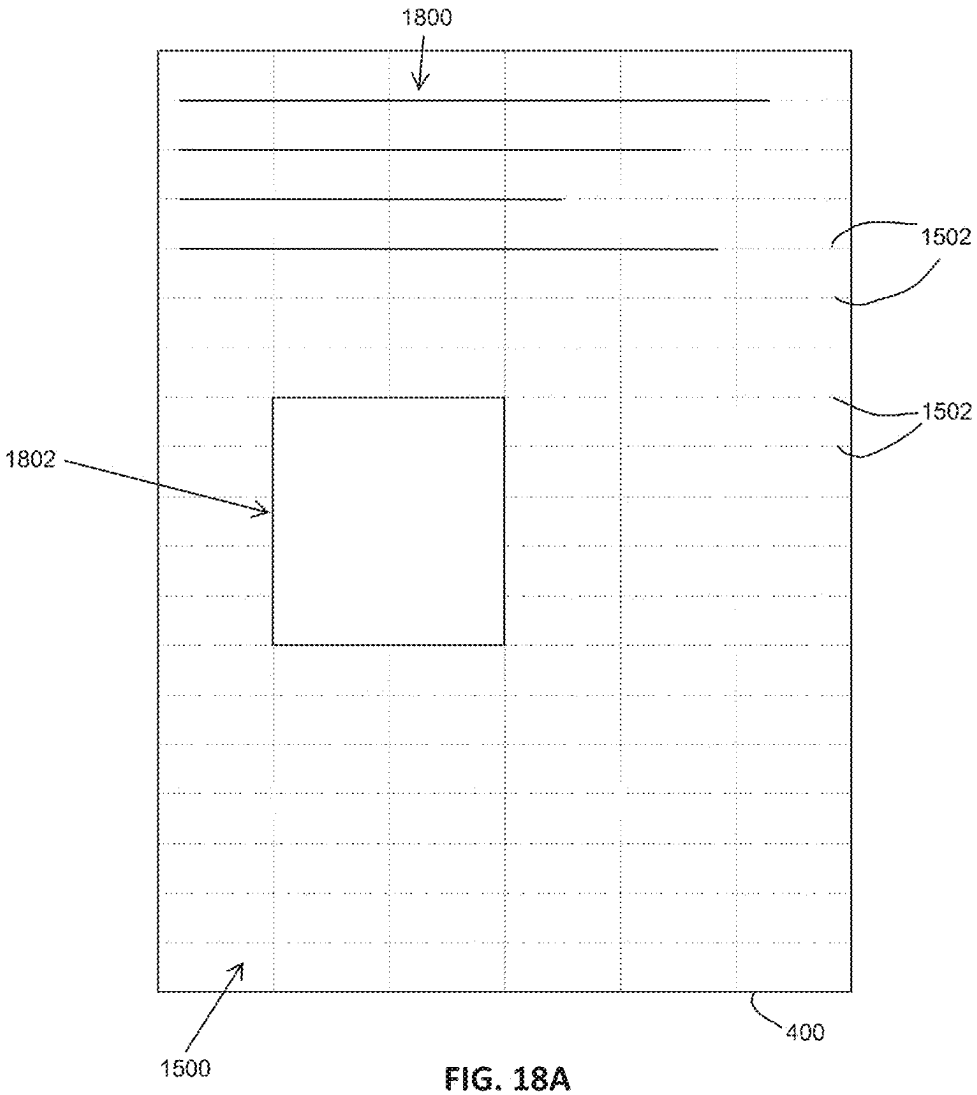


FIG. 18A

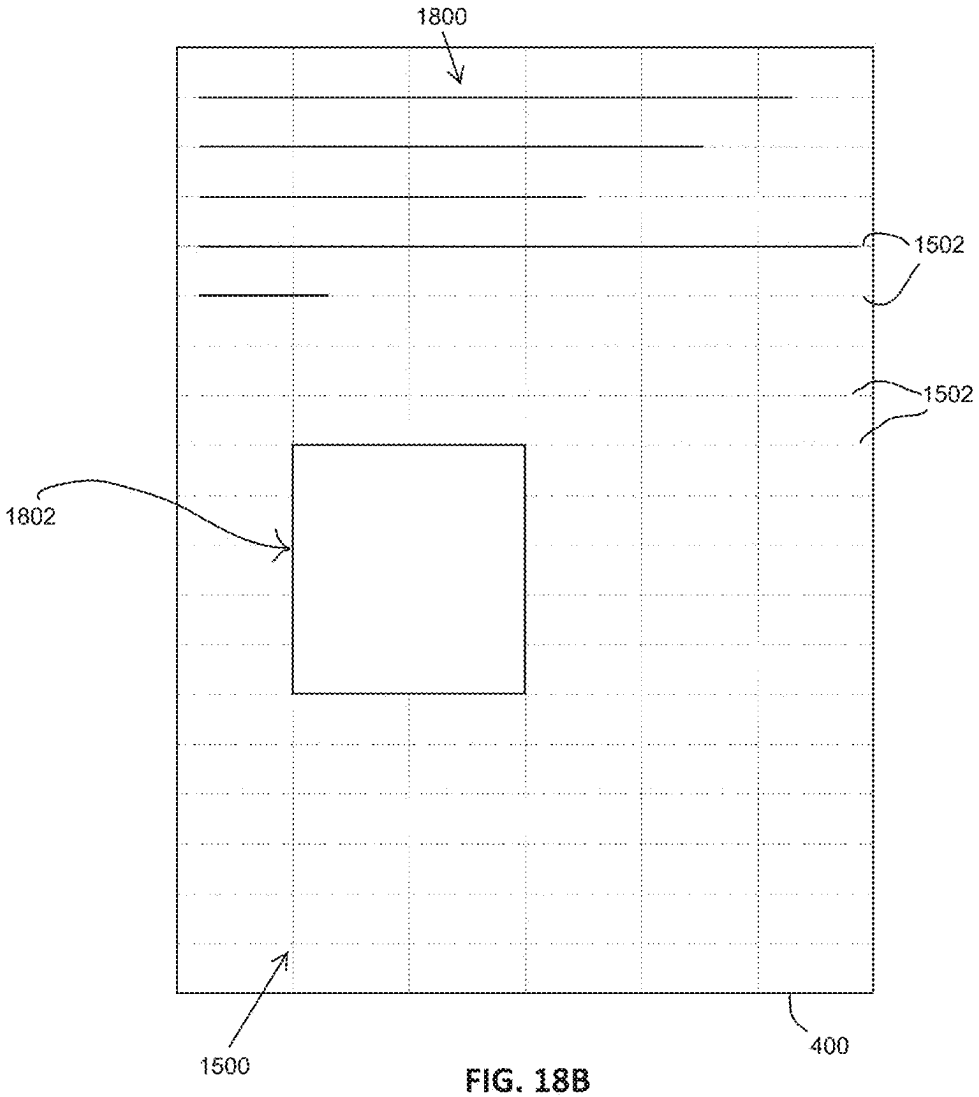


FIG. 18B

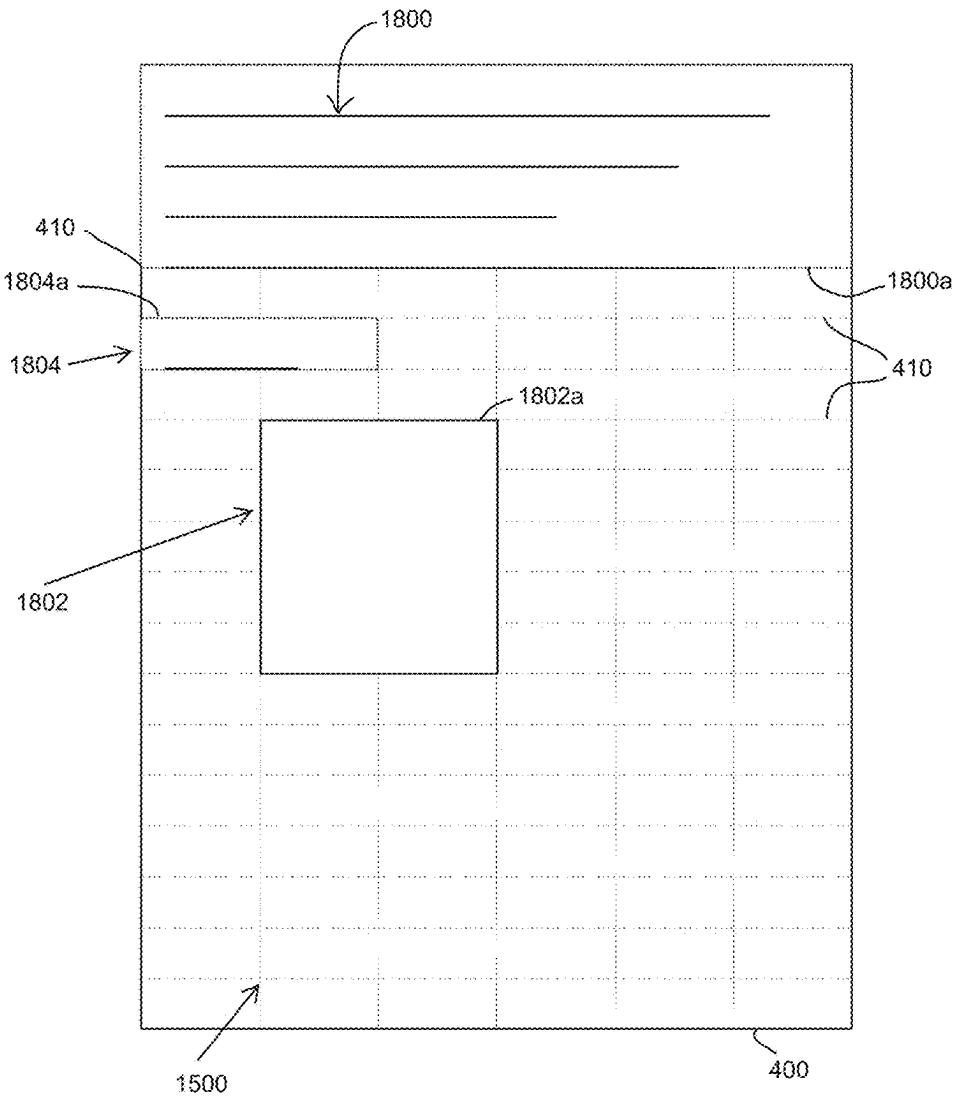


FIG. 18C

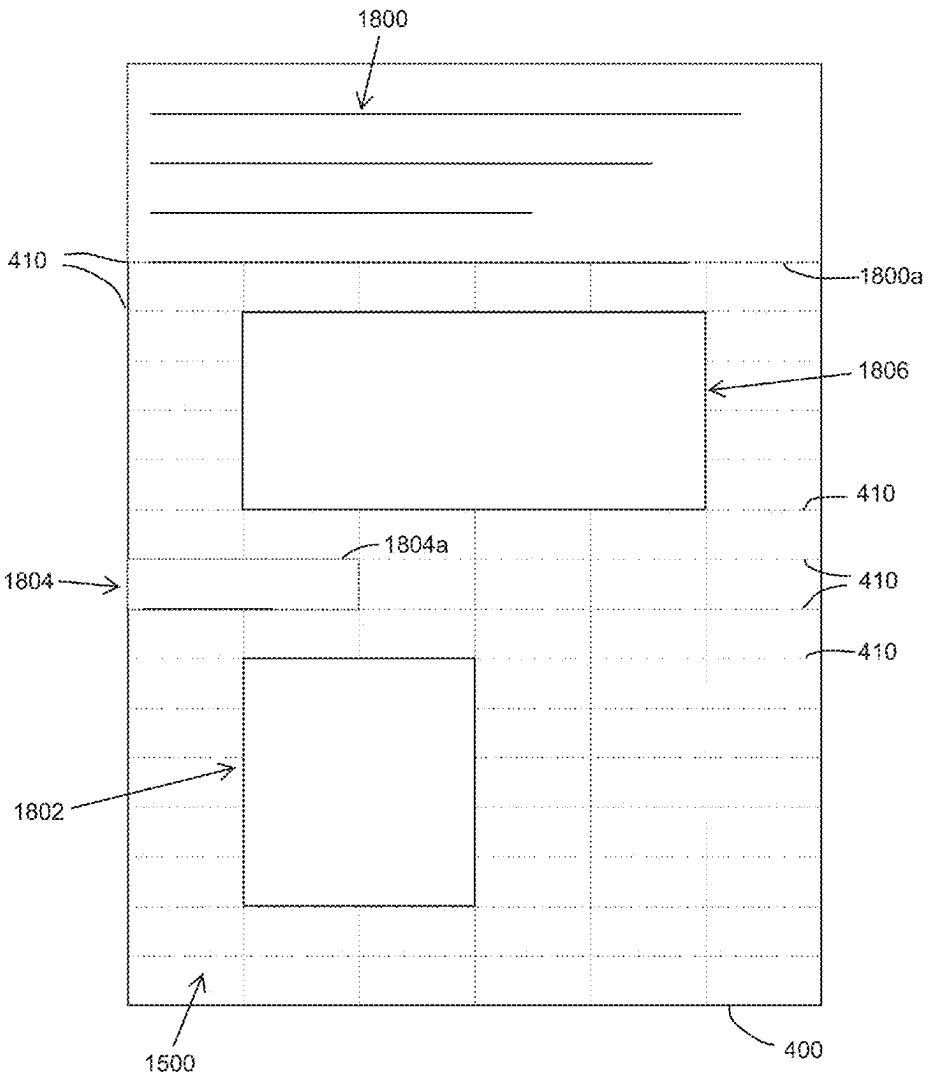


FIG. 18D

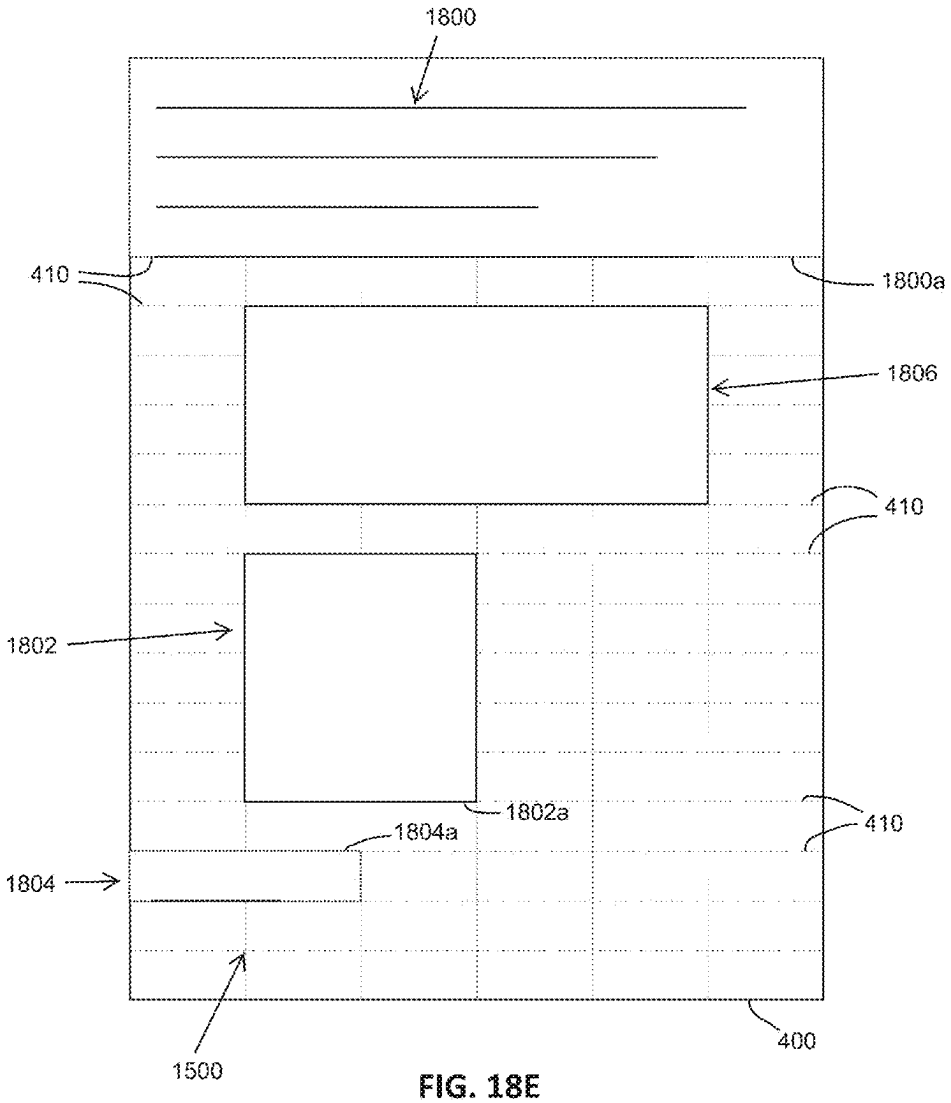


FIG. 18E

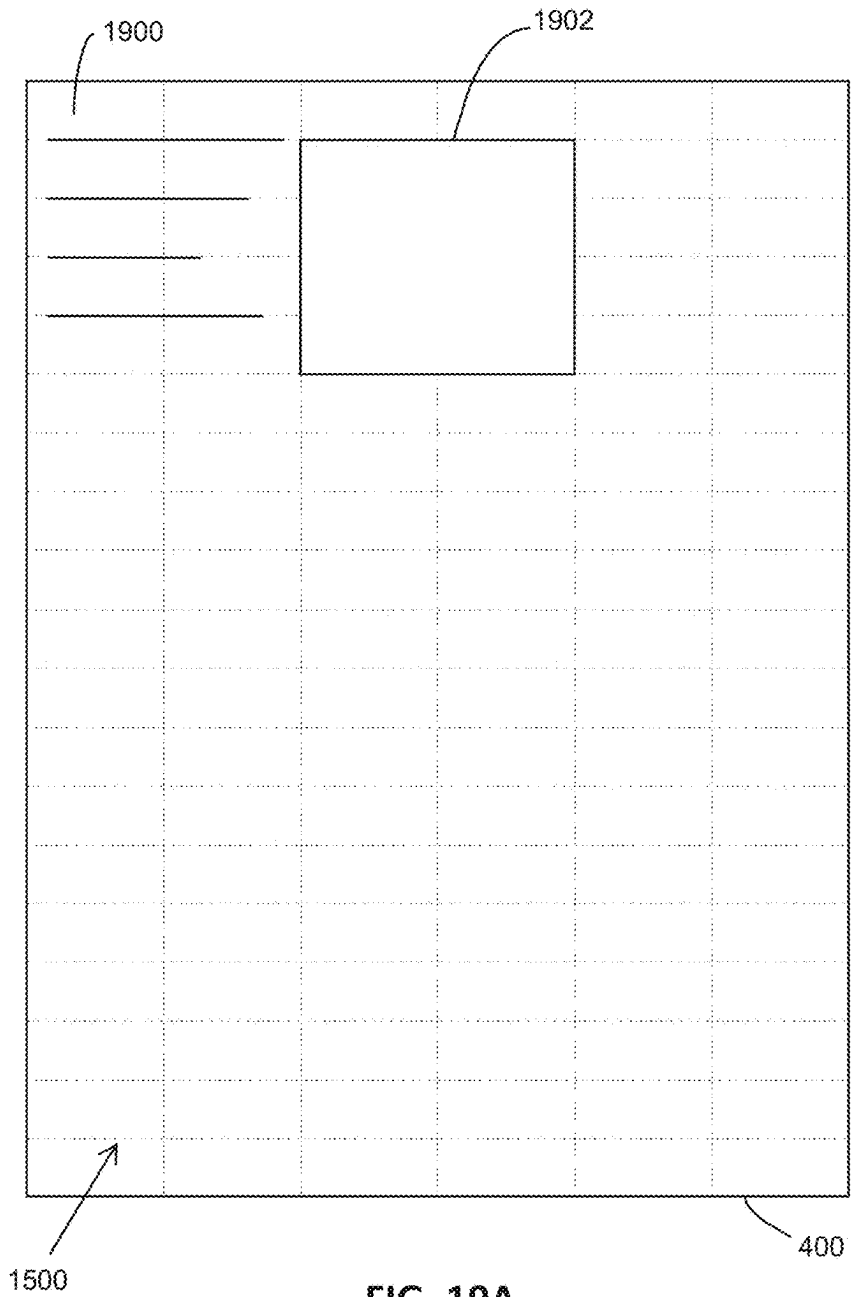


FIG. 19A

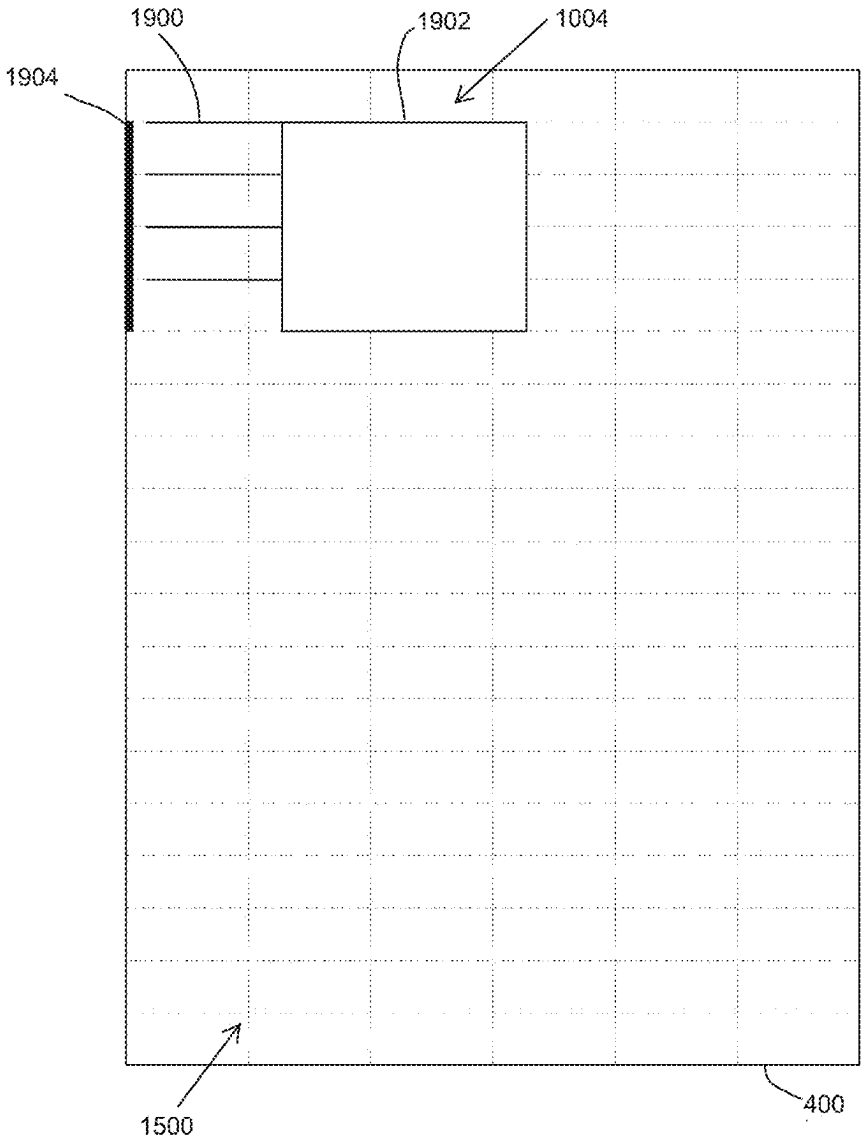


FIG. 19B

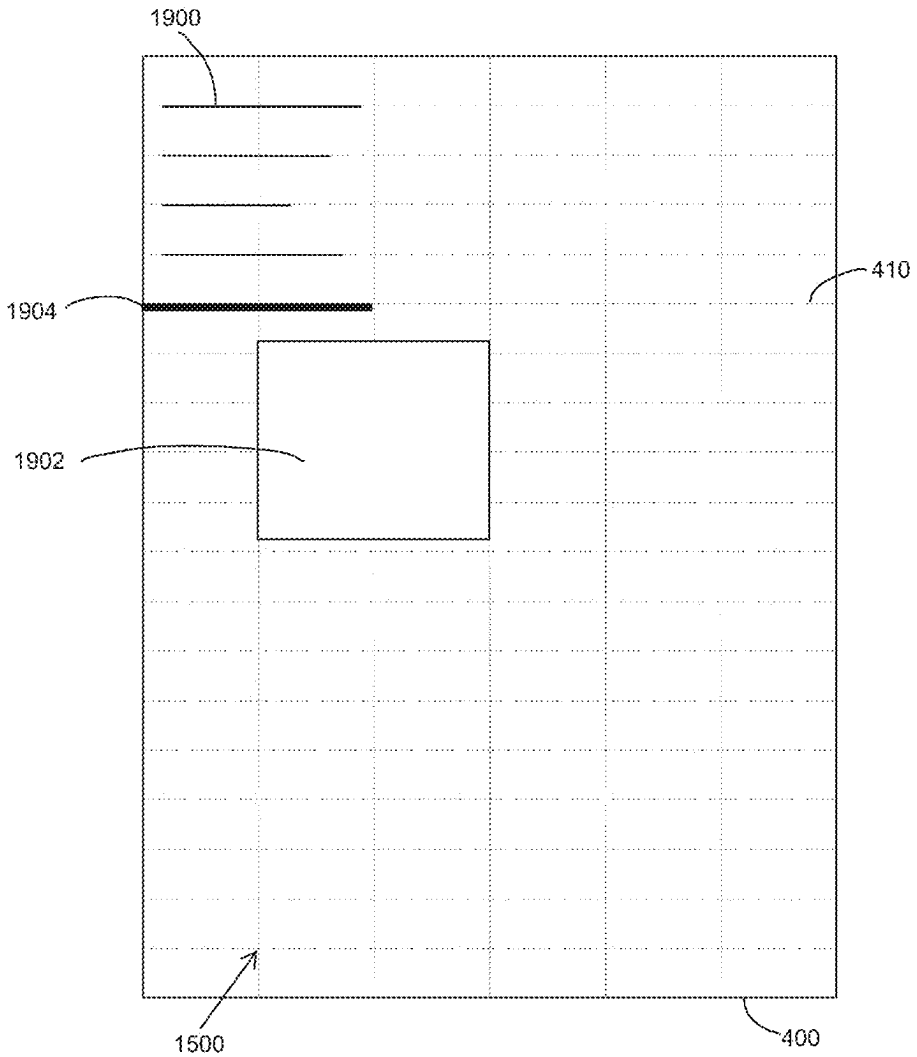


FIG. 19C

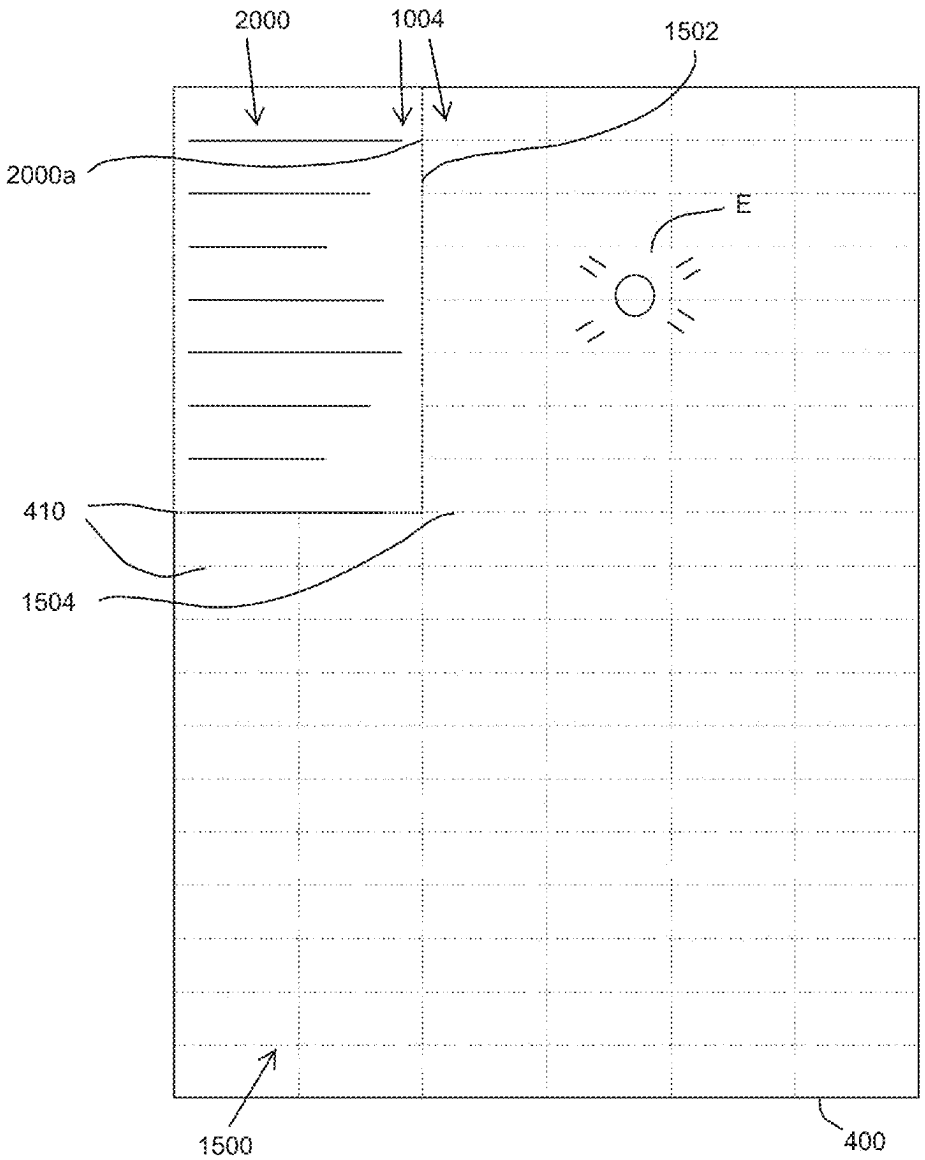


FIG. 20A

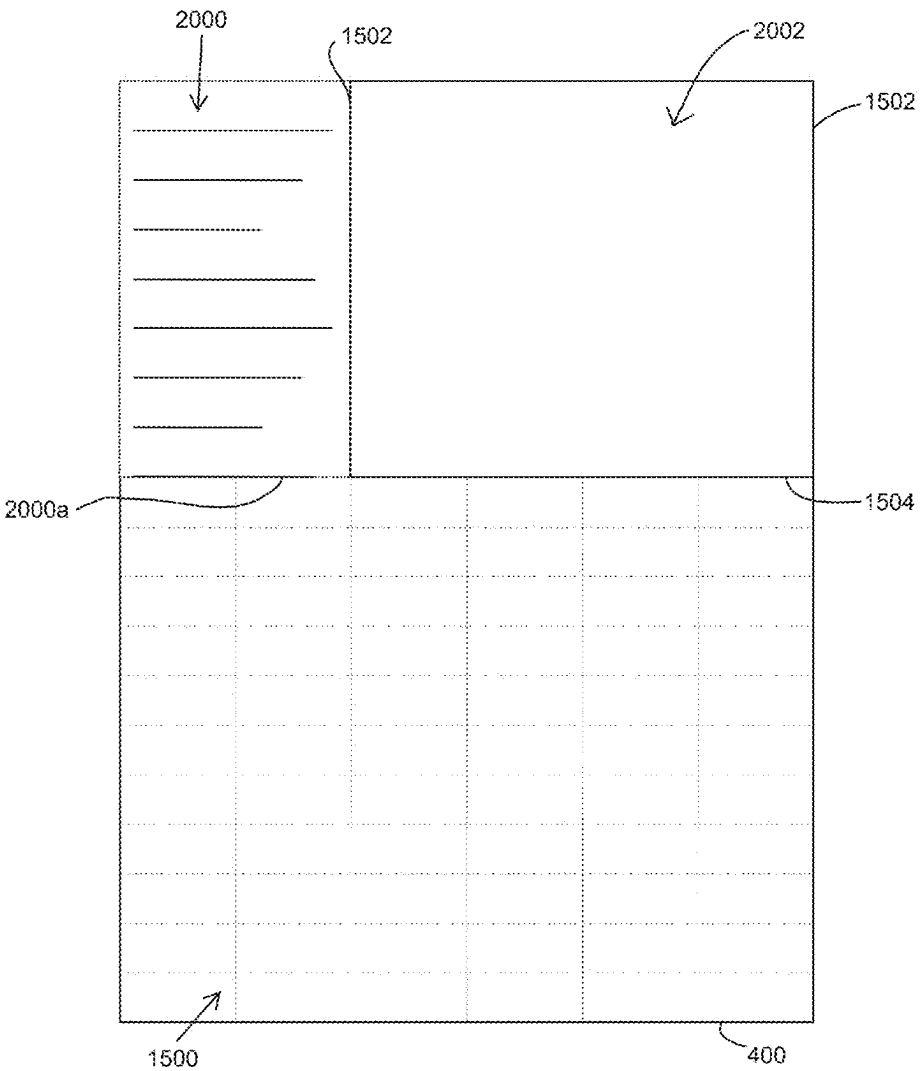


FIG. 20B

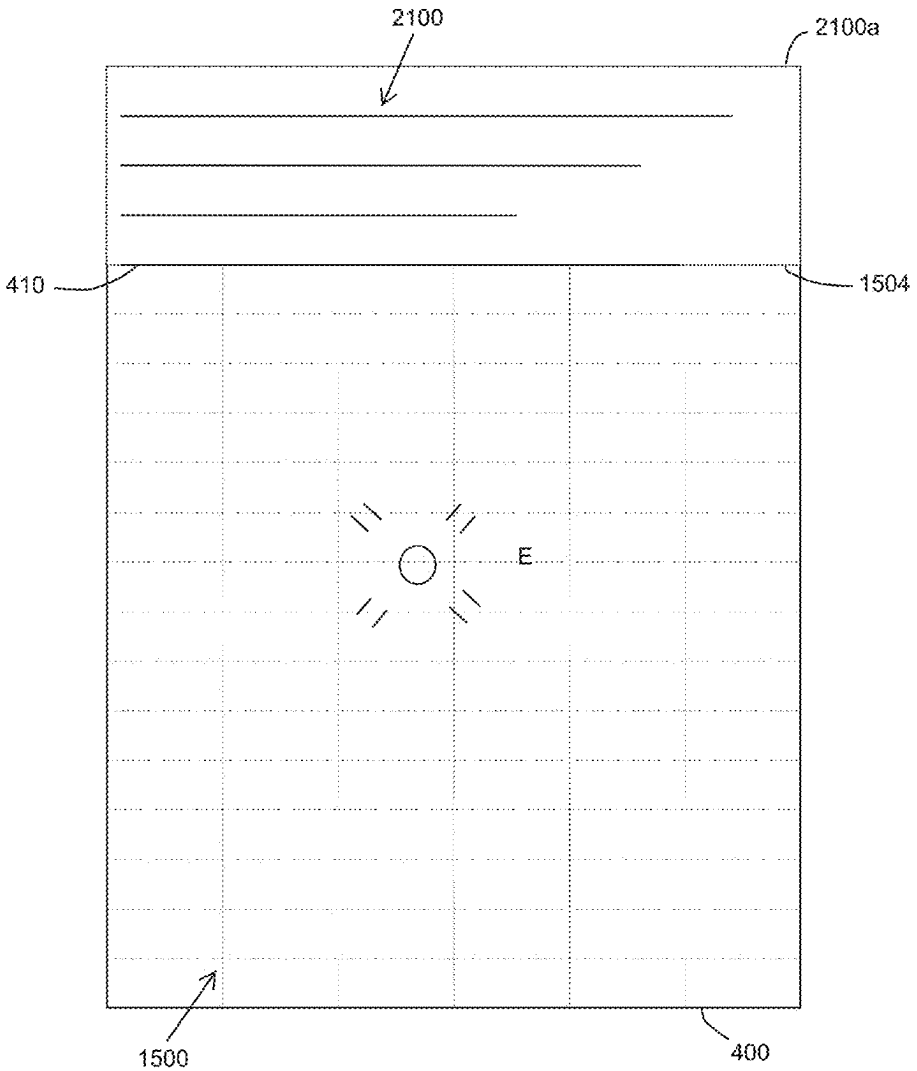


FIG. 21A

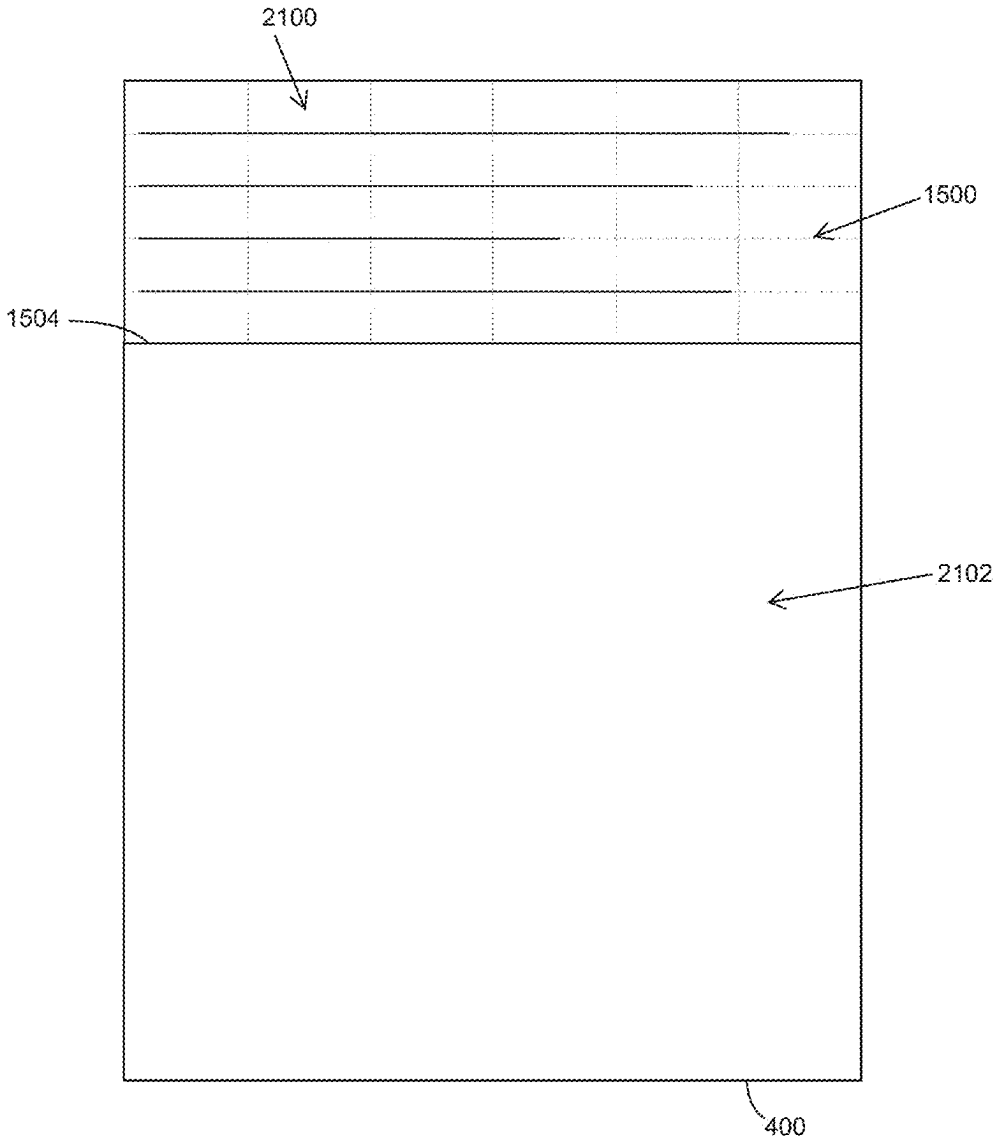


FIG. 21B

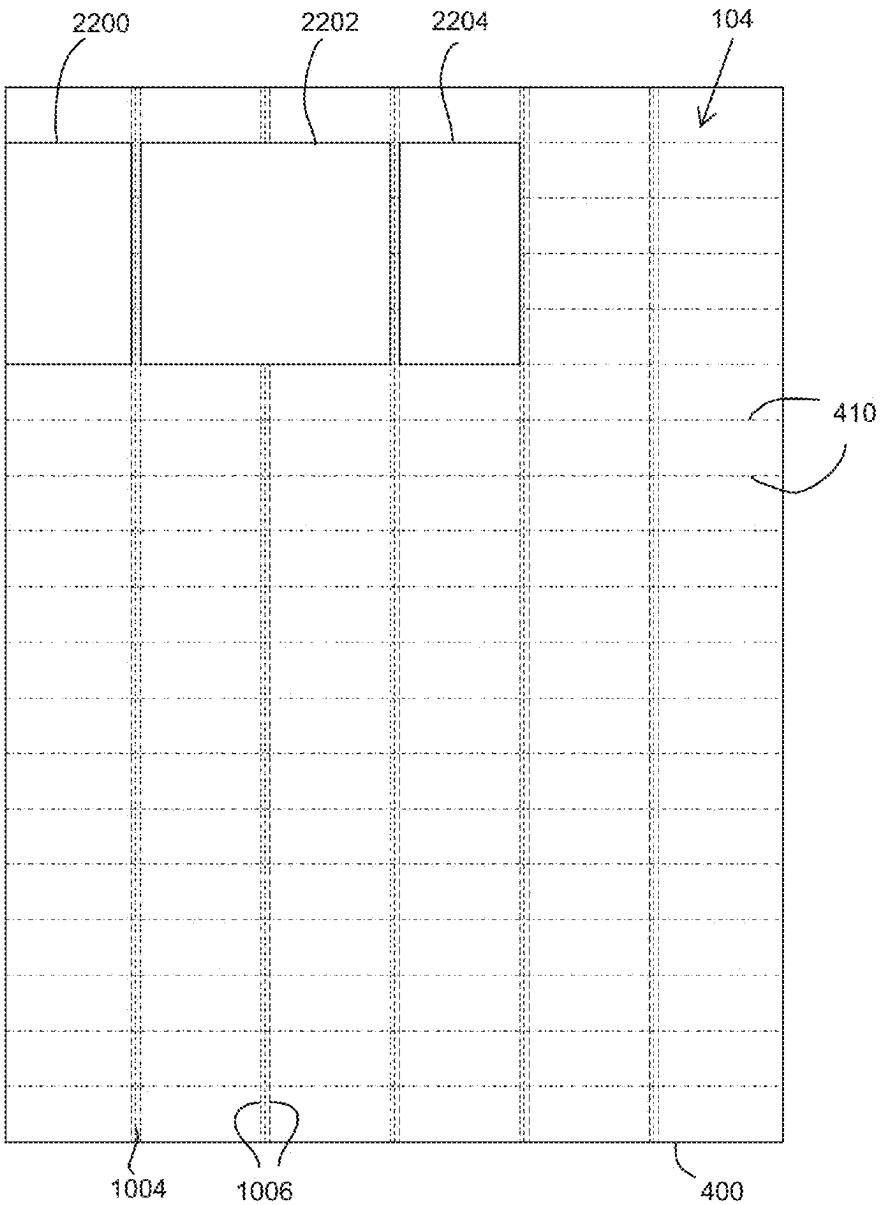


FIG. 22A

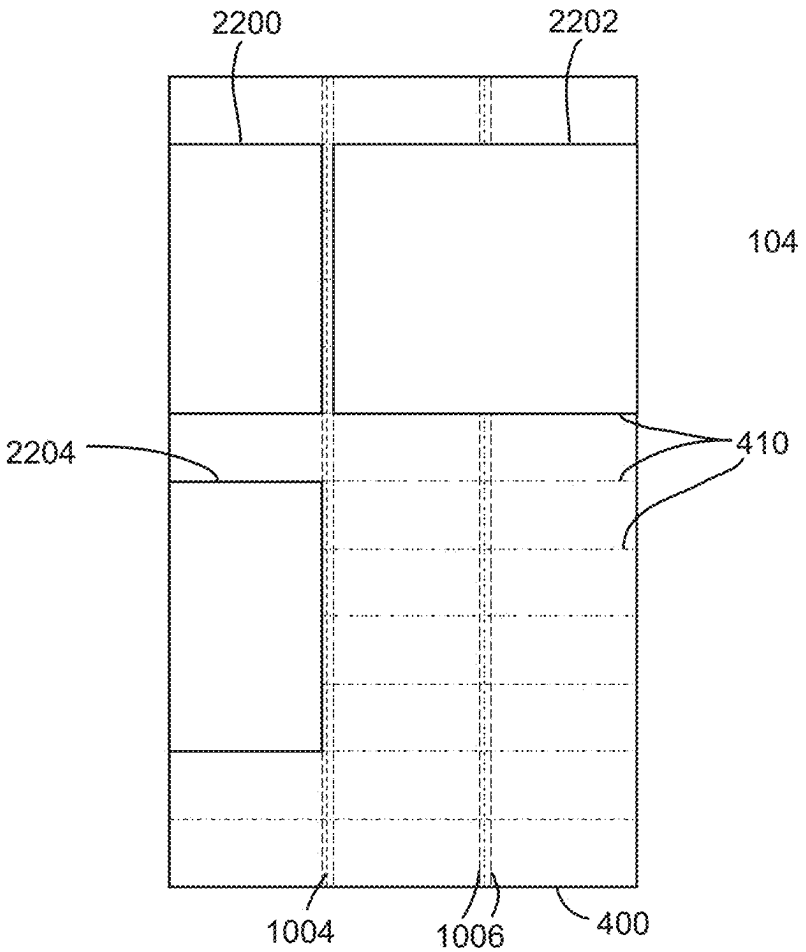


FIG. 22B

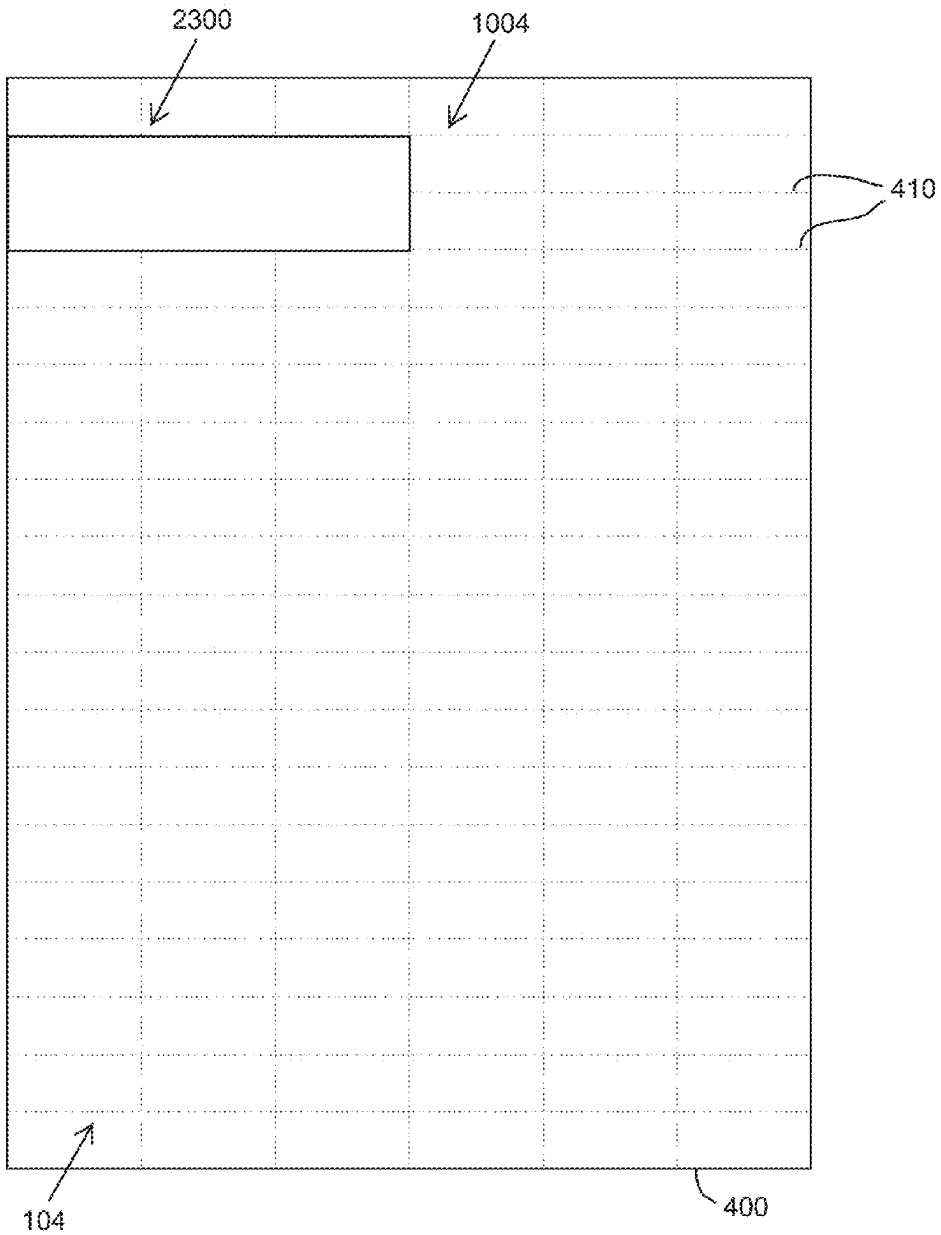


FIG. 23A

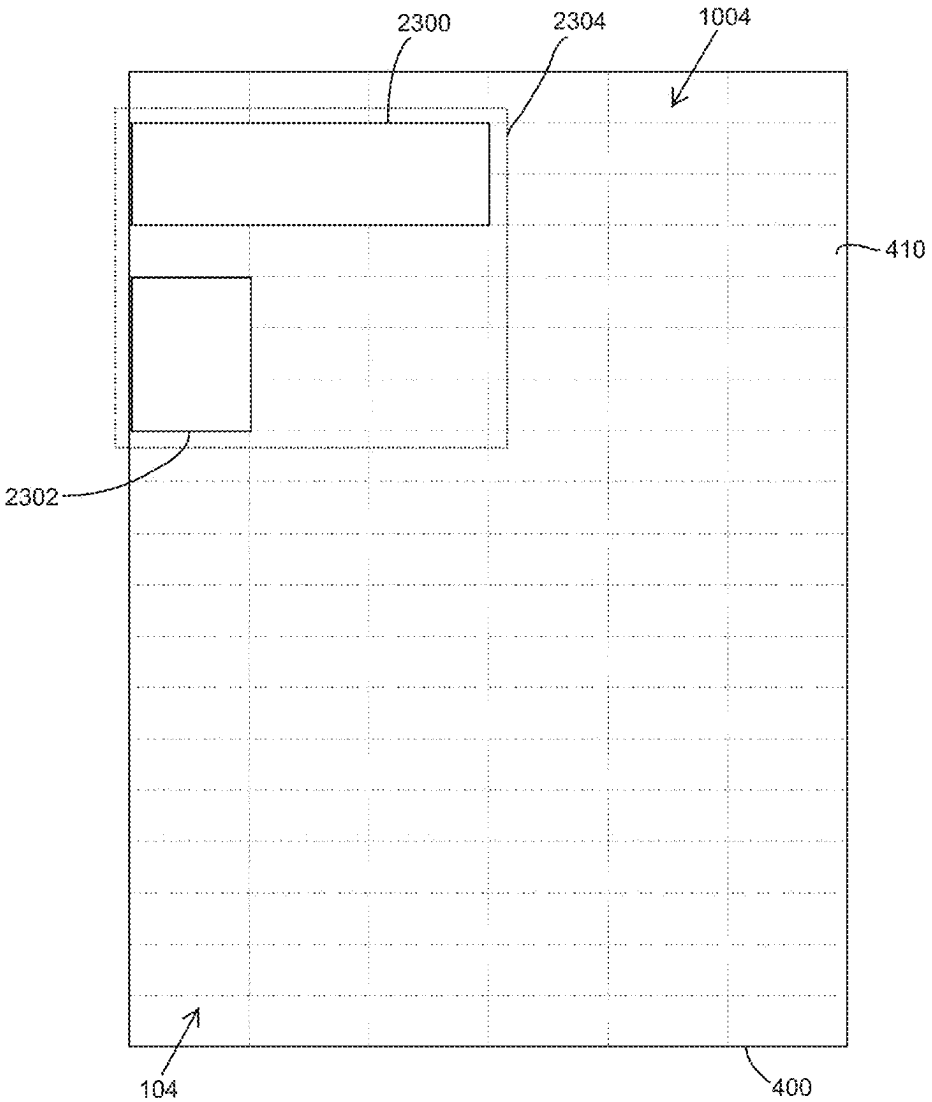


FIG. 23B

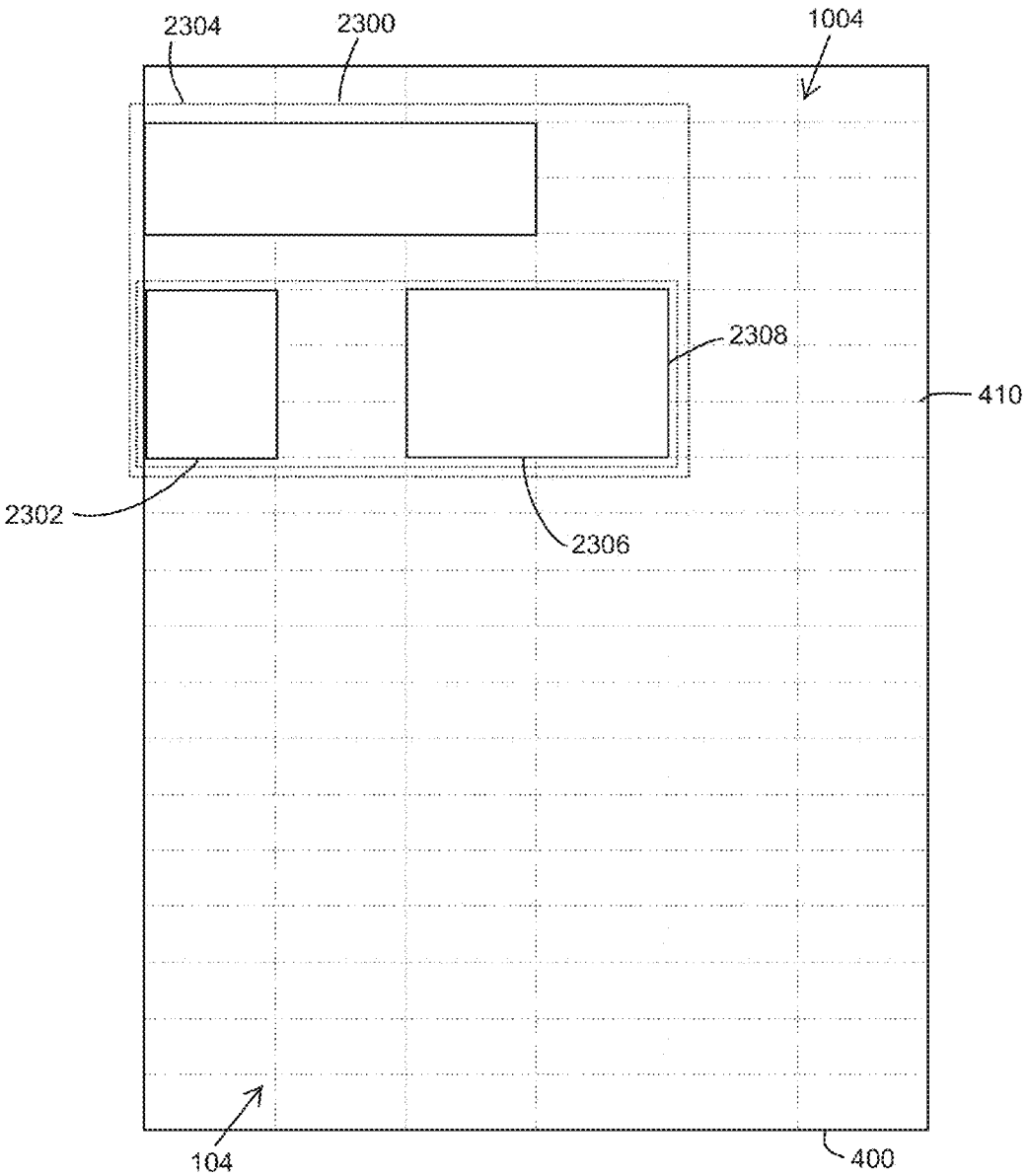


FIG. 23C

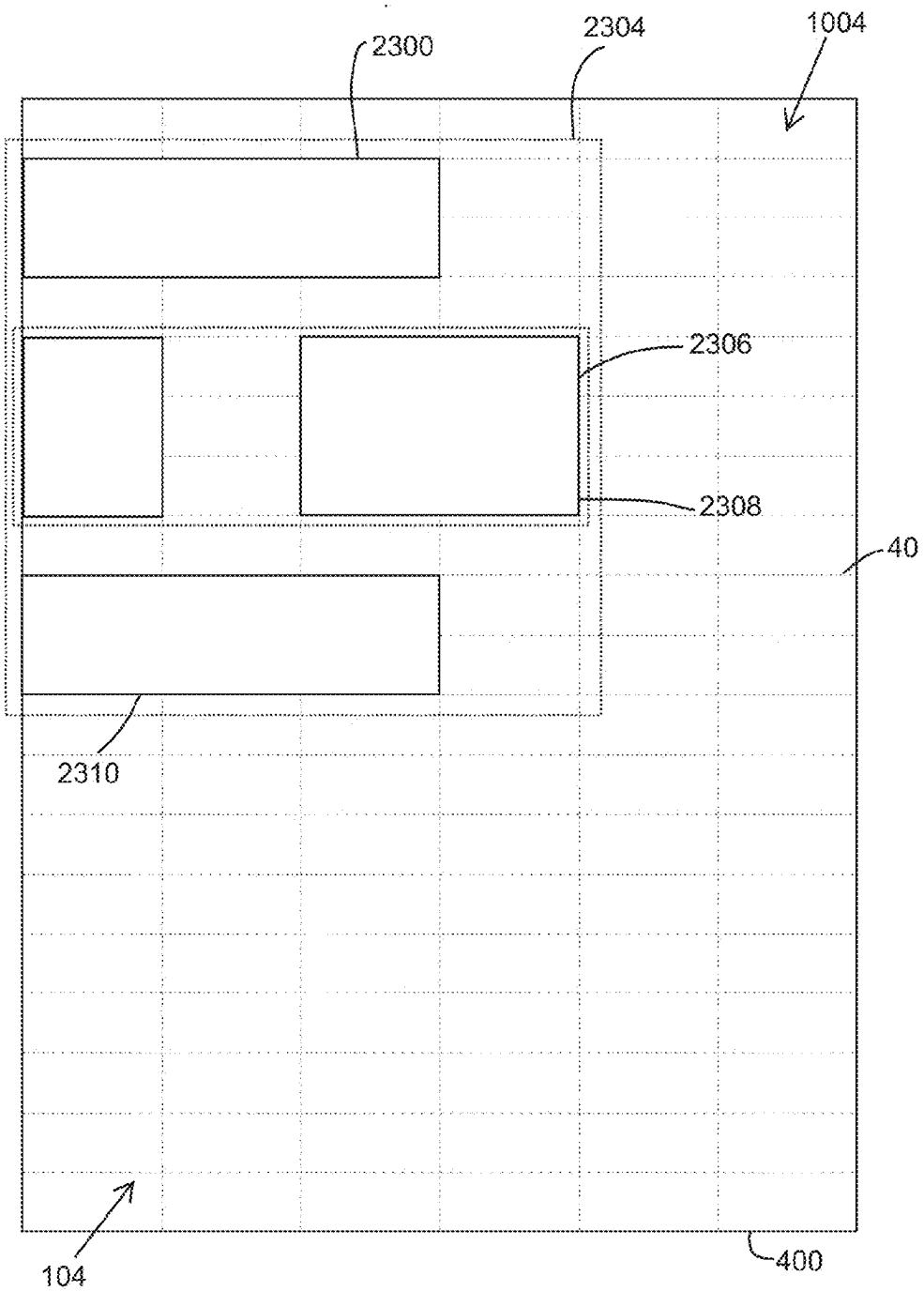


FIG. 23D

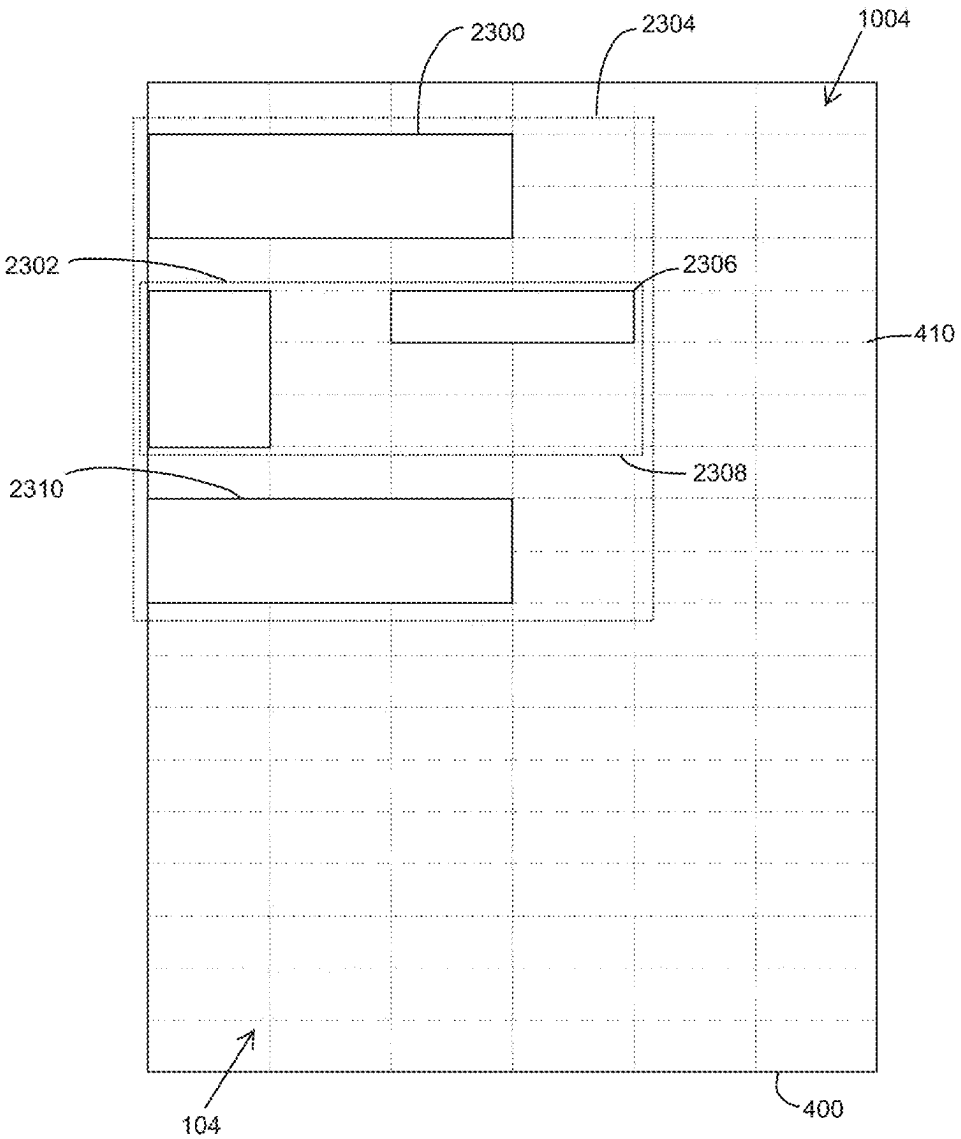


FIG. 23E

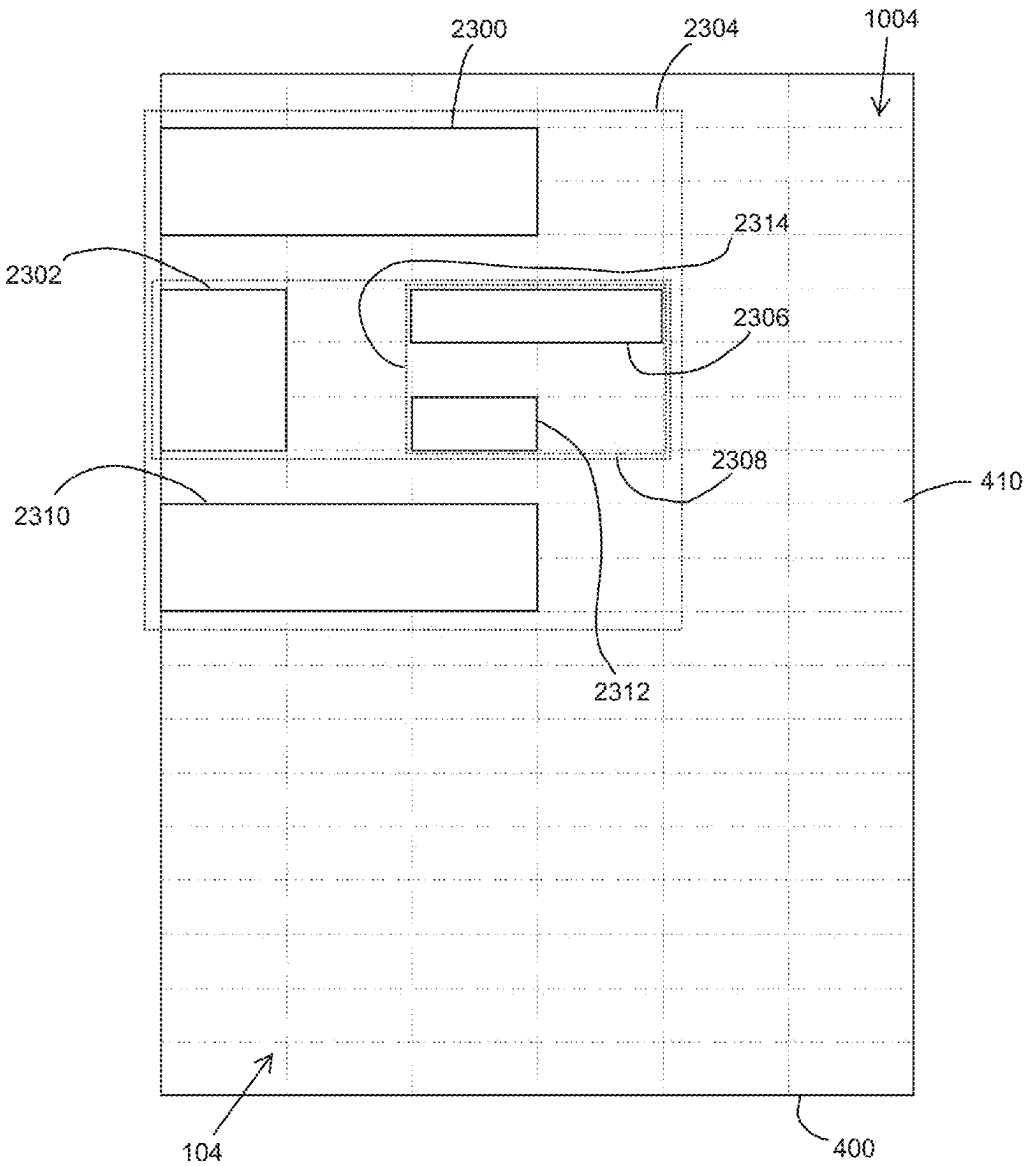


FIG. 23F

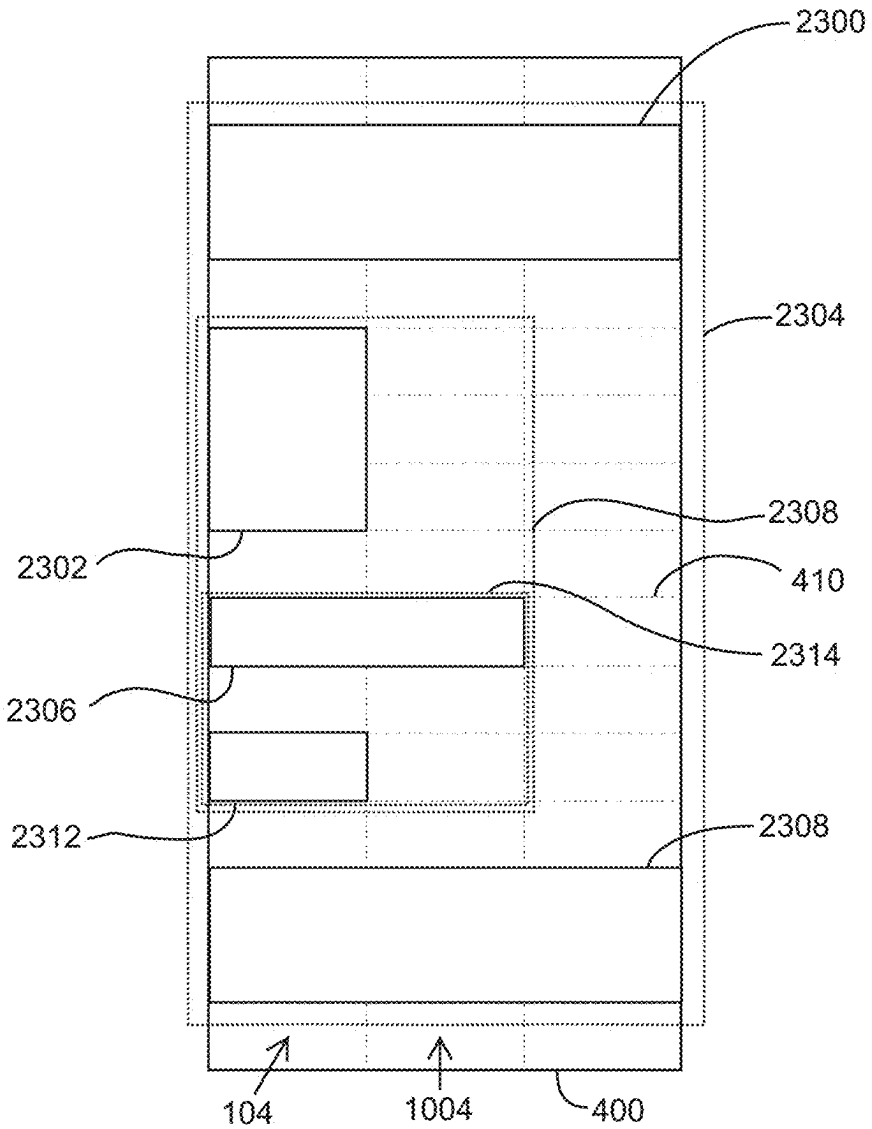


FIG. 24

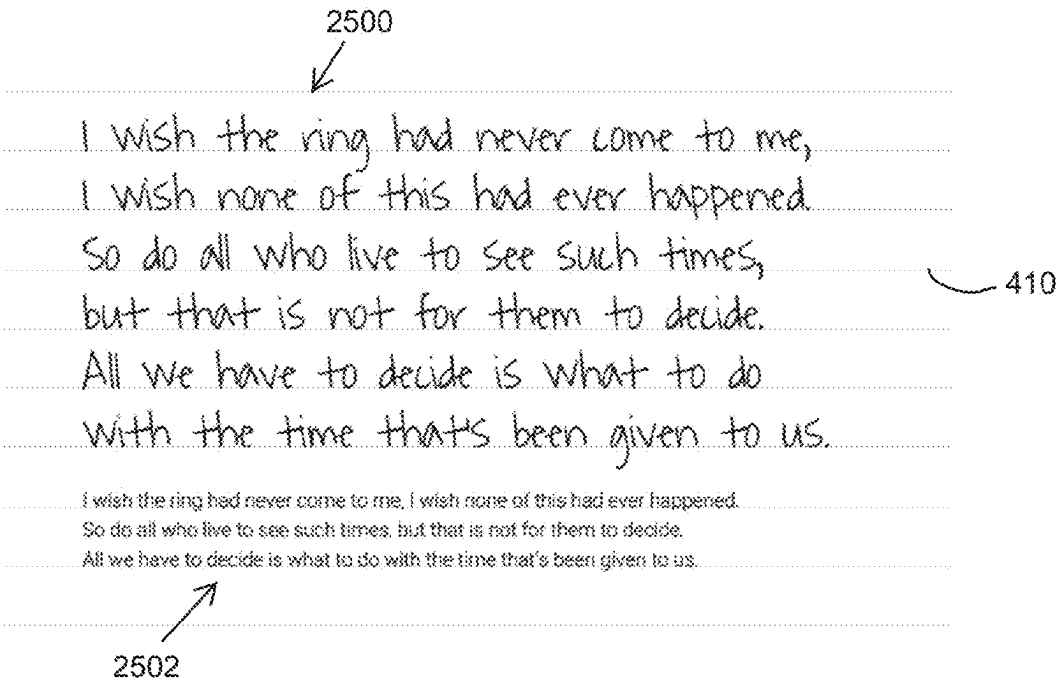

2500
I wish the ring had never come to me,
I wish none of this had ever happened.
So do all who live to see such times,
but that is not for them to decide.
All we have to decide is what to do
with the time thats been given to us.
410
I wish the ring had never come to me, I wish none of this had ever happened.
So do all who live to see such times, but that is not for them to decide.
All we have to decide is what to do with the time that's been given to us.
2502

FIG. 25

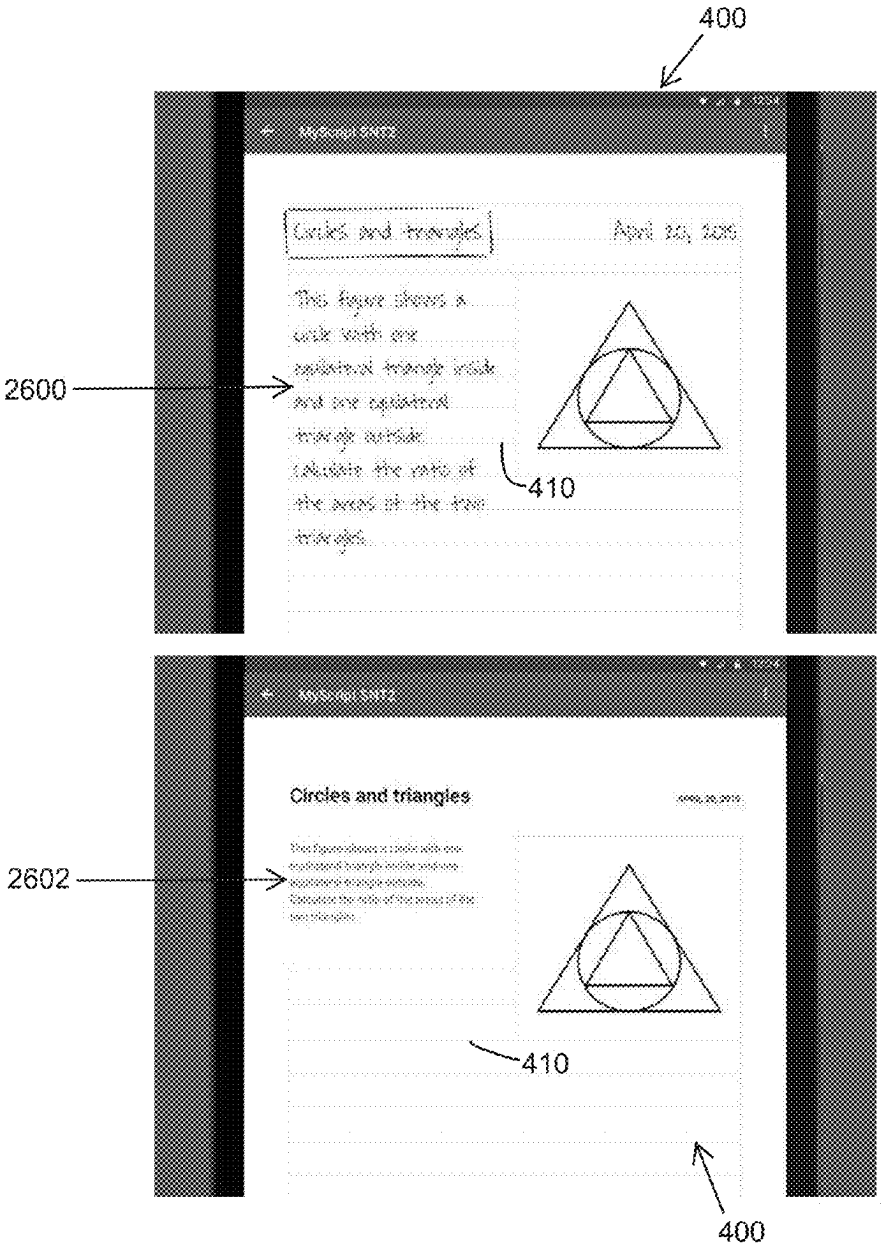


FIG. 26

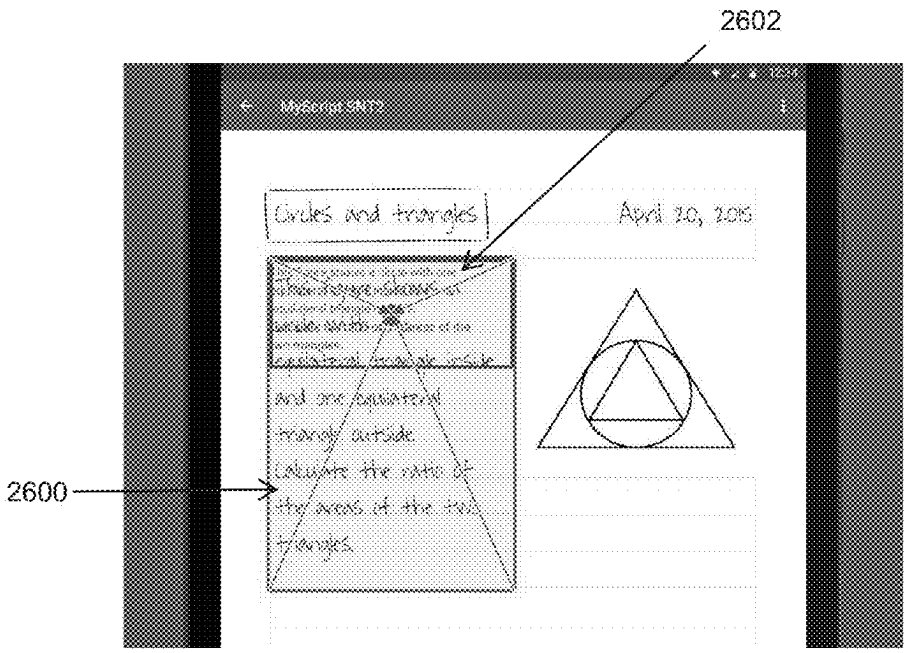


FIG. 27

SYSTEM AND METHOD OF DIGITAL NOTE TAKING

CROSS-REFERENCE TO RELATED APPLICATIONS

This application claims priority to European Application No. 15290219.3 filed on Aug. 25, 2015, the entire contents of which is incorporated by reference herein.

TECHNICAL FIELD

The present description relates generally to the field of digital note taking systems and methods using computing device interfaces. The present description relates more specifically to digital note taking systems and methods capable of recognizing user input handwriting of various characters, symbols and objects.

BACKGROUND

Computing devices continue to become more ubiquitous to daily life. They take the form of computer desktops, laptop computers, tablet computers, e-book readers, mobile phones, smartphones, wearable computers, global positioning system (GPS) units, enterprise digital assistants (EDAs), personal digital assistants (PDAs), game consoles, and the like. Further, computing devices are being incorporated into vehicles and equipment, such as cars, trucks, farm equipment, manufacturing equipment, building environment control (e.g., lighting, HVAC), and home and commercial appliances.

Computing devices generally consist of at least one processing element, such as a central processing unit (CPU), some form of memory, and input and output devices. The variety of computing devices and their subsequent uses necessitate a variety of interfaces and input devices. One such input device is a touch sensitive surface such as a touch screen or touch pad wherein user input is received through contact between the user's finger or an instrument such as a pen or stylus and the touch sensitive surface. Another input device is an input surface that senses gestures made by a user above the input surface. Either of these methods of input can be used generally for drawing or inputting text. When user input is text the user's handwriting is interpreted using a handwriting recognition system or method.

One application of handwriting recognition in portable computing devices, such as smartphones, phablets and tablets, is in note taking. This particularly occurs in education and business settings where the user of the computing device captures notes, for example, during a lecture or meeting. This is usually done by the user launching a handwritten note taking application on the computing device which accepts and interprets, either locally in the device or remotely via a communications link of the device, handwritten notes input on the touch sensitive surface. Conventionally such handwritten note taking applications are limited in their capabilities to provide a full document creation experience to users from the notes taken, since the focus of these applications has primarily been recognition accuracy rather than document creation. That is, available applications provide recognition of handwriting and conversion of the recognized handwriting into typeset text, with various feedback mechanisms to the user, but that is generally the extent of the interaction with the input.

If the user desires any further interaction with the output text, such as editing the content, manipulating the layout of the notes, or converting or adding the notes into a document, the text generally needs to be imported into a separate document processing application. This in itself is no great problem, however as the original layout of the handwritten notes and the actual input handwriting itself, so-called digital ink, is typically discarded in the import process, the user must refer back to the original notes in order to ascertain the intent in the notes taken. For example, the user may have emphasized certain words or passages in the notes either by annotation or decoration, or through the layout of the notes content itself.

Further, once the import is performed into the separate application the ability to add or edit content through further note taking is generally lost, and the user must in effect begin a new note which must itself be imported into the document. Such a cumbersome method clearly limits the ability of continuing or updating notes for inclusion in documents over different sessions, e.g., over multiple lectures or multiple business meetings, or through collaboration and sharing. Of course, users could delay importation to a document processing application until the note taking is finished over such multiple sessions, with or without multiple users. However, since many people nowadays have multiple portable computing devices which are interconnected it may be desired to perform these multiple sessions on these different computing devices, or at least access the notes on these different devices.

For example, a user may begin a note on their laptop during a lecture or meeting, then wish to add to that note using their tablet during a conversation with a colleague or friend, and then later share the updated note using their smartphone. The available input and display space on the screens of these devices can vary significantly. Further, some devices allow display at different orientations, i.e., portrait and landscape. As such, the user's layout of the notes using one device at one orientation may be significantly changed when the notes are displayed and interacted with on a different device or at a different orientation. Accordingly, continued note taking on specific note layout over these different display scenarios is presently difficult.

Other issues with the note taking layout occur without even considering the complexities of multiple devices and orientations. Available handwritten note taking applications generally provide users with the ability to typeset the handwritten input either manually or automatically (e.g., on-the-fly). However, as typeset text is generally smaller and more uniform than handwritten text (described in detail later), the user's layout of the handwritten input, such as in sentences in paragraphs, may not translate well to typeset text, e.g., the sentences become too short such that the paragraphs lose visual meaning and look more like lists than paragraphed text. This can be overcome by reflowing the typeset text so that the layout appears to be preserved, e.g., paragraphs are retained. However, the user's original handwritten layout may have actually been a list not a paragraph, for example, and without an explicit indication of such layouts it is difficult for the handwriting recognition process to detect these instances.

Conventional processes require users to input certain symbols or characters, e.g., bullet points, or use particular gestures to indicate such layouts. However, the user should be able to input handwritten notes in their desired layout without needing to consider the manner in which the handwritten note taking application will handle that input. This is because, the Applicant has found that when using handwriting note taking applications users generally are unable or do

not desire to learn specific input methods or gestures that are not natural or intuitive, or to make settings through menus and the like.

Further, note taking generally takes many forms, and user's don't only take notes using text characters. For example, notes may include many types of content, such as text, drawings, diagrams, mathematical or chemical equations and music notation. Further, the user may want to insert other media into the note, such as images and audio or video recordings. Accordingly, such multi-content and multi-media input should be supported whilst respecting the user's layout of these inputs.

SUMMARY

The examples of the present invention that are described herein below provide systems, methods and a computer program product for use in digital note taking with handwriting input to a computing device. The computer program product has a computer usable medium with a computer readable program code embodied therein adapted to be executed to implement the method.

The computing device is connected to an input device in the form of an input surface. A user is able to provide input by applying pressure to or gesturing above the input surface using either his or her finger or an instrument such as a stylus or pen. The present system and method monitors the input strokes.

The computing device has a processor and at least one application for recognizing the handwriting input under control of the processor. The at least one system application is configured to cause display of, on a display interface of a computing device, digital ink in a block layout in accordance with a layout of blocks of the handwriting input and a configuration of the computing device display interface.

Another aspect of the disclosed system and method provides each block of digital ink configured to be interactive with the at least one system application configured to cause change in the layout display in response to receiving interaction with the blocks.

Another aspect of the disclosed system and method provides the at least one system application configured to cause change in the layout display in response to aspect view changes of the display interface, where the configuration of the computing device display interface includes a first aspect view at a first orientation computing device and a second aspect view at a second orientation of the computing device. The first aspect view is portrait and the second aspect view is landscape.

Another aspect of the disclosed system and method provides the at least one system application configured to cause change in the layout display in response to display on computing devices having display interfaces of different horizontal and/or vertical dimensions, where the configuration of each computing device display interface is defined by horizontal and vertical dimensions of that display interface.

BRIEF DESCRIPTION OF THE DRAWINGS

The present system and method will be more fully understood from the following detailed description of the examples thereof, taken together with the drawings. In the drawings like reference numerals depict like elements. In the drawings:

FIG. **1** shows a block diagram of a computing device in accordance with an example of the present system and method;

FIG. **2** shows a block diagram of a system for handwriting recognition in accordance with an example of the present system and method;

FIG. **3** shows a block diagram illustrating detail of the handwriting recognition system of FIG. **2** in accordance with an example of the present system and method;

FIG. **4** shows a schematic view of an example visual rendering of a scroll page on a portion of an input surface of the computing device;

FIGS. **5**A and **5**B show schematic views of the scroll page with and without input content in different scrolled positions;

FIGS. **6**A and **6**B show schematic views of the scroll page with and without input content in different scrolled positions;

FIGS. **7**A and **7**B show schematic views of the scroll page with and without input content in different scrolled positions;

FIG. **8** shows a schematic view of an example visual rendering of the scroll page on the device input surface;

FIG. **9** shows a schematic view of an example visual rendering of the scroll page on the device input surface;

FIG. **10** shows a schematic view of an example of alignment elements of the scroll page;

FIG. **11** shows a graph of different column counts of the alignment elements with respect to input surface size of different devices;

FIGS. **12**A and **12**B shows tables of rules for adaption of the columns with respect to the input surface size;

FIG. **13** shows a schematic view of an example of alignment elements of the scroll page;

FIG. **14** shows a schematic view of an example visual rendering of the scroll page on a portion of the device input surface with the alignment elements;

FIG. **15** shows a schematic view of part of the scroll page with example input content object blocks;

FIGS. **16**A and **16**B shows a schematic view of part of the scroll page with object blocks and example operations thereon;

FIGS. **17**A and **17**B shows a schematic view of part of the scroll page with object blocks and example operations thereon;

FIGS. **18**A to **18**E shows a schematic view of part of the scroll page with object blocks and example operations thereon;

FIGS. **19**A to **19**C shows a schematic view of part of the scroll page with object blocks and example operations thereon;

FIGS. **20**A and **20**B shows a schematic view of part of the scroll page with object blocks and example operations thereon;

FIGS. **21**A and **21**B shows a schematic view of part of the scroll page with object blocks and example operations thereon;

FIG. **22**A shows a schematic view of a first alignment rendering of part of the scroll page with object blocks and FIG. **22**B shows a schematic view of a second alignment rendering of part of the scroll page with the object blocks;

FIGS. **23**A to **23**F show a schematic view of a first alignment rendering of part of the scroll page with successive input of object blocks;

FIG. **24** shows a schematic view of a second alignment rendering of the part of the scroll page of FIG. **23** with the object blocks;

FIG. **25** shows an example digital ink paragraph and its typeset equivalent;

FIG. **26** shows schematic views of an example visual rendering of the scroll page on a portion of the device input surface showing a typesetting operation on digital ink; and

FIG. **27** shows a schematic view of the scroll page of FIG. **26** showing an animation of the typesetting operation.

DETAILED DESCRIPTION

In the following detailed description, numerous specific details are set forth by way of examples in order to provide a thorough understanding of the relevant teachings. However, it should be apparent to those skilled in the art that the present teachings may be practiced without such details. In other instances, well known methods, procedures, components, and/or circuitry have been described at a relatively high-level, without detail, in order to avoid unnecessarily obscuring aspects of the present teachings.

Reference to and discussion of directional features such as up, down, above, below, lowest, highest, horizontal, vertical, etc., are made with respect to the Cartesian coordinate system as applied to the input surface on which the input to be recognized is made. Further, the use of the term 'text' in the present description is understood as encompassing all alphanumeric characters, and strings thereof, in any written language and common place non-alphanumeric characters, e.g., symbols, used in written text. Furthermore, the term 'non-text' in the present description is understood as encompassing freeform handwritten content and rendered text and image data, as well as non-alphanumeric characters, and strings thereof, and alphanumeric characters, and strings thereof, which are used in non-text contexts.

The various technologies described herein generally relate to capture, processing and management of handwritten content on portable and non-portable computing devices in a manner which allows conversion of that content into publishable documents. The systems and methods described herein may utilize recognition of users' natural writing or drawing styles input to a computing device via an input surface, such as a touch sensitive screen, connected to, or of, the computing device or via an input device, such as a digital pen or mouse, connected to the computing device. Whilst the various examples are described with respect to recognition of handwriting input using so-called online recognition techniques, it is understood that application is possible to other forms of input for recognition, such as offline recognition in which images rather than digital ink are recognized.

FIG. **1** shows a block diagram of an example computing device **100**. The computing device may be a computer desktop, laptop computer, tablet computer, e-book reader, mobile phone, smartphone, wearable computer, digital watch, interactive whiteboard, global positioning system (GPS) unit, enterprise digital assistant (EDA), personal digital assistant (PDA), game console, or the like. The computing device **100** includes components of at least one processing element, some form of memory and input and/or output (I/O) devices. The components communicate with each other through inputs and outputs, such as connectors, lines, buses, cables, buffers, electromagnetic links, networks, modems, transducers, IR ports, antennas, or others known to those of ordinary skill in the art.

The computing device **100** has at least one display **102** for outputting data from the computing device such as images, text, and video. The display **102** may use LCD, plasma, LED, iOLED, CRT, or any other appropriate technology that is or is not touch sensitive as known to those of ordinary skill in the art. At least some of the display **102** is co-located with at least one input surface **104**. The input surface **104** may employ technology such as resistive, surface acoustic wave, capacitive, infrared grid, infrared acrylic projection, optical imaging, dispersive signal technology, acoustic pulse recognition, or any other appropriate technology as known to those of ordinary skill in the art to receive user input. The input surface **104** may be bounded by a permanent or video-generated border that clearly identifies its boundaries.

In addition to the input surface **104**, the computing device **100** may include one or more additional I/O devices (or peripherals) that are communicatively coupled via a local interface. The additional I/O devices may include input devices such as a keyboard, mouse, scanner, microphone, touchpads, bar code readers, laser readers, radio-frequency device readers, or any other appropriate technology known to those of ordinary skill in the art. Further, the I/O devices may include output devices such as a printer, bar code printers, or any other appropriate technology known to those of ordinary skill in the art. Furthermore, the I/O devices may include communications devices that communicate both inputs and outputs such as a modulator/demodulator (modem; for accessing another device, system, or network), a radio frequency (RF) or other transceiver, a telephonic interface, a bridge, a router, or any other appropriate technology known to those of ordinary skill in the art. The local interface may have additional elements to enable communications, such as controllers, buffers (caches), drivers, repeaters, and receivers, which are omitted for simplicity but known to those of skill in the art. Further, the local interface may include address, control, and/or data connections to enable appropriate communications among the other computer components.

The computing device **100** also includes a processor **106**, which is a hardware device for executing software, particularly software stored in the memory **108**. The processor can be any custom made or commercially available general purpose processor, a central processing unit (CPU), a semiconductor based microprocessor (in the form of a microchip or chipset), a macroprocessor, microcontroller, digital signal processor (DSP), application specific integrated circuit (ASIC), field programmable gate array (FPGA) or other programmable logic device, discrete gate or transistor logic, discrete hardware components, state machine, or any combination thereof designed for executing software instructions known to those of ordinary skill in the art. Examples of suitable commercially available microprocessors are as follows: a PA-RISC series microprocessor from Hewlett-Packard Company, an 80x86 or Pentium series microprocessor from Intel Corporation, a PowerPC microprocessor from IBM, a Sparc microprocessor from Sun Microsystems, Inc., a 68xxx series microprocessor from Motorola Corporation, DSP microprocessors, or ARM microprocessors.

The memory **108** can include any one or a combination of volatile memory elements (e.g., random access memory (RAM, such as DRAM, SRAM, or SDRAM)) and nonvolatile memory elements (e.g., ROM, EPROM, flash PROM, EEPROM, hard drive, magnetic or optical tape, memory registers, CD-ROM, WORM, DVD, redundant array of inexpensive disks (RAID), another direct access storage device (DASD)). Moreover, the memory **108** may incorporate electronic, magnetic, optical, and/or other types of storage media. The memory **108** can have a distributed architecture where various components are situated remote from one another but can also be accessed by the processor **106**. Further, the memory **108** may be remote from the device, such as at a server or cloud-based system, which is remotely accessible by the computing device **100**. The memory **108** is coupled to the processor **106**, so the proces-

sor **106** can read information from and write information to the memory **108**. In the alternative, the memory **108** may be integral to the processor **106**. In another example, the processor **106** and the memory **108** may both reside in a single ASIC or other integrated circuit.

The software in the memory **108** includes an operating system **110** and an application **112**. The software optionally further includes a handwriting recognition (HWR) system **114** which may each include one or more separate computer programs. Each of these has an ordered listing of executable instructions for implementing logical functions. The operating system **110** controls the execution of the application **112** (and the HWR system **114**). The operating system **110** may be any proprietary operating system or a commercially available operating system, such as WEBOS, WINDOWS®, MAC and IPHONE OS®, LINUX, and ANDROID. It is understood that other operating systems may also be utilized.

The application **112** includes one or more processing elements related to detection, management and treatment of the layout of user input (discussed in detail later). The software may also include one or more other applications related to handwriting recognition, different functions, or both. Some examples of other applications include a text editor, telephone dialer, contacts directory, instant messaging facility, computer-aided design (CAD) program, email program, word processing program, web browser, and camera. The application **112**, and the other applications, include program(s) provided with the computing device **100** upon manufacture and may further include programs uploaded or downloaded into the computing device **100** after manufacture.

The HWR system **114**, with support and compliance capabilities, may be a source program, executable program (object code), script, application, or any other entity having a set of instructions to be performed. When a source program, the program needs to be translated via a compiler, assembler, interpreter, or the like, which may or may not be included within the memory, so as to operate properly in connection with the operating system. Furthermore, the handwriting recognition system with support and compliance capabilities can be written as (a) an object oriented programming language, which has classes of data and methods; (b) a procedure programming language, which has routines, subroutines, and/or functions, for example but not limited to C, C++, Pascal, Basic, Fortran, Cobol, Perl, Java, Objective C, Swift, and Ada; or (c) functional programming languages for example but no limited to Hope, Rex, Common Lisp, Scheme, Clojure, Racket, Erlang, OCaml, Haskell, Prolog, and F#. Alternatively, the HWR system **114** may be a method or system for communication with a handwriting recognition system remote from the device, such as server or cloud-based system, but is remotely accessible by the computing device **100** through communications links using the afore-mentioned communications I/O devices of the computing device **100**. Further, the application **112** and the HWR system **114** may operate together or be combined as a single application.

Strokes entered on or via the input surface **104** are processed by the processor **106** as digital ink. A user may enter a stroke with a finger or some instrument such as a pen or stylus suitable for use with the input surface. The user may also enter a stroke by making a gesture above the input surface **104** if technology that senses motions in the vicinity of the input surface **104** is being used, or with a peripheral device of the computing device **100**, such as a mouse or joystick. A stroke is characterized by at least the stroke initiation location, the stroke termination location, and the path connecting the stroke initiation and termination locations. Because different users may naturally write the same object, e.g., a letter, a shape, a symbol, with slight variations, the HWR system accommodates a variety of ways in which each object may be entered whilst being recognized as the correct or intended object.

FIG. **2** is a schematic pictorial of an example of the HWR system **114**, in either its local (i.e., loaded on the device **100**) or remote (i.e., remotely accessible by the device **100**) forms. The HWR system **114** includes stages such as preprocessing **116**, recognition **118** and output **120**. The preprocessing stage **116** processes the digital ink to achieve greater accuracy and reducing processing time during the recognition stage **118**. This preprocessing may include normalizing of the path connecting the stroke initiation and termination locations by applying size normalization and/or methods such as B-spline approximation to smooth the input. The preprocessed strokes are then passed to the recognition stage **118** which processes the strokes to recognize the objects formed thereby. The recognized objects are then output **120** to the display **102** generally as a typesetted version of the handwritten elements/characters.

The recognition stage **118** may include different processing elements or experts. FIG. **3** is a schematic pictorial of the example of FIG. **2** showing schematic detail of the recognition stage **118**. Three experts, a segmentation expert **122**, a recognition expert **124**, and a language expert **126**, are illustrated which collaborate through dynamic programming to generate the output **120**.

The segmentation expert **122** defines the different ways to segment the input strokes into individual element hypotheses, e.g., alphanumeric characters and mathematical operators, text characters, individual shapes, or sub expression, in order to form expressions, e.g., words, mathematical equations, or groups of shapes. For example, the segmentation expert **122** may form the element hypotheses by grouping consecutive strokes of the original input to obtain a segmentation graph where each node corresponds to at least one element hypothesis and where adjacency constraints between elements are handled by the node connections. Alternatively, the segmentation expert **122** may employ separate experts for different input types, such as text, drawings, equations, and music notation.

The recognition expert **124** provides classification of the features extracted by a classifier **128** and outputs a list of element candidates with probabilities or recognition scores for each node of the segmentation graph. Many types of classifiers exist that could be used to address this recognition task, e.g., Support Vector Machines, Hidden Markov Models, or Neural Networks such as Multilayer Perceptrons, Deep, Convolutional or Recurrent Neural Networks. The choice depends on the complexity, accuracy, and speed desired for the task.

The language expert **126** generates linguistic meaning for the different paths in the segmentation graph using language models (e.g., grammar or semantics). The expert **126** checks the candidates suggested by the other experts according to linguistic information **130**. The linguistic information **130** can include a lexicon, regular expressions, etc. The language expert **126** aims at finding the best recognition path. In one example, the language expert **126** does this by exploring a language model such as final state automaton (determinist FSA) representing the content of linguistic information **130**. In addition to the lexicon constraint, the language expert **126** may use statistical information modeling for how frequent a given sequence of elements appears in the specified lan-

guage or is used by a specific user to evaluate the linguistic likelihood of the interpretation of a given path of the segmentation graph.

The present system and method make use of the HWR system **114** in order to recognize handwritten input to the device **100**. As mentioned earlier, the application **112** includes application(s) for handling the layout of the recognized user input. Such applications may be provided in an architecture with separate layers for handling different processing. One or more of these layers may be remote to the device **100** accessible via the communications channels mentioned earlier. The layers may include application wrapper(s), platform wrapper(s) and (platform specific) application user interface(s).

The application **112** provided by the present system and method allows users, such as students, academic and working professionals, to take handwritten notes, such as during lectures, meetings and brainstorming sessions, which they can transform into sharable formatted documents on their portable or non-portable computing devices. The application allows taking of 'clean' notes which can be searched and memorized properly using their computing devices. The present system and method further allow natural writing to be input since the writing is made on (ruled) lines, and paragraphs and other layout elements are respected responsive to orientation (i.e., portrait or landscape) and device display capacity. Synchronization of the captured notes between devices and auto-saving of notes are also supported. Typesetting of the handwritten notes, via the HWR system, either on-demand or on-the-fly is also supported. Searching of both the recognized typeset text and the digital ink, such as with the systems and methods described in US Patent Application Publication US 2009/0077053 assigned to the present Applicant, the entire contents of which is incorporated by reference herein, is also supported, including indexation-on-the-fly. Export options to various formats, such as a text, Latex, HTML, image, pdf, etc., is also supported. These and other aspects are now described.

The application **112** utilizes a framework for the definition and handling of documents and document elements so as to provide structured content allowing layout definition and preservation. This document model handles different types of document elements, which are best described with HTML5 reference, for example, see W3C HTML5 DOM—Kinds of content: http://www.w3.org/TR/html5/dom.html#kinds-of-content. The document model utilized by the application **112** does not reproduce HTML specifications, rather it takes inspiration from them. A subset of the elements of the document model is flow, sectioning, heading, phrasing, embedded and interactive.

Regarding flow, most elements that are used in the body of documents and applications are categorized as flow content. Content flow basically works just like an HTML DOM, being a structured tree of nodes rendered vertically into main text view. All input goes by default into content flow, cascading vertically from top of the document and left-aligned for left-to-right written languages or right-aligned for right-to-left written languages. Sectioning content is content that defines the scope of headings and footers. Heading content defines the header of a section (whether explicitly marked up using sectioning content elements, or implied by the heading content itself). Phrasing content is the text of the document, as well as elements that mark up that text at the intra-paragraph level. Runs of phrasing content form paragraphs. Embedded content is content that imports another resource into the document, or content from another vocabulary that is inserted into the document. Interactive content is content that is specifically intended for user interaction. These document elements, and others elements, are utilized in the document model with respect to the content.

An initial aspect of the document model and feature of the application **112** is the provision of an effectively infinite vertical scrolling interface for the input of handwritten notes and other content by a user. A thousand years ago, content was written on scrolls that one would unroll to read/write. The Codex (ancestor of the book) was invented based on user experience optimization as large rolls were cut into pages thereby providing random and faster access to any part of the content, instead of unrolling the scroll from start to end. This completely and permanently changed the recording of written content, and the book has been the one and only content holder for the last millennium. However, the current digital era in which writing is performed on a dynamic screen interface provides the potential for different paradigms to be adopted. Accordingly, note taking applications and associated user interfaces need not blindly mimic the page paradigm, rather solutions which offer more or enhanced features over either simple page-turn or (infinite) scroll should be evaluated.

Conventional handwritten note taking applications are based on separated and fixed-dimensions pages, imitating real world pages. This 'card' approach is convenient for users to intuitively embrace creating content using handwriting on-screen, however it does not fit with other digital document contexts which all have unlimited height, such as:

- Websites, which care freely scrollable
- ePubs, which are paginated and freely scrollable
- text editors, which use pagination with freely scrollable pages.

The Applicant has found that users of existing digital note taking applications have issues understanding how text behaves when editing, with text reflow and uncontrolled text/object position. In order to lower the learning curve for use of such applications, handwriting should be closer to what users are familiar with. That is, writing with keyboard in text editing software/web services, with the same model, behaviors and infinite scrollable pages. Any digital device user is already accustomed to scrolling (either reading or writing). Scrolling is also a natural and intuitive pattern on touch devices. A quick solution in the handwritten note taking environment would be to just change the way pages are laid out, such as:

- fixed page size (defined by device screen or viewport when creating content)
- pages stacked vertically
- allow page breaks for users who prefer paginated content
- allow free vertical scroll for users who prefer scrolling.

However, such a solution does not address all the issues discussed earlier. For example, users would still face layout issues when typesetting. Accordingly, the note taking application of the present system and method provides a typeset 100% compatible solution as it utilizes a pure vertical scroll model featuring an infinite-height single page. This model allows:

- users to write on an infinite page or 'scroll page' as in a digital document, where:
  - users are not limited by fixed page size;
  - scroll pages can be sliced into fragments, with one fragment being about 500 sections or more, for example;
  - time to load a large scroll page of content is about 50 milliseconds;

users can scroll using gestures on or above the touch sensitive surface of the device interface, such as using two fingers

users to select page mode display where:

the infinite page is broken into sections;

a visualization option is provided to select between displaying the sections on top of each other or displaying top and bottom margins between sections as a strong visual separator of vertical pages

responsive display in which notes are displayed correctly independent of device size/orientation

users to find existing content and sections easily via navigation

conversion of existing notes into scroll pages flawlessly.

Using vertical scroll for digital note taking means that writing is performed in a content flow instead of absolute positioned blocks, with all text cascading vertically just like an HTML page or a word processing document. By default, vertical content order defines model structure and all layout changes have impact on all content below, pushing down or pulling up. Accordingly, these changes to the users' layout must be considered and taken account of. However, users should still be provided with the ability to write anywhere on the page, i.e., both vertically and horizontally, create two-dimensional objects, move things around and create layout on-the-fly. These abilities need to be balanced however so that a 'perfect' looking responsive document is created. This means:

all input goes into content flow

writing offset from margins creates floating blocks that are anchored to content flow

document model can create invisible container nodes to handle responsive layout of floating blocks

document model regularly edits and moves content from node/fragment to another to reflect layout/model changes in real-time

all paragraphs/blocks are aligned to a (vertical and horizontal) grid during content input (along with reflow action and typeset/beautification).

These and other features of the present system and method are now described in detail. FIG. **4** shows a schematic view of an example visual rendering of a scroll page **400** on a portion of the input surface **104** of an example computing device **100**. The scroll page **400** is shown in editing view in which all content input is performed. Other views may be used for document structure editing and notebook navigation. Besides providing user interface (UI) controls (described later), the main object of the editing view is to provide a full-width page that overflows vertically from the display **102** of the device **100**. The infinite scroll page **400** embodies a page of flexible height. In order to maximize writing space for users (particularly on smaller devices), an empty scroll page is defined to occupy at least the whole editing view height.

In the editing view (and other views as appropriate) several page elements are displayed including a top padding **402**, a left margin **404** and a right margin **406**. The padding and margins define an input area **408** which has a line pattern background **410**. The line pattern **410** has horizontal lines separated by a multiple of the vertical rhythm height unit being the density independent pixel (dp). Regardless of the particular device **100**, the vertical distance between the horizontal lines **410** is defined by a line pattern unit (LPU) and the vertical rhythm height unit provides a graduated measure of the LPU on a particular device. For example, the LPU may be set at about one centimeter for any device being a certain multiple of the vertical rhythm height unit or users may be allowed to customize the line pattern unit to a different multiple of the vertical rhythm height unit according to their writing style. Alternatively, the vertical rhythm may be based on typeset text size (e.g., the minimum text size) and the LPU is provided as a multiple of this typeset text size. All lines **410** are displayed with the same very light and subtle color, e.g., grey, that is just between visible and faded. In this way the line pattern is noticeable but unobtrusive so as to guide the handwriting input without distracting from the note taking itself. The line pattern background **410** is displayed in handwritten text context and serves multiple purposes:

forces users to write onto it so that handwriting recognition is optimized

helps users adopt regular vertical rhythm writing leading to same writing size and better block dimensions and positions (described later)

gives users a hint about expected content, for example, dedicated backgrounds can be displayed for other contexts, such as drawings and equations

helps users control line skipping to define paragraphs (described later).

The top padding **402** is displayed on any new page or section and does not contain the line pattern background. This helps users write with proper vertical spacing, i.e., not too close from the interface **104** border and leaves room for other elements, such as section titling and tagging. The top padding **402** is defined by multiples of the LPU, e.g., in the example of FIG. **4** the top padding is two times the LPU.

The left and right margins **404** and **406** are displayed with vertical lines. In the example of FIG. **4** these lines are similar to the line pattern **410** in color and definition, however these lines may be more boldly displayed if desired. The vertical line margins are positioned from the corresponding left and right 'page' edges with a multiplying factor of the vertical rhythm unit. This multiplying factor changes according to the particular device **100**, defining different margin width calculations. Example multiplying factors are detailed in Table 1.

TABLE 1

Device	Screen size	Multiplier	Margin width
Smartphone	less than about 17 cm (<7")	2/3	32 dp
Phablet	about 17 to about 25 cm (7" < 10")	3/3	48 dp
Tablet	about 25 to about 32 cm (10" < 13")	4/3	64 dp
Laptop computer	about 32 cm to about 37 cm (13" < 15")	5/3	80 dp
Desktop computer	about 37 cm to about 50 cm (15" < 21")	6/3	96 dp
Television	greater than about 50 cm (>21")	8/3	128 dp

The margins **404**, **406** are displayed in handwritten text context, and serve multiple purposes:

help users write with proper alignment for direction of written language, and not too close from interface border

allow secure single-point scrolling within right and/or left margin (described later)

offer dedicated zone in left and/or right margin for content tagging

offer dedicated zone in left and/or right margin for slice gesture to define sections of the content input on the scroll page.

In the example of FIG. **4**, the margins **404**, **406** are not displayed in the top padding **402** and start from the first line pattern **410** horizontal line. However, the margins may extend into the top padding if desired. Further, the line pattern **410** is not displayed in the margin elements in order to discourage writing in these areas. However, it may be desired to allow writing in the margins for annotations and the like. Alternatively, the margins may only be displayed on hover-over or contact interaction with the horizontal extents of the page **400** or totally omitted, thereby, for example, relying on natural handwriting behaviors of users to not write too close to the screen/in edges.

FIGS. **5**A to **7**B show schematic views of the example visual rendering of the scroll page **400** with and without input content in various scrolled positions in relation to the interface **104** of the device **100**. In the present example, the default height of an empty scroll page is slightly larger than the height of the interface **104** so as to compensate for the top padding **402** and to allow discoverability of the scrolling feature of the application **112** by users. The empty page height is defined by the editing view height plus the top padding height, such that the editing view extends below a viewable area **132** of the interface **104**. That is, as shown in FIG. **5**A, the empty scroll page has the padding **402** within the viewable area **132** and an extend area **412** outside of the viewable area. In FIG. **5**B, the empty scroll page has been scrolled upwards, using a multi-point scroll gesture A (described later), such that the padding **402** is outside of the viewable area **132** and the extend area is now part of the viewed scroll page **400**. This empty page height is adapted when the scroll page is rendered on other devices with different screen sizes.

Any time content input occurs, the page height extends accordingly and remains scrollable to offer an uninterrupted way of writing by removing paper size limitations. That is, as shown in FIG. **6**A, the height of input content B defines a content area **414** and a further extend area **416** is defined at the bottom of the scroll page **400** having the same dimensions. In this way, whatever content is entered, users are able to push it upwards outside of view to "clear" writing space without needing to create a new section (as shown in FIG. **6**B), and even without writing any content the page is scrollable so that users understand the navigable aspects of the scroll page **400**.

Any content input updates page properties in real-time (or as soon as possible), allowing users to scroll and push content outside of view. Page height extension equals content input height (starting from top of the first line to bottom of the last line=number of lines+1). This gives on a tablet for instance:

1 line of content: (1+1)×vertical rhythm unit=2×48=96 dp
2 lines of content: (2+1)×vertical rhythm unit=3×48=144 dp
4 lines of content: (4+1)×vertical rhythm unit=5×48=240 dp
and so on . . .

Content input may occur at any position on the page **400**, with offset from the topmost position, or may be moved after input to any position on the page. That is, as shown in FIG. **7**A, the height of an offset C of the input content B from the top padding **402** defines an offset area **418** and a further extend area **420** is defined at the bottom of the scroll page **400** having the same dimensions. In this way, whatever content is entered, users are able to push it upwards outside of view to "clear" writing space without needing to create a new section (as shown in FIG. **7**B). Accordingly, the page height extension integrates a content offset, giving on a tablet for instance:

1 line of content with 1 line offset: ((1+1)+(1))×vertical rhythm unit=3×48=144 dp
2 lines of content with 3 lines offset: ((2+1)+(3))×vertical rhythm unit=6×48=288 dp
4 lines of content with 2 lines offset: ((4+1)+(2))×vertical rhythm unit=7×48=336 dp
and so on . . .

Vertical scroll may be handled differently depending on the ability of the computing device **100** to differentiate a users' finger from a stylus or pen (by which the device defines passive and active styli) or the ability of a stylus to indicate or communicate to the device that it is being used for handwriting or the ability of users to provide such an indication. By default, and in devices which do not differentiate, any single-point touch or hover event within the input area **408** is to be considered as content input. Thus, in order for a user to scroll the page **400** without inputting new content a multi-point interaction can be used anywhere on the input area, such as the multi-point vertical scroll gesture A using two fingers touching or hovering over the interface **104** at about the same time shown in FIG. **8**.

However, in order to allow scrolling without forcing multi-point interaction, a dedicated area may be defined in one or more of the margins, e.g., in FIG. **8** the right margin **406** is defined with a scroll zone **422**. A single-point gesture D (shown as a one-finger or stylus gesture touching or hovering over the interface **104** in FIG. **8**) which starts in this zone and has subsequent movement that (wholly or partly) occurs within this zone is not considered as content input but as a scroll gesture. The width of the scrolling area may be defined by the margin width or independent therefrom. For example, the scroll zone may be defined by touch target dimensions of users' fingers and stylus tips (about 48 dp) so that the scrolling area remains the same even with responsive resizing of the margins (e.g., wider with bigger devices).

Once a scroll sequence is launched (either with multi-point anywhere or single-point in the margin area) and until the page stops moving, scrolling may be continued using just a one-point gesture anywhere. When using a device that is able to differentiate finger from stylus, or a stylus which indicates its use to the device, or the user provides this indication, thus defining an active stylus for content input for example, multi-point interaction is not necessary and scrolling is available with a one-finger vertical scroll gesture anywhere on the interface surface **104**.

The scroll page **400** is not automatically scrolled during writing input. This allows users to have control of page position. However, automatic scrolling does occur when a user drags selected content near the top or bottom edges of the viewable area so as to allow dropping anywhere on the infinite page. That is, when selecting any type of content (text or objects), users may want to reposition that content anywhere else on the page including above or below the visible part of the page. In the example shown in FIG. **9**, content **424** is selected (in a manner described later) and then dragged or pushed (with sequence display of the dragging) to move the selected content into automatic scroll areas **426** temporarily defined during selection and/or dragging/pushing close to the top or bottom edges of the viewable area **132**. Once the content is dragged/pushed into the scroll areas **426**, the scroll page **400** is scrolled with the scrolling speed bound to the drag position, i.e., the closer to the edge of the page **400** the faster the page scrolls. As long as content is not

dropped, both top/bottom automatic scrolling areas are active to allow users to scroll in the opposite direction if scrolling went too far, for example. When content is dropped and the drag/push sequence ends, top/bottom automatic scrolling areas are disabled so that content input is re-enabled on the full viewable page.

The scroll page can be provided as either a constraint-free canvas that allows users to create object blocks (blocks of text, drawings, etc.) anywhere without worrying about sizing or alignment. However, in order to provide professionally formatted documents containing structured content, an alignment pattern may be defined onto which all content is to be aligned/defined (either at input or along user interaction). In one example of the present system and method, the alignment pattern is defined as responsive to the device size/orientation and is not itself displayed but used by the application **112** to define alignment of content elements. Various elements of an example alignment pattern **1000** are now described with reference to FIGS. **10** to **14**.

The alignment pattern **1000** has left and right padding areas **1002** that prevent content from being input too close to the edges of the interface **104** of the device **100**. The size of the padding areas **1002** is based on the text size related to the root element font size (rem), e.g., the font size of the <html> element. As with CSS styling (see, for example, https://developer.mozilla.org/en/docs/Web/CSS/length), the rem unit represents the default font size of the root element (e.g., one rem=16 pixels (px) for web browsers). The rem unit is considered as a responsive unit with text size being built on a modular scale for the responsive vertical rhythm unit (dp). The rem unit principle is that when the default font size for the root element is changed, all other properties inherit from this single change and all elements are adapted accordingly. However, as some browser applications may not support the rem unit, such layout can be achieved using the "em" unit instead. This definition of unit size is related with mobile device operating system (OS) Dynamic Type which implements the same behavior with other wording and variables (see, for example, https://developer.apple-.com/library/ios/documentation/StringsTextFonts/Conceptual/TextAndWebiPhoneOS/CustomTextProcessing/CustomTextProcessing.html#//apple_ref/doc/uid/TP40009542-CH4-SW65).

The area between the padding areas **1002** defines the available horizontal space as the input area **408**. The alignment pattern **1000** defines a variable number of columns **1004**, which are all the same width and flexible to fill up the whole available horizontal space between the padding **1002** (shown divided with dashed lines). Accordingly, the input area **408** is equally divided into the columns **1004** as shown in FIG. **10**. The columns define breakpoints as minimum application width values in dp, e.g., the screen width when the application **112** is provided in full-screen or just window width when reduced, such as on desktop/laptop, or split-screen on tablet.

Column sizing depends on the column count, so column width is defined by the percentage of the whole available horizontal space, that is the input area **408** (e.g., four columns =25%). The column count is provided so as to provide minimum content display within each column so that meaningful layout can be provided. FIG. **11** shows a graph of different column counts depending on the screen size of different devices. Certain device examples in accordance with this graph are (in full screen context):

iPhone ® 6 (portrait) =	375 dp =	2 columns
Nexus 6 (landscape) =	730 dp =	6 (or 8) columns
iPad Air ® (portrait) =	768 dp =	6 (or 8) columns
Surface ® Pro 3 (landscape) =	960 dp =	8 columns
HP ® Chromebook 14" =	1366 dp =	8 columns
Dell ® Inspiron ® 23" =	1920 dp =	16 columns

The devices and column counts shown in FIG. **11** are merely illustrative of the concept of using columns to provide responsive layout display. The actual values may be different, such that more or less columns are used for any given device. For example, for large screen devices, such as big-screen televisions, electronic/interactive whiteboards, and large-screen touch monitors, the propensity of users to handwrite using larger character sizes may be accommodated by using a lower column count than illustrated in FIG. **11**, and for small screen devices, such as smartphones, the ability of users to handwrite using small character sizes due to precise stylus use may be accommodated by using a higher column count than illustrated in FIG. **11**. When changing window size or rotating the device, the alignment pattern adds or retrieves columns along which content has to adapt and align/snap to new columns (described later). Various column-wide blocks are adapted using the rules detailed in FIGS. **12**A and **12**B.

The alignment pattern **1000** further has a gutter space **1006** (shown in FIG. **13**) defined about vertical boundaries of each column **1004** on both the left and right edges so as to prevent object blocks from being positioned too close to each other. Like the other elements of the alignment pattern, the gutters **1006** are defined relative to the rem unit and define actual guides onto which layout of object blocks **1008** is performed (see FIG. **14**). This means that all object blocks are aligned/snapped onto the gutter edges and not column edges themselves and the UI page margins **404**, **406** are positioned onto the leftmost and rightmost column gutters (see FIGS. **13** and **14**). In this way, the alignment pattern elements are all defined using a text-related unit since the layout spacing depends on text size. The gutters could be omitted if desired and absolute column boundaries used instead.

As explained previously, the metrics used are based on both the input area **408** width and the rem unit for responsive sizing. These different metrics are related as shown in Table 2.

TABLE 2

Minimum width (dp)	No. of columns	Column width (% of input area width)	Margin width (rem)	Gutter width (rem)
0	1	100	0	0.75
240	2	50	1.5	0.75
480	4	25	1.5	0.75
720	6	16.66666667	2.25	0.75
960	8	12.5	2.25	0.75
1440	12	8.33333333	3.0	1.0
1920	16	6.25	4.0	1.0

As mentioned earlier, snap-to-pattern behavior is used for object block alignment. Vertical snap is handled through reference to the line pattern **410** in two ways, either by making sure that all object block heights are integer multiples of the LPU or at least the top boundary of each block is aligned with the next available line of the line pattern background **410**. In the former case, whatever the user interaction (e.g., new, edited, moved, resized), all object blocks are always LPU-rounded and positioned according to

vertical LPU (e.g., top and bottom boundaries are aligned with the line pattern background **410**). Either way a LPU portion of a snap-to-grid mechanism is provided. For horizontal snap, whatever the user interaction (e.g., new, edited, moved, resized), left and right boundaries of all object blocks stay aligned with the columns **1004** (or gutters **1006**) in any case. This provides a layout portion of the snap-to-grid mechanism. Accordingly, the vertical and horizontal snap operations provide vertical alignment and horizontal alignment to the LPU (vertical) and layout (horizontal) grid.

Alternatively, in order to provide fluid interaction and let users feel that they are in control of the layout, any move/resize of an object block may be updated in real-time and aligned with current touch or hover position during the interaction (i.e., not aligned or snapped with the LPU and layout grid). For instance, during block resize (detailed later) object block boundaries are updated with exact touch or hover position even though it might be in the middle of a column/line.

Besides updating other surrounding elements in real-time to preview the overall result, users should have a visual feedback indicating what layout will become once an object is snapped onto the LPU and layout grid. One possible feedback would be to display a darkened/colored snapped-to-grid overlay beneath the interacted block during interaction. Once a user stops interacting with an object block, all block boundaries are snapped to the LPU and layout grid with a swift animation. When writing, text appears substantially where the finger or stylus is in relation to the device interface so as to preserve the free writing experience for the user. However, paragraph boundaries are aligned with the LPU and layout grid no matter what (discussed in more detail later). For example, the left boundary of the paragraph is aligned with closest left gutter **1006** and the right boundary of the paragraph is aligned with closest right gutter **1006**. Further, in order to assist users' perception of appropriate spacing display of the line pattern background within the gutters of at least a current input column may be omitted during input. For example, during input in the second column of a four column grid, the line pattern background is omitted/masked in the left and right gutters of the second column, and may further be omitted/masked in the right gutter of the first column and the left column of the third column. Internal paragraph features, such as indentations (e.g., text horizontally offset from paragraph left origin for left-to-right written languages) may either be aligned with the columns or stored as a property, for example in the memory **108**, for each line of the paragraph unrelated with LPU and layout grid sizing.

Certain example operations involving object block creation and management are now described with reference to FIGS. **15** to **24**. The examples shown in these drawings are in a left-to-right written language context, and therefore any reference to positions can be adapted for written languages having different directional formats. In each of FIGS. **15** to **24** part of the scroll page **400** is shown in schematic form as depicted in the viewable area of the device interface. The scroll page **400** is shown with the LPU and layout grid **1500** configured from the line pattern background **410**, as horizontal lines **1502** of the grid **1500**, and the alignment pattern **1000**, defined by vertical lines **1504** of the grid **1500** which divide the page **400** into a number of columns **1004**, e.g., six columns in all the drawings (except FIGS. **22**B and **24** in which three columns are depicted). The top padding and margins of the scroll page, and the padding areas and column gutters of the alignment pattern are omitted in the drawings (except in FIG. **22**) for ease of description. Accordingly, except in FIG. **22**, the layout elements described are shown aligned to the vertical lines **1504** of the grid **1500** rather than the column gutters. As described earlier, the alignment grid itself is not displayed to users and acts as an invisible guide for the management of the layout of object blocks for the application **112**, however the line pattern background **410** may be displayed to users to provide guidance on writing size and placement.

The layout management employed by the present system and method handles the way object blocks are defined and can be manipulated by users generally through use of simple gestures. This is done in a way that lets users feel they are in control of the layout, such that there are limited automatic behaviors and some flexibility is provided for users to position the content as desired. As the digital note taking application **112** is primarily for the handwritten capture of notes, the main object block can be considered to be a paragraph of text or a text block, with sub-elements being line breaks and alinea (i.e., subsets of paragraphs between two line breaks). The management of paragraphs includes the ability for users to create multiple paragraphs, split paragraphs, merge paragraphs, remove content in a paragraph, move paragraphs and resize paragraphs. Other object blocks that can be created and need to be managed, particularly in relation to the text blocks, include content such as images, drawings and equations.

The layout management provides the ability to create the different object blocks in a desired layout and perform operations on the separate object blocks, such as:

- block insertion or drag/push & drop (where there is empty room) to right or left of a block/paragraph
- block insertion or drag/push & drop (where there is empty room) below or above a block/paragraph
- insertion inside a paragraph (inline)
- insertion in between paragraphs.

In FIG. **15** a first paragraph is shown as a margin-aligned paragraph **1506**, which is aligned to the left-most or first vertical line **1504** of the grid **1500** and spans four of the horizontal lines **1502** of the grid **1500**, or four LPUs. The paragraph contains handwritten text depicted as solid horizontal lines. A second paragraph is shown as a floating text block **1508**, which is aligned to the second vertical line **1504** of the grid **1500** and spans four LPUs. The snapped-to-grid extent of the floating text block **1508** is shown by the dashed boundary **1508***a*, inside of which the grid pattern is suppressed for clarity. The concepts of 'margin-aligned' and 'floating' in the context of the present example scroll page **400** are provided to distinguish those blocks which are aligned to a margin of the page, such as paragraphs of text that are left- or right-aligned on a page of a document editor application, and those blocks which are freely (with constraint to the grid) positioned relative to the margins. As shown the text of the paragraphs **1506**, **1508** itself is not snapped to the grid so that the user does not gain the feeling that their control is lost. The actual input position of the text within the paragraph is provided by defining and storing, for example in the memory **108**, the text indentation which is a property of each line of a paragraph corresponding to the distance between the start of the content and the closest column on the left of the paragraph.

Another floating block **1510** is shown which is aligned to the fourth vertical line **1504** of the grid **1500** and spans five LPUs. The floating block **1510** may contain text or a drawing, image, diagram, equation, etc. Both aligned and floating blocks are anchored to a part or section of the scroll page, with the margin-aligned blocks being further anchored to the margin, and move vertically with it as the page is

scrolled. If the block is moved (i.e., selected to be dropped somewhere else), it is associated to the new part or section of the scroll page to which it is moved (described in detail later). It is noted that the boundaries of the illustrated blocks are those used by the present system and method to determine the extent of the input elements in relation to the alignment pattern and as such are not typically displayed by the application **112** to users. However, during editing operations, such as movement of blocks, etc., the boundaries may be displayed to assist users in proper placement of the elements relative to other elements of a layout.

FIG. **16**A shows several paragraphs residing in margin-aligned and floating blocks. In particular, a first paragraph **1600** has been created spanning four columns, followed by a second paragraph **1602** horizontally adjacent the first paragraph **1600** and spanning two columns, and a third paragraph **1604** vertically below both the paragraphs **1600** and **1602** and spanning all six columns. The use of blocks and the alignment grid allow content within blocks to be edited while retaining the relative layout of the blocks. For example, in FIG. **16**B the user has added further text to the second paragraph **1602** on the next horizontal line **1502** of the alignment grid **1500**, however since the paragraph **1604** is below the paragraph **1602** in the vertical flow of the scroll page **400**, the lower paragraph **1604** is consequentially moved down to the next horizontal line **1502** of the alignment grid **1500** so as to retain the spacing of two LPUs between the paragraphs **1602** and **1604**. Indeed, any content below is shifted down by the same amount so that relative layout is retained, since the user has not specifically changed those layouts. Accordingly, text written upstream in a block moves below content down if there is not enough space to expand the text block. This mechanism is the same as that of the extension of the scroll page **400** by the height of input content described earlier with reference to FIG. **6**A, such that the content moved below is moved out of the viewable area **132** of the interface **104**.

Alternatively, if there is enough space, then the text block will be expanded without movement of lower blocks. For example, FIG. **17**A shows the paragraphs **1600**, **1602** and **1604** but with the third paragraph **1604** only being vertically below the first paragraph **1600** and spanning only four columns. As such, as shown in FIG. **17**B, when the user adds further text to the second paragraph **1602** on the next horizontal lines **1502** of the alignment grid **1500** the text block bounding the second paragraph **1602** can freely expand adjacent the lower paragraph **1604**.

The operation of writing text upstream causing movement of the whole below content downwards if there is not sufficient layout space for that text is applicable to whatever content is below, i.e., text or non-text blocks. For example, in FIG. **18**A a paragraph **1800** has been created and an object block **1802** (either text or non-text) has been created (or dropped) vertically below the paragraph **1800**. In FIG. **18**B the user has added further text to the paragraph **1800** on the last text line carrying onto the next horizontal line **1502** of the alignment grid **1500**. However, since the block **1802** is below the paragraph **1800** in the vertical flow of the scroll page **400**, the block **1802** is consequentially moved down to the next horizontal line **1502** of the alignment grid **1500** so as to retain the spacing of three LPUs between the objects **1800** and **1802**.

In the example of FIGS. **16** to **18**, the text added to the existing paragraph (or being added during creation of the paragraph) is a continuation of the text of the paragraph onto the next line of the line pattern background **410**, where either it is written directly on the next line or added to the existing last line of text and continued onto the next line. In either case, it is considered that the added text belongs to the existing paragraph since it is added to the immediate next line **410** in the vertical flow. Determination of the added text actually being part of a new paragraph could be made using the recognition stage **118** of the HWR system **114**, particularly with reference to the offset of the text of the adjacent lines and their relationship to the left margin or aligned edge of the (existing) paragraph. However, the result of such recognition is not known by the user. That is, without some specific indication (such as, display of a paragraphing symbol or automatic shifting of the existing and/or new paragraph blocks, which may be too intrusive on the note taking process and therefore distracting for the user) misrecognition of separate paragraphs would not been known until some manipulation of the layout occurs (such as display of the note on a different device or in a different orientation) by which time context of the original input may be forgotten.

Accordingly, a simple constraint is placed on the input of separate paragraphs (and object blocks) in the vertical flow, in a manner which is easily understood and applied by users. This constraint is that there is always to be an empty line **410** between object blocks, e.g., a spacing of two LPUs, which overlap horizontally, e.g., overlap in at least one column **1004**. This rule provides a way of cleanly maintaining relative layout between object blocks in the vertical flow of the scroll page **400** which can be applied to directly input content and dragged & dropped or pushed & dropped blocks.

For example, in the operation described in relation to FIGS. **18**A and **18**B, the three LPU spacing between the paragraph **1800** and the block **1802** was maintained when vertical content was added to the paragraph **1800**. Alternatively, because the spacing after addition of the extra content to the paragraph **1800** remains equal to or more than two LPUs, the block **1802** (and any other lower object blocks) need not be consequently moved. Accordingly, if space between new and existing content is larger than one line, nothing is moved downward in the vertical stream, letting the new content fill up available space. This is shown in the example of FIG. **18**C where a paragraph **1806** is input two lines **410** below the last line **410** of the paragraph **1800** (e.g., the lower side of boundary **1800***a* of the block **1800** is spaced one LPU from the upper side of boundary **1804***a* of the block **1804**, such that the text lines are two LPUs apart) and since there remains an empty line **410** between the new paragraph **1804** and the block **1802**, the block **1802** is not moved (e.g., the lower side of the boundary **1804***a* of the block **1804** is spaced one LPU from the upper side of boundary **1802***a* of the block **1802**, such that if the block **1802** is a text block the text lines are two LPUs apart). As such, this rule is more correctly defined as the text lines of text blocks must be spaced apart by two LPUs, meaning that vertically adjacent block boundaries can be spaced one LPU apart.

Whenever new content reaches existing content (leaving less than one empty line therebetween), the whole content below is moved one line down to reintroduce the one empty line spacing. Similarly, each time new content is written on an empty line between object blocks, the content below is moved one line down. Similarly, each time a block is added upstream on a new line the content below is moved down and there should always be a line above and below the block. An example of this is shown in FIG. **18**D, in which an object block **1806** is dropped onto the line **410** below the paragraph **1800** (such that there is a spacing of one LPU therebetween) and the paragraph **1804** and the block **1802** are consequently

moved down so that there is a spacing of one LPU between the new block **1806** and the paragraph **1804**, with no relative layout change between the blocks **1802** and **1804**. This example is further extended in FIG. **18**E where the block **1802** is moved above the paragraph **1804** and dropped one LPU spacing below the block **1806** with consequential downward movement of the paragraph **1804** so that one LPU spacing remains between the lower side of the boundary **1802***a* of the block **1802** and the upper side of the boundary **1804***a* of the block **1804**.

As described earlier, blocks can be dragged & dropped or pushed & dropped from their current position in the scroll page **400** to a new position, or resized, perhaps with consequential movement of other blocks and with guiding mechanism for users. For example, FIGS. **19**A to **19**C show an operation in which a paragraph **1900** and a non-text block **1902** (e.g., containing an image) are laid out and the non-text block **1902** is moved. The initial position of the block **1902** shown in FIG. **19**A is horizontally adjacent the paragraph **1900**. If the block **1902** is moved horizontally as shown in FIG. **19**B from its initial position, during this movement a caret **1904** is displayed showing the next available horizontal position in which the block **1902** can be dropped, e.g., in the example shown the block **1092** has been moved over the paragraph **1900** and therefore the next available drop position is on the left margin where the caret **1904** is displayed. Dropping the block **1902** in this suggested position will cause consequential horizontal movement of the paragraph **1900** snapped to the next available column of the alignment grid **1500**, e.g., the third column **1004**. Similarly, if the block **1902** is moved vertically as shown in FIG. **19**C from its initial position, during this movement the caret **1904** is displayed showing the next available vertical position in which the block **1902** can be dropped, e.g., in the example shown the block has been moved below the paragraph **1900** and therefore the next available drop position is on the line **410** below the paragraph **1900**. Similar mechanisms can be used when resizing object blocks. By guiding users in this way, alignment of objects to other objects or to the alignment grid itself is provided as much as possible. Besides blocks, text selected in a paragraph can itself be inserted inside another paragraph by drag & drop (or other editing mechanisms such as push, copy, cut, paste) or if dropped in or pushed to empty space, a new text block is created with that text.

When moving existing content, the operations described earlier can be used with the corresponding consequential behavior of the layout. When creating new text content, such as a new paragraph, users simply need to handwrite on the next available line of the line pattern background, following the empty line between rule described earlier. When creating new non-text content, such as a drawing or equation, if users were to simply begin inputting the non-text content on the next available line for example, the HWR system **114** would be required to detect the input of the non-text content so that the handwriting recognition is not carried out on the new content. Alternatively, users can be provided with a mechanism to indicate that non-text content is to be input. Conventionally this has been done through menu selections or the use of specialized styli, such as a drawing pen versus a writing pen. The present system and method simplifies this indication by leveraging existing touch interface actions known to users.

That is, in order to input non-text content users merely double tap on empty space upon which the application automatically creates a block area or canvas in which the user can insert an image, drawing, diagram, equation, etc. In this way, a double tap on empty space is interpreted as the user desiring to create a block object which has a different context than just text. The application **112** can then use other recognition mechanisms to determine and recognize the actual content input, if handwritten, such as the systems and methods described in US Patent Application Publication US 2015/0228259, US Patent Application Publication No. 2016/0154998 filed Dec. 2, 2014 entitled "System and Method for Recognizing Geometric Shapes", and European Application No. 15290183.1 filed Jul. 10, 2015 and U.S. application Ser. No. 14/870,735 filed Sep. 30, 2015 both entitled "System for Recognizing Multiple Object Input and Method and Product for Same", all assigned to the present Applicant and the entire contents of which are incorporated by reference herein.

Like the block management so far described herein, the generation of the non-text input canvas is performed in accordance with the alignment grid/pattern. That is, when the page **400** in the viewable area **132** of the interface **104** is empty, the generated canvas takes all the available space in the viewable area **132**. Alternatively, when the canvas is to be generated horizontally next to an object block, the generated canvas takes the available width to the corresponding margin/padding area and aligns to the height of the existing object block. Further, if there is content above the to-be-generated canvas, the generated canvas takes the available space below the existing block and an empty line is left between the existing block and the canvas (e.g., a spacing of one LPU) so that it is not included in the content above. Further still, if there is content above and below the to-be-generated canvas, the generated canvas takes the available full width of the scroll page and the available height between the existing blocks with an empty line left above and below the canvas relative to the existing blocks (e.g., a spacing of one LPU top and bottom) so that it is not included in the existing content. FIGS. **20** and **21** show examples of two of these operations.

In FIG. **20**A a paragraph **2000** is shown as a margin-aligned block spanning two columns **1004** and eight lines **410**, such that the boundary **2000***a* of the block **2000** is aligned on the second vertical line **1502** and the eighth horizontal line **1504** of the alignment pattern **1500**. In response to receiving a double tap gesture E to the right of the paragraph **2000** a block **2002** is created from the second vertical line **1502** to the last vertical line **1502** of the alignment pattern **1500** and with the same vertical extent as the paragraph **2000** (e.g., to the eighth horizontal line **1504**) as shown in FIG. **20**B. Non-text input within the block **2002** can then be performed.

In FIG. **21**A, a paragraph **2100** is shown as a full page width (two margin-aligned) block spanning all six columns **1004** and four lines **410**, such that the boundary **2100***a* of the block **2000** is aligned on the fourth horizontal line **1504** of the alignment pattern **1500**. In response to receiving a double tap gesture E below the paragraph **2100** a block **2102** is created from the fifth horizontal line **1504** of the alignment pattern **1500** to the vertical extent of the page **400** as shown in FIG. **21**B. Non-text input within the block **2102** can then be performed. In a further example, if the canvas to be generated is too near the edge of the input area **408** of the scroll page **400** to provide meaningful content input, the scroll page could be automatically scrolled. With respect to the top and bottom extents of the input area **408**, by setting a minimum canvas width and height (e.g., say, two columns and two LPUs), automatic scrolling could be achieved similar to that described earlier in relation to FIG. **9**, since when a canvas is to be generated such that part of the

minimum canvas overlaps the automatic scroll areas upon generation the scroll page will be automatically scrolled. Such a minimum canvas size also provides a way to ensure vertical overflow of a canvas below horizontally adjacent objects (e.g., to the left of the canvas) or of the horizontally adjacent objects (e.g., to the right of the canvas) when the canvas is to be created too close to the margin extents of the scroll page or between objects that are not sufficiently spaced apart.

Users may subsequently determine that the generated canvas is either too small or large for the purposes intended. As such, resizing of the created non-text block can be subsequently performed before or after content input. Resizing of all object blocks is possible under given conditions and with certain consequential actions on the layout. For example, resized objects snap to the alignment grid, resized content is always fully visible in the note (e.g., not cropped), except for images if necessary. With respect to image resizing, images imported into or as non-text blocks in the scroll page **400** are embedded within a bounding box that serves as a layout container. Image positioning within its bounding box is handled as following without user interaction:

- image is scaled up/down so that it always covers the whole bounding box area (overflowing parts being cropped)
- image always keeps its aspect ratio in any case (e.g., no stretching)
- image is centered within its bounding box.

Resizing of images, and of other object blocks, is performed using handles (see FIG. **9**) displayed around a selected image and bound with the bounding box, not the inner image itself. Access to the handles, with corresponding display of the block boundaries, is provided by the present system and method in a natural and well understood manner to users, such as by the user simply performing a single tap on the content to be resized/edited. As is understood by those skilled in the art, once displayed users touch or hover over the handles and drag/push the selected handle across the interface so as to provide different behaviors with dragging/pushing of corner handles causing scaling of the bounding box (moves two sides, thus keeping the bounding box aspect ratio) and dragging/pushing of side handles causing stretching of the bounding box (move single side, thus changing bounding box aspect ratio). Changing the bounding box aspect ratio has no impact on the inner image aspect ratio, rather the image is automatically scaled/centered.

Other types of non-text blocks and text blocks can be explicitly resized in the same manner. However, when a paragraph is resized the font size of the text content is not changed (i.e., scaled) only the paragraph's dimensions (see detail discussion later for exceptions). With respect to the layout, if a block is made smaller content below is moved upward and if a block is made larger content below is moved downward, in accordance with afore-described rules.

As previously described, automatic resizing of blocks also occurs due to user input of content within blocks. For example, when users write in a text block (e.g., adding to or continuing a paragraph), the block extends automatically as the users write, with alineas and font size being maintained (and with any consequential movement of lower content). Automatic resizing can also occur due to display of a note created on one device being subsequently displayed on a different device having a screen of a different size or a change in orientation of the device display, e.g., landscape to portrait. As described earlier, the users' created layout of note taking object blocks is respected as much as possible by the present system and method during such transitions, as is the content within the blocks themselves. As such, operations such as automatic resizing are kept to a minimum and only performed where necessary or where the users' original input is not adversely effected. Upon translation, if users desire resizing or scaling of content, they are able to perform such operations manually, for example, in the manner explained above.

As mentioned, the computing device **100** on which the application **112** is utilized may have an interface **104** which responds to a change in the orientation of the device from portrait to landscape and vice-versa. Accordingly, the layout and characteristics of content elements included in a note of a scroll page of the digital note taking application must also be responsive whilst following the same principles of minimal undesired effect on layout. The same applies to displaying a particular scroll page, or note book, on devices having different display sizes. The alignment grid, and in particular the layout pattern, of the present system and method provides an effective way of providing responsiveness to such transitions. An example transition is now described with reference to FIGS. **22**A and **22**B.

FIG. **22**A shows an example of the scroll page **400** on a device interface **104** on which six columns **1004** of the layout pattern **1000** can be meaningfully provided. The column gutters **1006** of the columns **1004** are also shown. A first object block **2200** spans the first column **1004**, a second object block **2202** spans the second and third columns **1004** and a third object block **2204** spans the fourth column **1004**, where the blocks **2200**, **2202** and **2204** are horizontally adjacent and the vertical boundaries of each block are aligned with the corresponding column gutters **1006**, as previously described.

FIG. **22**B shows the scroll page **400** translated to a different device interface **104** or the device interface **104** in a different orientation in which only three columns **1004** of the layout pattern **1000** can be meaningfully provided. Accordingly, while the first and second object blocks **2200**, **2202** fit within the three columns **1004** of the translated scroll page **400**, the third object block **2204** which spanned four of the columns **1004** in the scroll page **400** of FIG. **22**A does not. Accordingly, the third block **2204** is moved vertically to the next available line **410** of the line pattern background (e.g., leaving a one LPU spacing between it and the closest block above) and is margin-aligned so as to now span the first column **1004** aligned with the corresponding column gutters **1006**. Any content below is also moved downward as previously described. In this way, the original layout is maintained as much as possible whilst providing meaningful and well-sized display of the layout content in different views of the scroll page **400**. For example, the margin-alignment of the vertically moved, previously floating, block is done so that it is clear that the block initially cam directly after the previously left-adjacent horizontal block.

By this responsive layout process, the present system and method is able to maintain layout in situations where a note is created/displayed by an initial device in an initial orientation and then sequentially displayed on smaller/larger devices in same/different orientations, where the display in the original (first) format is maintained when the note is displayed in that format again. In order to achieve this, the relative positioning of the object blocks in the layout relative to the alignment pattern needs to be known. The application **112** handles this by determining the relative block positions and creating sections or divisions of the scroll page to manage retention of block proximity when translating the

scroll page view. When there is a horizontal/vertical overlay of more than one element/block in the vertical/horizontal direction a division is created. The elemental content of a division is not changed when the elements of the division are moved within the division either through direct action of users or due to translation of the scroll page in line with the viewable area of the device interface. The elemental content of a division is changed when elements are removed from the division or editing actions on the content are performed.

FIGS. **23**A to **23**F illustrate an example division scheme of the present system and method. In determining whether to divide the layout or not, the present system and method first determine in the vertical direction whether there are elements which horizontally overlap one another by checking the horizontal direction for each vertically displaced element, and whenever overlap is found the direction of the search changes from vertical to horizontal to vertical and so on.

FIG. **23**A shows an example of the scroll page **400** with one object block **2300** input in the interface **104**. The block spans three columns **1004** and two LPUs **410**. In determining whether to divide the layout or not, the present system and method first determine the presence of multiple vertical elements and then determine in the vertical direction whether there are elements which horizontally overlap by checking the horizontal direction for each vertically displaced element. Whenever overlap is found the direction of the search changes from vertical to horizontal to vertical and so on. Thus, in the example of FIG. **23** the following occurs:

A1. Processing begins at the block **2300** (being the first element in the vertical direction) and with a search of the vertical direction below (e.g., within the four column span of the block **2300**) it is determined that there are no vertically overlapping blocks. Accordingly, a division is not created.

A2. Processing continues at the block **2300** and with a search of the horizontal direction to the left and right (e.g., within the two LPU span of the block **2300**) it is determined that there are no horizontally adjacent blocks. Accordingly, a division is not created.

A3. Division processing ends, as there are no further blocks of content in either direction.

FIG. **23**B shows the scroll page **400** with a second object block **2302** input below the first block **2300**. The second block **2302** spans one column **1004** and three LPUs **410**. As a further block has been input the following occurs:

B1. Step A1 is repeated but this time it is determined that the block **2302** is vertically overlapped with the block **2300**. Accordingly, a division **2304** containing the blocks **2300** and **2302** is created.

B2. With the creation of the division **2304**, it is determined whether any sub-divisions should be created at the block **2300** by repeating step A2 which achieves the same result in the present example.

B3. Processing continues at the block **2302** (being the next element in the vertical direction) and with a search of the vertical direction below (e.g., within the single column span of the block **2302**) it is determined that there are no vertically overlapping blocks. Accordingly, the division **2304** is maintained (e.g., there is no need to extend the division to contain another block).

B4. With the confirmation of the division **2308**, it is determined whether any sub-divisions should be created by processing continuing at the block **2302** and with a search of the horizontal direction to the left and right (e.g., within the three LPU span of the block **2302**) it is determined that there are no horizontally adjacent blocks. Accordingly, a sub-division is not created.

B5. Division processing ends, as there are no further blocks of content in either direction.

FIG. **23**C shows the scroll page **400** with a third object block **2306** input horizontally adjacent the second block **2302**. The third block **2306** spans three columns **1004** and three LPUs **410**. As a further block has been input the following occurs:

C1. Steps B1 to B3 are repeated with resulting maintenance of the division **2304**.

C2. Step B4 is repeated but this time it is determined that the block **2306** is horizontally adjacent on the right. Accordingly, the division **2304** is updated so as to contain all three blocks **2300**, **2302** and **2306** and a sub-division **2308** containing the blocks **2302** and **2306** is created.

C3. With the creation of the sub-division **2308**, it is determined whether any further elements are within the sub-division **2308** by processing continuing at the block **2306** (being the next element in the horizontal direction within the sub-division **2308**) and with a search of the horizontal direction to the left and right (e.g., within the three LPU span of the block **2306**) it is determined that only the block **2302** is horizontally adjacent on the left, which is already known. Accordingly, the sub-division **2308** is maintained (e.g., there is no need to extend the division to contain another block).

C4. With the creation and confirmation of the sub-division **2308**, it is determined whether any further sub-divisions within the sub-division **2308** should be created by processing continuing at the block **2306** and with a search of the vertical direction below within the sub-division **2308** (e.g., within the three column span of the block **2306**) it is determined that there are no vertically adjacent blocks within the sub-division **2308**. Accordingly, a sub-division is not created.

C5. Division processing ends, as there are no further blocks of content in either direction.

FIG. **23**D shows the scroll page **400** with a fourth object block **2310** input below the existing blocks. The further block **2310** spans three columns **1004** and two LPUs **410**. As a further block has been input the following occurs:

D1. Steps B1 and B2 are repeated with resulting maintenance of the division **2304**.

D2. Step B3 is repeated but this time it is determined that the block **2310** is vertically overlapped with the block **2302**. Accordingly, the division **2304** is updated to contain the blocks **2300**, **2302**, **2306** and **2310**.

D3. Steps C2 to C4 are repeated with resulting maintenance of the updated division **2304** and the sub-division **2308**.

D4. Division processing ends, as there are no further blocks of content in either direction.

FIG. **23**E shows the scroll page **400** with the third object block **2306** having been resized to still span three columns **1004** but now span one LPU **410** rather than three. As an editing operation on the layout has occurred, the division processing is performed but with the result that the existing divisions are maintained as they are since the relative layout of the blocks has not been changed.

FIG. **23**F shows the scroll page **400** with a fifth object block **2312** input below the (resized) third object block **2306**. The further block **2312** spans one column **1004** and one LPU **410**. As a further block has been input the following occurs:

F1. Steps D1 and D2 are repeated with resulting maintenance of the division **2304**.

F2. Steps C2 and C3 are repeated with resulting maintenance of the division **2304** and the sub-division **2308**.

F3. Step C4 is repeated but this time it is determined that the block **2312** is vertically overlapped with the block **2306**. Accordingly, a further sub-division **2314** containing the blocks **2306** and **2312** is created.

F4. With the creation of the sub-division **2314**, it is determined whether any further elements are within the sub-division **2314** by processing continuing at the block **2312** (being the next element in the vertical direction within the sub-division **2314**) and with a search of the vertical direction below within the sub-division **2314** (e.g., within the three column span of the block **2312**) it is determined that there are no vertically adjacent blocks within the sub-division **2314**. Accordingly, the sub-division **2314** is maintained (e.g., there is no need to extend the division to contain another block).

F5. With the creation and confirmation of the sub-division **2314**, it is determined whether any further sub-divisions within the sub-division **2314** should be created by processing continuing at the block **2302** and with a search of the horizontal direction to the left and right (e.g., within the single LPU span of the block **2312**) it is determined that only the block **2302** is horizontally adjacent on the left, which is already known. Accordingly, a sub-division is not created.

F6. Division processing ends, as there are no further blocks of content in either direction.

The example layout of FIG. **23** is therefore divided into three divisions or sections. With the first or root division containing all content blocks, the second division containing three blocks and the third division containing two blocks. In the example of FIG. **23** the divisions are created, updated and processed incrementally as input is made to the application **112**. These divisions are therefore stored in the memory **108**, for example, by the application **112** as they are created and updated so that the order of the blocks is maintained through movement of individual blocks within or between the divisions either through user interaction or display/orientation changes (described in relation to a specific example later). The division processing is also applicable to an already created layout of blocks (e.g., when an existing note is loaded on the application **112**). Whilst the root division and first level sub-division (e.g., the sub-division **2304**) are shown in FIG. **23** as surrounding the blocks contained therein, it is understood that these higher level divisions basically divide the width of the scroll page **400** into sections.

FIG. **24** shows the scroll page **400** of FIG. **23** translated to a different device interface **104** and/or device interface **104** in a different orientation. In response to the translation, the application **112** has rendered the scroll page **400** with three columns **1004** of the layout pattern **1000**. Accordingly, while the blocks **2300** and **2310** of the root division **2304** fit within the three columns **1004** of the translated scroll page **400**, the combined blocks **2302**, **2306** and **2312** of the sub-division **2308** which spanned four of the columns **1004** in the scroll page **400** of FIG. **23** do not. However, unlike the situation depicted in FIG. **22**A, since the blocks **2306** and **2312** are further contained in the sub-division **2314**, then the sub-division **2314** as a whole is moved vertically to the next available line **410** of the line pattern background with consequential movement of the content block **2312** (e.g., leaving a one LPU spacing between the upper and lower blocks of the sub-division **2314** and the closest blocks above and below) with the left-most block (i.e., the block **2306**) being margin-aligned. In this way, the original layout is maintained as much as possible, both in the main vertical flow of elements and in the horizontal order of elements across the width of the scroll page, whilst providing meaningful and well-sized display of the layout content in different views of the scroll page **400**. Since the vertical and horizontal orders are maintained, the display of FIG. **24** is properly translated back to the display of FIG. **23**F (e.g., the layout to that original input) when the original device orientation or size is restored.

In the example of FIG. **23**, there was sufficient horizontal space (e.g., columns) in the divisions for the content added to those divisions. On the other hand, if added content or objects do not fit into the available width of the scroll page **410** within a division, all objects of the division may be automatically resized (as described earlier) depending on where the added object is input or dropped within the division and the objects to the right and below consequently moved down. Resizing of digital ink paragraphs or text blocks also involves reflow of the digital ink onto the more or less lines of the line pattern background. This reflow is handled so that the meaning and integrity of the paragraph is maintained. Accordingly, a particular paragraph of digital ink (and typeset text—see later) may be reflowed differently to another paragraph, and in any case, the minimum width of a paragraph is governed by the minimum word width (with or without hyphenation) of the text within the paragraph.

Automatic resizing may also be performed with a change of orientation or format so that the position of a block or paragraph in the layout and its percentage of the vertical occupation in the width of the scroll page is respected. For example, for handwritten text in a paragraph that has a certain relative vertical position and occupies a certain percentage of the viewable area width, say 75%, the same relative position and percentage is applied when changing the device orientation or size by reflowing the text so that more words can fit into the paragraph or the height of the paragraph is changed or the size of the text characters is adjusted accordingly. Text size is particularly relevant to the typesetting of the digital ink.

The manner of converting the input digital ink to typeset text has an impact on the layout as well. This is because, characters handwritten on either paper or on-screen are typically much larger than typeset text, which is generally rendered using standard OS settings (e.g., 16 px font size and 24 px line height) and are generally of varying size and alignment (the line pattern background assists in reducing the extent of these variations however). Accordingly, when a paragraph of digital ink is typeset the resulting typeset text paragraph is much smaller. An example of the relative sizes of a paragraph before and after typesetting is shown in FIG. **25** in which a simple multiline paragraph of digital ink **2500** having a span of six lines **410** (e.g., six LPUs) shrinks down to a multiline paragraph **2502** of only three lines (e.g., no longer a measure of LPUs) of typeset text when typeset. This is not such an issue for vertically laid out paragraphs which are typeset one-by-one or all together after writing is completed since the relative layout of the paragraphs remains the same. However, for horizontally laid out paragraphs and non-text blocks adjacent the paragraphs which are typeset drastic changes in the layout may occur if automatic movement of the surrounding object blocks is performed.

Alternatively, the adjacent blocks can be left as is, such as by not resizing the block (boundaries) of the paragraph which has been typeset. However, since users are unaware of

the existence of the blocks, since the block boundaries are only displayed during movement and editing for example, the size of the resultant typeset paragraph may seem odd with respect to the surrounding elements causing users to attempt moving those blocks closer to the typeset paragraph which may not be allowed due to the size of the enlarged block surrounding the typeset paragraph.

It is clear that handwriting requires a larger space than typeset for content to be input correctly (e.g., a larger line height) and the line pattern background is generally needed to assist the recognition process. Accordingly, since the line height of the digital ink is larger than that of the typeset ink, the present and system method removes display of the line pattern background from typeset paragraphs thereby making the content much clearer and readable, and ready to be shared. An example of this is shown in FIG. **26** in which a digital ink paragraph **2600** having the line pattern background **410** displayed is converted into a typeset paragraph **2602** with display of the line pattern background suppressed in the text block **2602** containing the typeset paragraph.

Transitioning from handwriting to typeset needs to be quick to prevent distracting users from content input. Accordingly, in the present system and method a simple, natural and well-understood mechanism to manually typeset paragraphs is provided and animation of the transformation is used with a simple mix of transparency (0% to 100%) and scaling (0% to 100%). In one example, when a user desires to typeset a paragraph, they simply double tap on the paragraph and in the resultant animation the origin of the transformation is aligned with the absolute middle of the smallest state (being typeset). An example of such transform animation is shown in FIG. **27** in which the transformation from the digital ink paragraph **2600** to the typeset paragraph **2602** is shown. If the user is unhappy with the size of the typeset paragraph in relation to the other elements they can resize the paragraph block, for example, through performing a single tap on the content to select the content, as described earlier. Further, reversal of the transformation may be provided by re-performing a double tap on typeset content. Alternatively, users could see the animation and resultant typeset paragraph, which either temporarily replaces or overlays the digital ink paragraph (as shown in FIG. **27**) by performing other natural and well understood operations, such as a long press or hover over the digital ink paragraph, before committing to the transformation.

Once a paragraph is typeset users may desire to change or add content to the typeset content by handwriting. The present system and method supports this in a manner which adheres to the afore-described rules for producing good quality and shareable digital notes. This is done by requiring the typeset paragraph to be displayed in a larger format. In this larger format, the line pattern background is displayed in the typeset paragraph block with the font size of the typeset text enlarged so as reside on the lines of the line pattern background. In this way, any handwritten text or editing gestures supported by the HWR system **114**, such as strike-through, scratch-out, overwrite, insertion, etc., are made in relation to the LPU thereby supporting proper recognition. The temporary transformation to the larger typeset size can be achieved, for example, by the user performing a simple and well understood operation, such as a single- or multi-point gesture like a pinch-out. In this situation the new handwritten text remains in this format, e.g., the paragraph is a mix of typeset and digital ink, until the user performs conversion (e.g., in the manner described earlier) at which the enlarged typeset text reverts to the usual sizing at the same time as the added digital ink being typeset to the same size. In this way, it is unnecessary for the HWR system **114** to perform re-recognition of all the paragraph content and the layout is maintained.

Additional to typesetting, simple and well understood gestures may be performed by users to interact with, edit and create new content within the notes. For example, when inserting a non-text canvas as described with reference to FIGS. **20** and **21**, full screen mode of that canvas may be entered, for example, by users performing a double tap on the inserted canvas area. Further, within a digital ink or typeset paragraph, interaction with specific text, e.g., words, phrases, sentences, may be done, for example, by users performing a long-press or hover over the text to be selected with or without resultant selection handles for changes to the selection (as described earlier). Further, in the spaces of the input area **408** outside of the defined object blocks, editing gestures supported by the HWR system **114**, such as strike-through, scratch-out, insertion, etc., could be performed by users to add or suppress space about existing elements (with consequential downstream movement of any object blocks below) without movement or resizing of the elements themselves.

The present system and method provide an application which allows users, such as students, academic and working professionals, to take handwritten notes, such as during lectures, meetings and brainstorming sessions, on their computing devices in a meaningful and shareable format akin to word processed documents. The present system and method further allow natural writing to be input since the writing is made on (ruled) lines, and paragraphs and other layout elements are respected responsive to orientation (i.e., portrait or landscape) and device display capacity.

Further interaction with the handwritten or typeset text, or non-text content, created in a note, such as editing the content, manipulating the layout of the notes, or converting or adding the notes into a document, can be performed directly within the application itself without the need to import the note into a separate document processing application. As such, the original layout of the handwritten notes is maintained, as is the actual input handwriting itself, the digital ink. Accordingly, annotations, decorations, special layout, etc. is maintained for easy comprehension of the notes taken.

While the foregoing has described what is considered to be the best mode and/or other examples, it is understood that various modifications may be made therein and that the subject matter disclosed herein may be implemented in various forms and examples, and that they may be applied in numerous other applications, combinations, and environments, only some of which have been described herein. Those of ordinary skill in that art will recognize that the disclosed aspects may be altered or amended without departing from the true spirit and scope of the subject matter. Therefore, the subject matter is not limited to the specific details, exhibits, and illustrated examples in this description. It is intended to protect any and all modifications and variations that fall within the true scope of the advantageous concepts disclosed herein.

We claim:

1. A system for digital note taking on computing devices, each computing device comprising a processor and at least one non-transitory computer readable medium for recognizing handwriting note taking input under control of the processor, the at least one non-transitory computer readable medium configured to:

cause display of, on a display interface of a computing device, digital ink in a block layout in accordance with

a layout of geometric blocks of the handwriting input and a configuration of the computing device display interface, said configuration of the computing device being defined by horizontal and vertical dimensions of the display interface, and wherein the at least one non-transitory computer readable medium is configured to: when the computing device display interface changes from a configuration whereby the geometric blocks of digital ink fit in horizontal adjacency within the horizontal dimension of the display interface to another configuration of reduced horizontal dimension, the reduced horizontal dimension being less than a cumulative horizontal geometric block dimension of the geometric blocks, cause moving of said one or more geometric blocks relatively to said others of geometric blocks while otherwise maintaining the layout of the said others of geometric blocks.

2. The system of claim **1**, wherein each geometric block of digital ink is configured to be interactive, the at least one non-transitory computer readable medium configured to cause change in the layout display in response to receiving interaction with the geometric blocks.

3. The system of claim **1**, wherein the configuration of the computing device display interface includes a first aspect view at a first orientation computing device and a second aspect view at a second orientation of the computing device, the at least one non-transitory computer readable medium configured to cause change in the layout display in response to aspect view changes of the display interface.

4. The system of claim **3**, wherein the first aspect view is portrait and the second aspect view is landscape.

5. The system of claim **1**, wherein the configuration of each computing device display interface is defined by horizontal and vertical dimensions of that display interface, the at least one non-transitory computer readable medium configured to cause change in the layout display in response to display on computing devices having display interfaces of different horizontal and/or vertical dimensions.

6. The system of claim **1**, wherein the display interface is scrollable.

7. A method for digital note taking on computing devices, each computing device comprising a processor and at least one non-transitory computer readable medium for recognizing handwriting note taking input under control of the processor, the method comprising:

causing, with the at least one non-transitory computer readable medium, display of, on a display interface of a computing device, digital ink in a block layout in accordance with a layout of geometric blocks of the handwriting input and a configuration of the computing device display interface, said configuration of the computing device being defined by horizontal and vertical dimensions of the display interface, and wherein the at least one non-transitory computer readable medium is configured to: when the computing device display interface changes from a configuration whereby the geometric blocks of digital ink fit in horizontal adjacency within the horizontal dimension of the display interface to another configuration of reduced horizontal dimension, the reduced horizontal dimension being less than a cumulative horizontal geometric block dimension of the geometric blocks, cause moving of said one or more geometric blocks relatively to said others of geometric blocks while otherwise maintaining the layout of the said others of geometric blocks.

8. The method of claim **7**, wherein each geometric block of digital ink is configured to be interactive, the at least one non-transitory computer readable medium configured to cause change in the layout display in response to receiving interaction with the geometric blocks.

9. The method of claim **7**, wherein the configuration of the computing device display interface includes a first aspect view at a first orientation computing device and a second aspect view at a second orientation of the computing device, the method causing, with the at least one non-transitory computer readable medium, change in the layout display in response to aspect view changes of the display interface.

10. The method of claim **9**, wherein the first aspect view is portrait and the second aspect view is landscape.

11. The method of claim **7**, wherein the configuration of each computing device display interface is defined by horizontal and vertical dimensions of that display interface, the method causing, with the at least one non-transitory computer readable medium, change in the layout display in response to display on computing devices having display interfaces of different horizontal and/or vertical dimensions.

12. The method of claim **7**, wherein the display interface is scrollable.

13. A non-transitory computer readable medium having a computer readable program code embodied therein, said computer readable program code adapted to be executed to implement a method for recognizing input of handwriting note taking to a computing device, the computing device comprising a processor and at least one system non-transitory computer readable medium for recognizing the input under control of the processor, the method comprising:

causing, with the at least one system non-transitory computer readable medium, display of, on a display interface of a computing device, digital ink in a block layout in accordance with a layout of geometric blocks of the handwriting input and a configuration of the computing device display interface, said configuration of the computing device being defined by horizontal and vertical dimensions of the display interface, and wherein the at least one non-transitory computer readable medium is configured to: when the computing device display interface changes from a configuration whereby the geometric blocks of digital ink fit in horizontal adjacency within the horizontal dimension of the display interface to another configuration of reduced horizontal dimension, the reduced horizontal dimension being less than a cumulative horizontal geometric block dimension of the geometric blocks, cause moving of said one or more geometric blocks relatively to said others of geometric blocks while otherwise maintaining the layout of the said others of geometric blocks.

14. The non-transitory computer readable medium of claim **13**, wherein each geometric block of digital ink is configured to be interactive, the at least one non-transitory computer readable medium configured to cause change in the layout display in response to receiving interaction with the geometric blocks.

15. The non-transitory computer readable medium of claim **13**, wherein the configuration of the computing device display interface includes a first aspect view at a first orientation computing device and a second aspect view at a second orientation of the computing device, the method causing, with the at least one system non-transitory computer readable medium, change in the layout display in response to aspect view changes of the display interface.

16. The non-transitory computer readable medium of claim **15**, wherein the first aspect view is portrait and the second aspect view is landscape.

17. The non-transitory computer readable medium of claim **13**, wherein the configuration of each computing device display interface is defined by horizontal and vertical dimensions of that display interface, the method causing, with the at least one system non-transitory computer readable medium, change in the layout display in response to display on computing devices having display interfaces of different horizontal and/or vertical dimensions.

18. The non-transitory computer readable medium of claim **13**, wherein the display interface is scrollable.

* * * * *

EXHIBIT I

US010082951B2

(12) **United States Patent**
Guiavarc'h et al.

(10) **Patent No.: US 10,082,951 B2**
(45) **Date of Patent: Sep. 25, 2018**

(54) **SYSTEM AND METHOD FOR DIGITAL INK INTERACTIVITY**

(71) Applicant: **MyScript**, Nantes (FR)

(72) Inventors: **Jean-Nicolas Guiavarc'h**, Nantes (FR); **Stéphane Guyetant**, Nantes (FR); **Gildas Lemarié**, Nantes (FR); **Alain Chateigner**, Nantes (FR)

(73) Assignee: **MyScript**, Nantes (FR)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **15/083,195**

(22) Filed: **Mar. 28, 2016**

(65) **Prior Publication Data**

US 2017/0199660 A1 Jul. 13, 2017

(30) **Foreign Application Priority Data**

Jan. 7, 2016 (EP) 16290002

(51) **Int. Cl.**

G06K 9/00	(2006.01)
G06F 3/0488	(2013.01)
G06F 3/0354	(2013.01)
G06T 7/00	(2017.01)

(52) **U.S. Cl.**

CPC ***G06F 3/04883*** (2013.01); ***G06F 3/03545*** (2013.01); ***G06K 9/00416*** (2013.01); ***G06T 7/0087*** (2013.01)

(58) **Field of Classification Search**

USPC 382/189

See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,500,937	A	3/1996	Thompson-Rohrlich	
5,864,636	A *	1/1999	Chisaka	G06K 9/00416 345/179
2015/0169975	A1 *	6/2015	Kienzle	G06K 9/34 382/189
2016/0012285	A1 *	1/2016	Ahn	G06K 9/00422 382/186
2016/0154997	A1 *	6/2016	Kim	G06F 3/04883 382/189

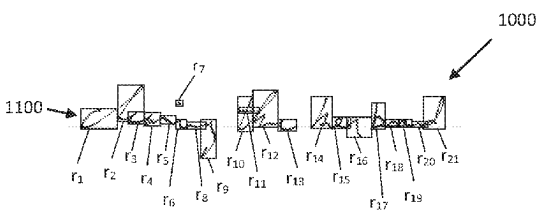
* cited by examiner

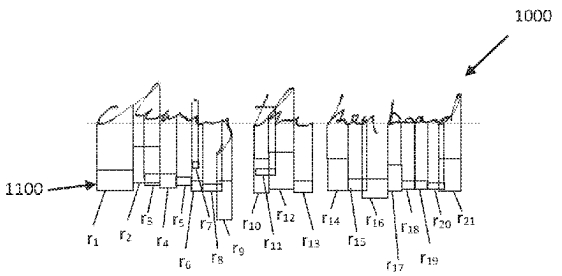
Primary Examiner — Jerome Grant, II
(74) *Attorney, Agent, or Firm* — McDermott Will & Emery LLP

(57) **ABSTRACT**

A system, method and computer program product for use in providing interactive ink from handwriting input to a computing device are provided. The computing device is connected to an input device in the form of an input surface. A user is able to provide input by applying pressure to or gesturing above the input surface using either his or her finger or an instrument such as a stylus or pen. The present system and method monitors the input strokes. The computing device further has a processor and an ink management system for recognizing the handwriting input under control of the processor. The ink management system is configured to cause display of, on a display interface of a computing device, first digital ink in accordance with first handwriting input, allocate references to ink elements of the first digital ink, map the references to corresponding recognized elements of the first handwriting input, and determine and store, in the memory of a computing device, ink objects including the references and mapped recognized elements.

18 Claims, 11 Drawing Sheets





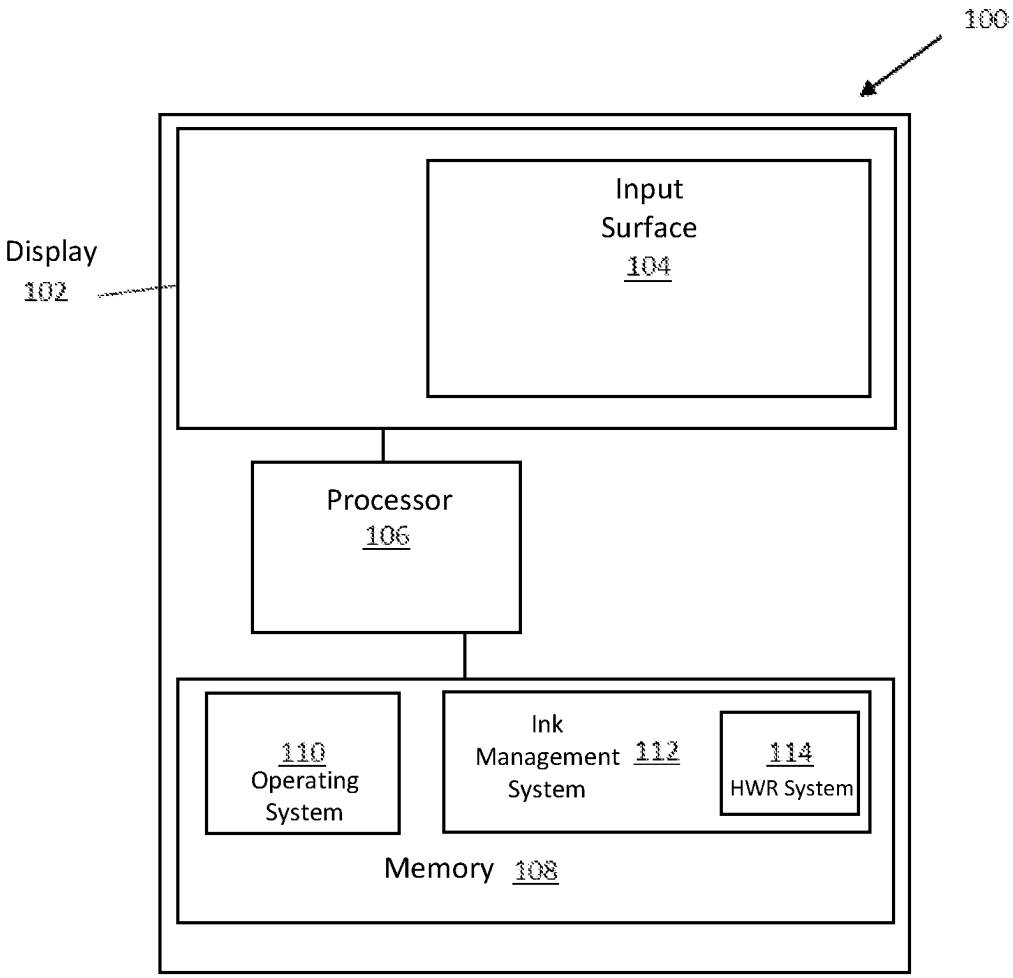

100
Display
102
Input
Surface
104
Processor
106
110
Operating
System
Ink
Management
System
112
114
HWR System
Memory
108

FIG. 1

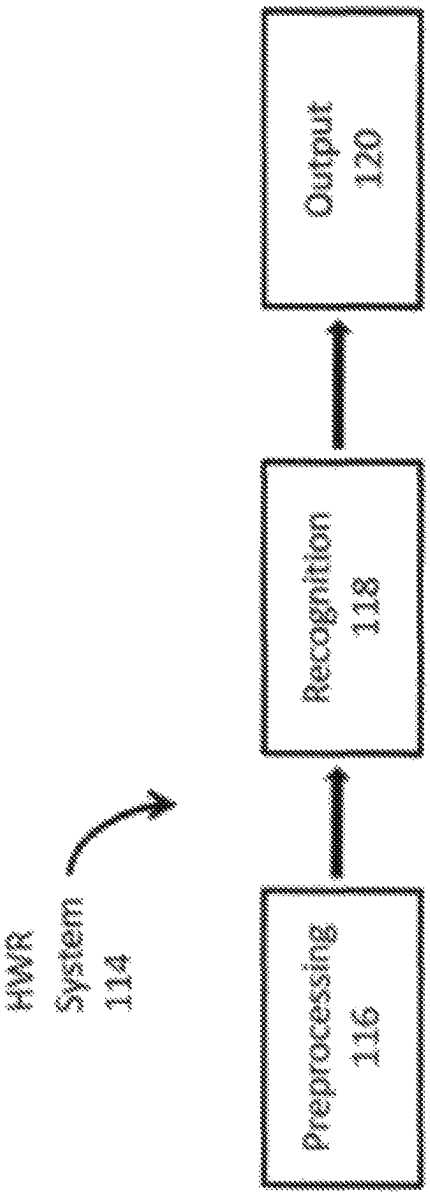

Output
120
Recognition
118
Preprocessing
116
HWR
System
114

FIG. 2

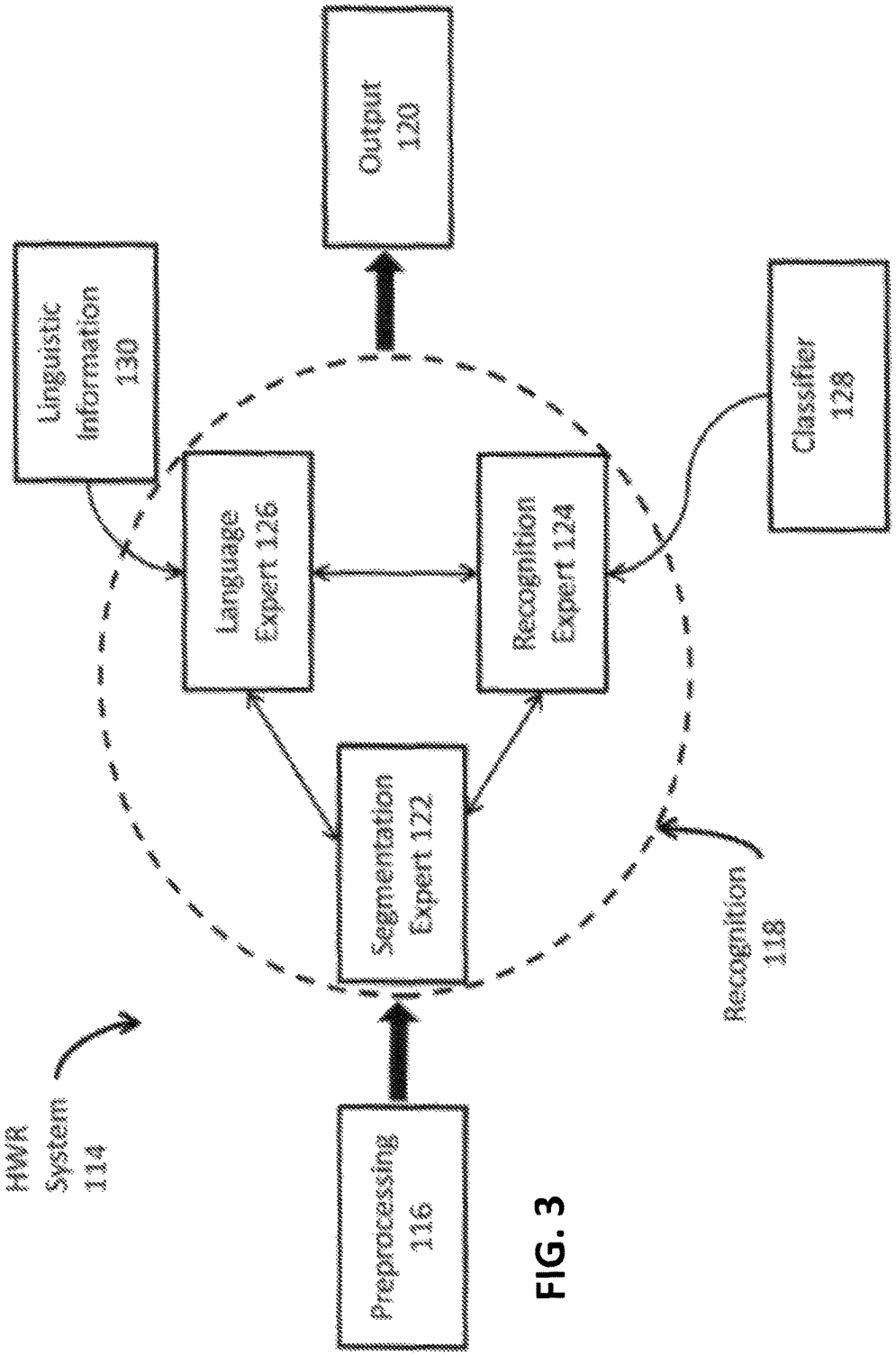

Output
120
Linguistic
Information
130
Classifier
128
Language
Expert 126
Recognition
Expert 124
Segmentation
Expert 122
Recognition
118
HWR
System
114
Preprocessing
116

FIG. 3

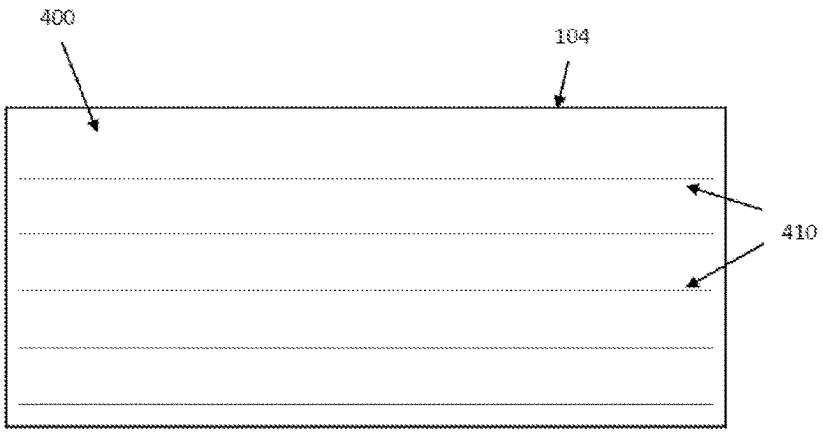

400
104
410

FIG. 4

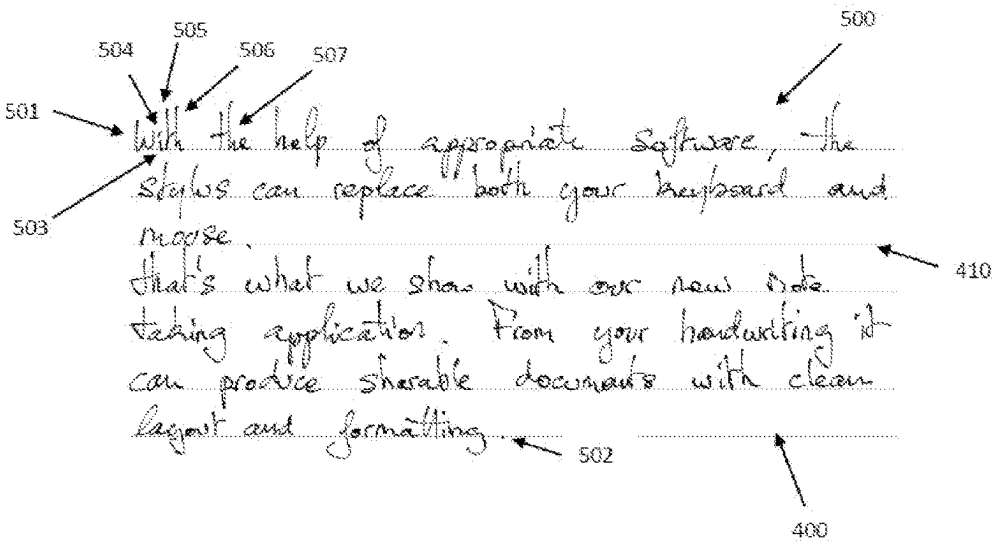

500
505
504
506
507
501
503
With the help of appropriate Software, the
Stylus can replace both your keyboard and
mouse.
That's what we show with our new Note
taking application. From your handwriting it
can produce shareable documents with clean
layout and formatting.
410
502
400

FIG. 5

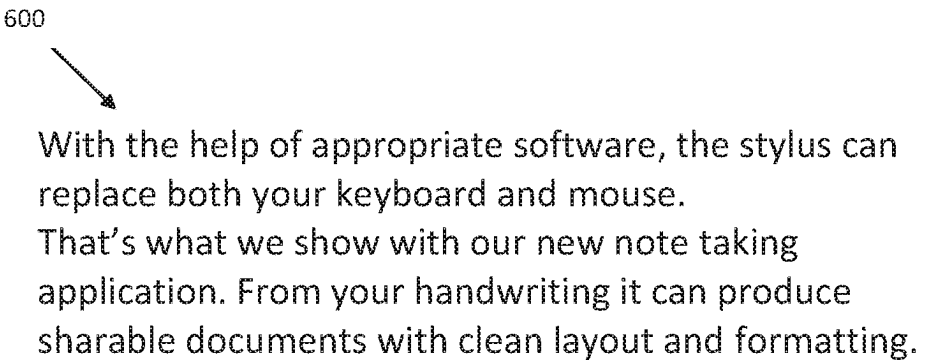

600
With the help of appropriate software, the stylus can replace both your keyboard and mouse.
That's what we show with our new note taking application. From your handwriting it can produce sharable documents with clean layout and formatting.

FIG. 6

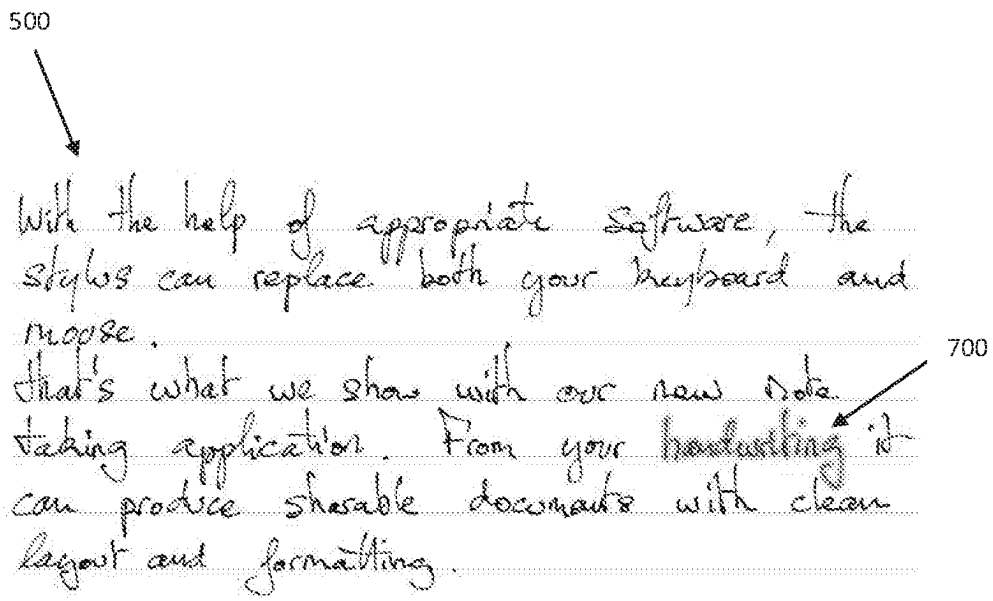

500
With the help of appropriate Software, the stylus can replace both your keyboard and mouse.
That's what we show with our new Note taking application. From your handwriting it can produce sharable documents with clean layout and formatting.
700

FIG. 7

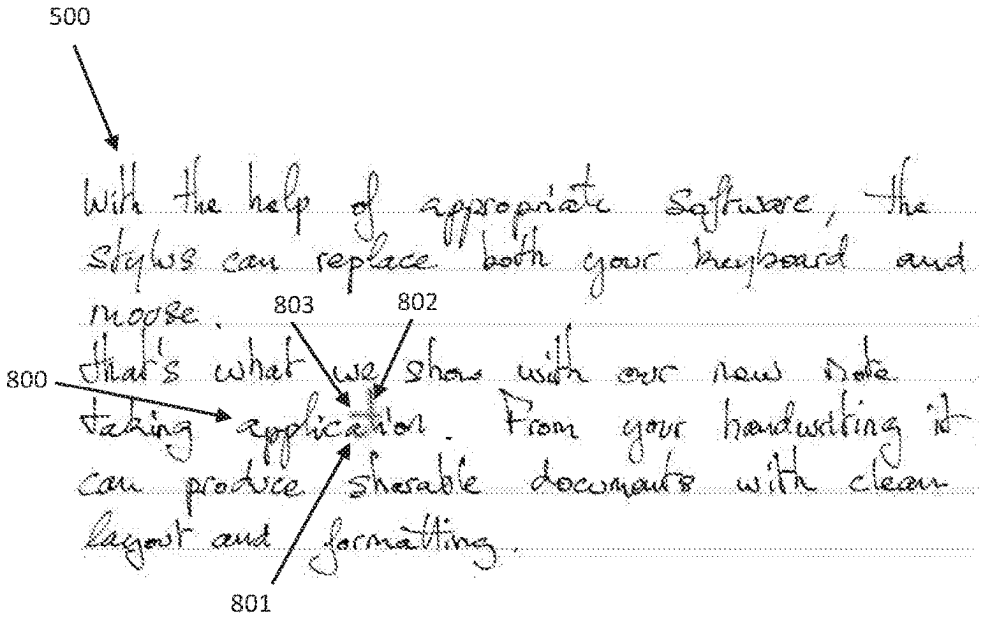

500
With the help of appropriate Software, the
Stylus can replace both your keyboard and
mouse.
803
802
800
that's what we show with our new note
taking application. From your handwriting it
can produce shesable documents with clean
layout and formatting.
801

FIG. 8

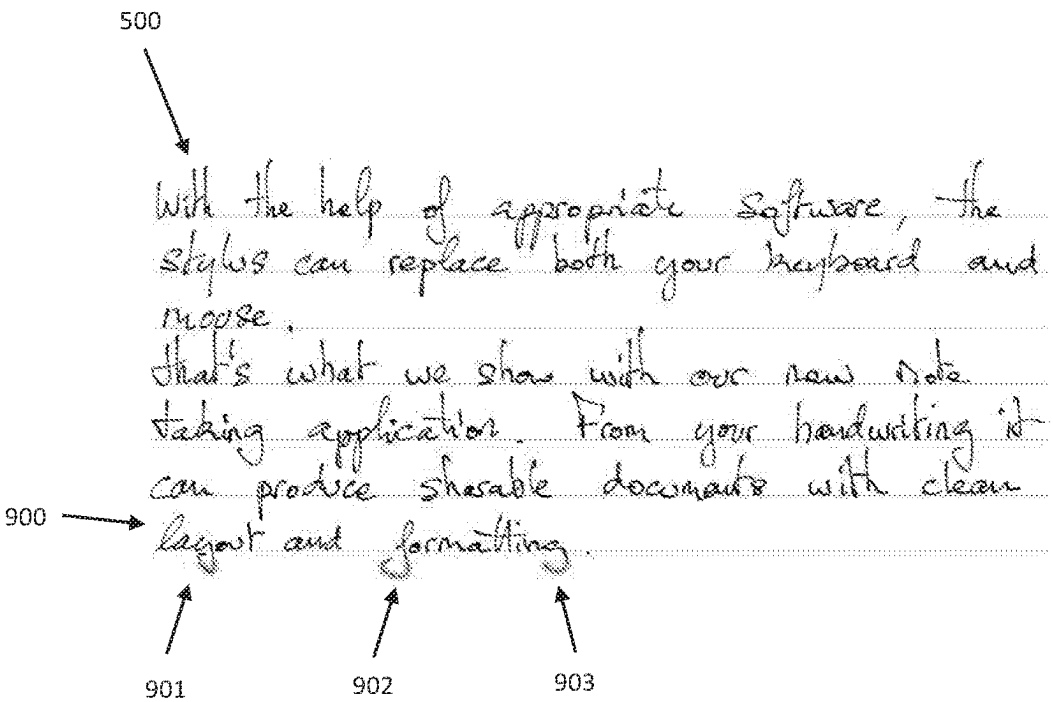

500
With the help of appropriate Software, the
Stylus can replace both your keyboard and
mouse.
that's what we show with our new note
taking application. From your handwriting it
can produce shesable documents with clean
900
layout and formatting.
901
902
903

FIG. 9

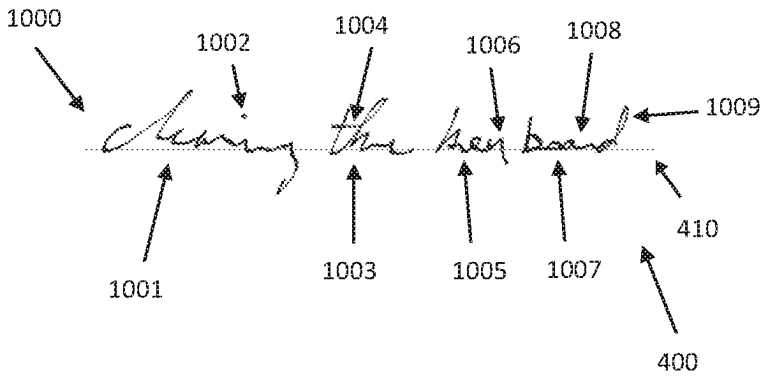


FIG. 10

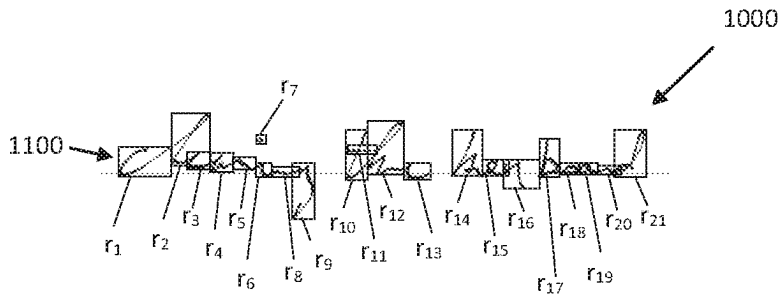


FIG. 11A

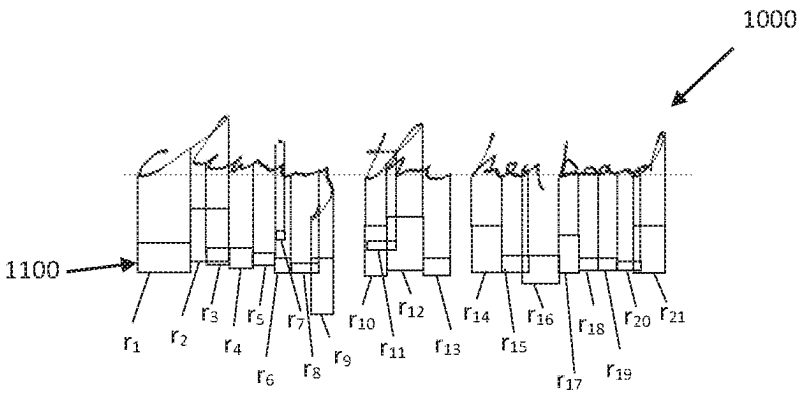


FIG. 11B

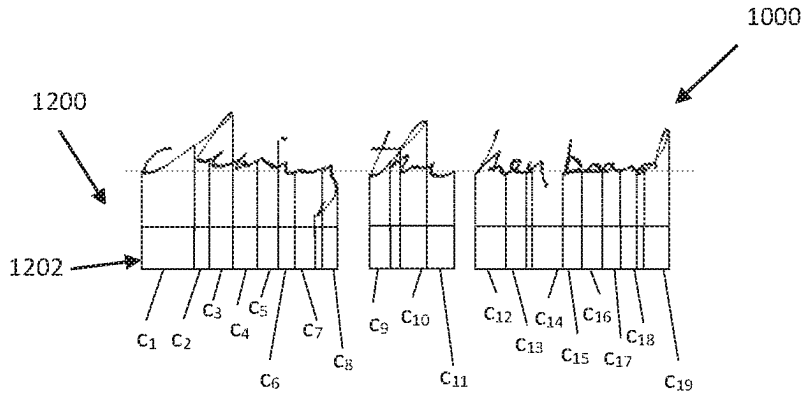


FIG. 12A

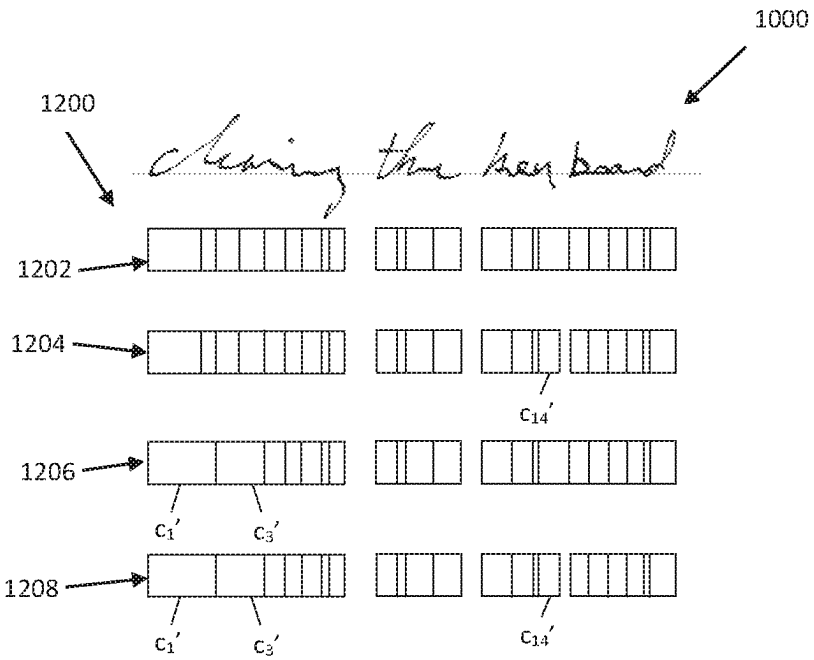


FIG. 12B

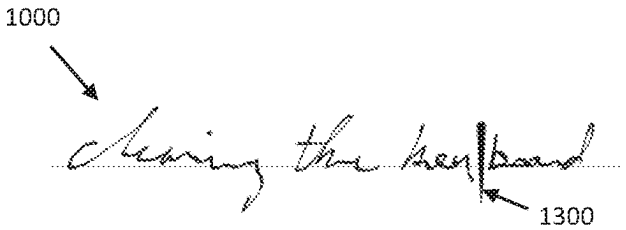


FIG. 13A

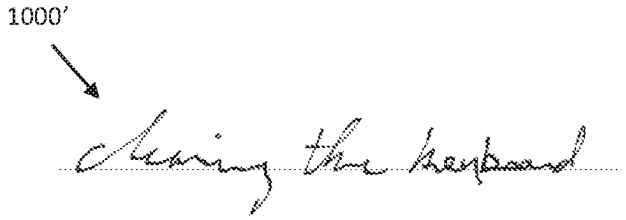


FIG. 13B

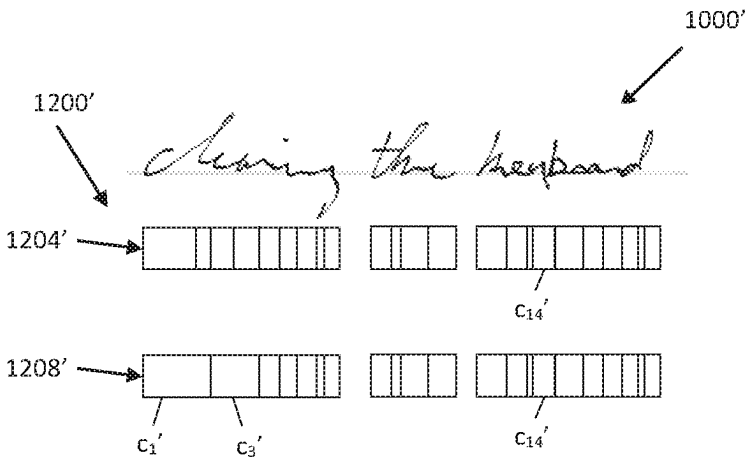


FIG. 13C

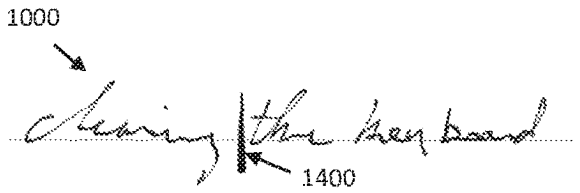


FIG. 14A

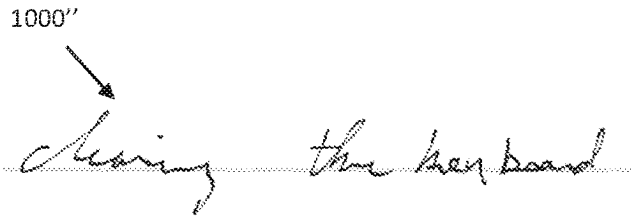


FIG. 14B

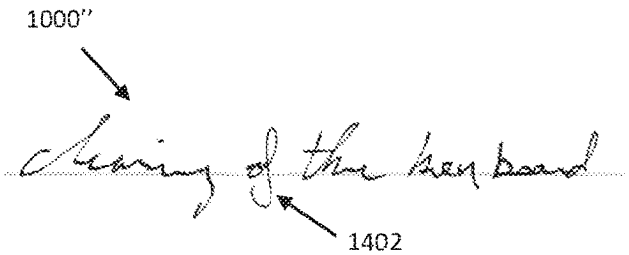


FIG. 14C

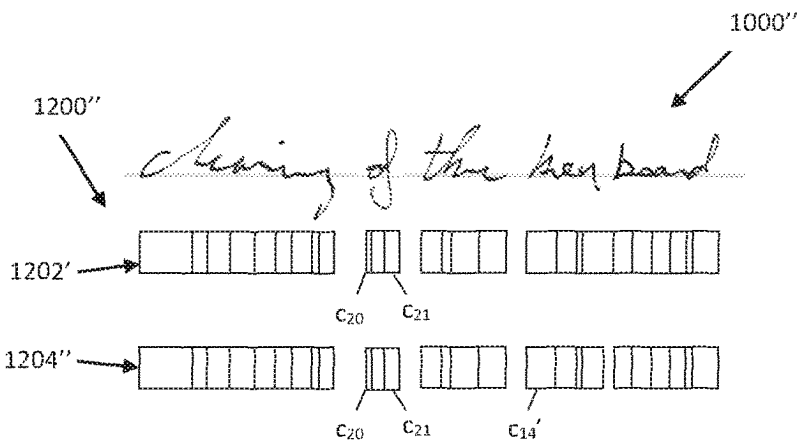


FIG. 14D

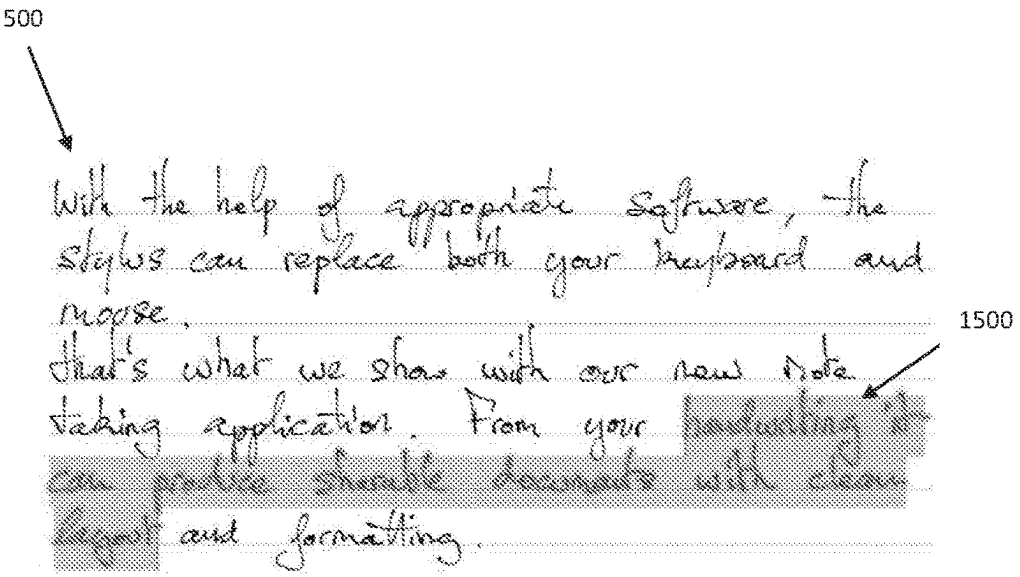


FIG. 15A

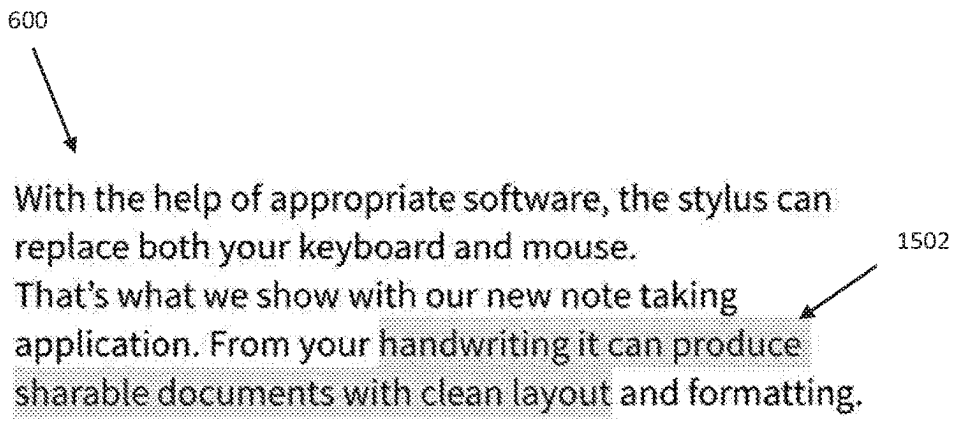


FIG. 15B

SYSTEM AND METHOD FOR DIGITAL INK INTERACTIVITY

CROSS-REFERENCE TO RELATED APPLICATION

This application claims priority to European Application No. 16290002.1 filed on Jan. 7, 2016, the entire contents of which are incorporated by reference herein.

TECHNICAL FIELD

The present description relates generally to the field of ink management systems and methods using computing devices. The present description relates more specifically to managing ink which is handwritten to computing devices interfaces to provide interactivity of the ink.

BACKGROUND

Computing devices continue to become more ubiquitous to daily life. They take the form of computer desktops, laptop computers, tablet computers, hybrid computers (2-in-1s), e-book readers, mobile phones, smartphones, wearable computers (including smartwatches, smart glasses/headsets), global positioning system (GPS) units, enterprise digital assistants (EDAs), personal digital assistants (PDAs), game consoles, and the like. Further, computing devices are being incorporated into vehicles and equipment, such as cars, trucks, farm equipment, manufacturing equipment, building environment control (e.g., lighting, HVAC), and home and commercial appliances.

Computing devices generally consist of at least one processing element, such as a central processing unit (CPU), some form of memory, and input and output devices. The variety of computing devices and their subsequent uses necessitate a variety of interfaces and input devices. One such input device is a touch sensitive surface such as a touch screen or touch pad wherein user input is received through contact between the user's finger or an instrument such as a pen or stylus and the touch sensitive surface. Another input device is an input surface that senses gestures made by a user above the input surface. A further input device is a position detection system which detects the relative position of either touch or non-touch interactions with a non-touch physical or virtual surface. Any of these methods of input can be used generally for drawing or inputting text. The user's handwriting is interpreted using a handwriting recognition system or method.

There are many applications of handwriting recognition in portable computing devices, such as smartphones, phablets and tablets, such as is in note taking, document annotation, mathematical equation input and calculation, music symbol input, sketching and drawing, etc. Handwriting may also be input to non-portable computing devices, particularly with the increasing availability of touchscreen monitors for desktop computers and interactive whiteboards. These types of input are usually performed by the user launching a handwriting input application on the computing device which accepts and interprets, either locally in the device or remotely via a communications link of the device, handwritten input on the touch sensitive surface and displays or otherwise renders this input as so-called 'digital ink'. Conventionally such handwriting input applications are limited in their capabilities to provide a full document creation experience to users from the text and non-text (e.g., drawings, equations), since the focus of these applications has

primarily been recognition accuracy rather than document creation. That is, available applications provide recognition of handwriting and conversion of the recognized handwriting into digital content, rendered as fontified or 'typeset ink', with various feedback mechanisms to the user, but that is generally the extent of the interaction available for the input.

If the user desires any further interaction with the output text, such as editing the content, manipulating the layout of content, or converting or adding the notes or other annotations into a document, the recognized handwriting content generally needs to be imported or otherwise integrated into a separate document processing application. This may be done automatically though the typesetting of the recognized handwriting into suitable typeset ink of suitable format or manually through redundant typing input of the handwriting, for example. The latter manual process is inherently counterproductive and is particularly performed when the perceived accuracy of the handwriting recognition is low, or the ability of the application to preserve the layout of the original handwriting is unreliable. The former automatic process itself does not present a great problem, however as the original layout of the handwriting and the actual input handwriting itself, the digital ink, is typically discarded in the import process, the user must refer back to the original handwriting in order to ascertain the original intent. For example, the user may have emphasized certain words or passages either by annotation or decoration, or through the layout of the content itself.

Some available digital handwriting applications provide the ability to edit the digital ink. However, this is generally done through the input of particular gestures for causing some sort of control, e.g., the launching of menus or running processes. The Applicant has found that when using handwriting applications users generally are unable or do not desire to learn specific gestures that are not natural or intuitive, or to make editing selections through menus and the like. Further, the requirement for particular learned methods to provide digital ink editing limits the usability of such applications, and digital handwriting more generally, as all users must learn the necessary behaviors for digital ink interaction.

Accordingly, in sharing or collaborative uses in which documents are created, edited, revised, reviewed, etc. by multiple parties, in particular parties from different areas or disciplines within enterprise organizations, for example, the ability of different 'team' members to interact with a digital document using handwriting varies such that the productivity possible is limited. For example, a member of upper management in the organization hierarchy may have limited experience with digital handwriting and therefore may not interact with documents using handwriting. Similar problems occur with non-handwriting tools conventionally used with document processing applications such as hardware tools, e.g., keyboard and mouse, and/or software tools, e.g., user interface (UI) tools and menus.

Further, the need to use specific UI and/or hardware tools for editing or other interactions with content created using digital handwriting, may cause disruption to the creative flow of users. As such, the manner of interaction with the content should be easy to use, intuitive and non-disruptive to the input of the content itself.

SUMMARY

The examples of the present invention that are described herein below provide systems and methods for providing interactive ink on computing devices.

In one example, the computing device may include a processor and at least one non-transitory computer readable medium for recognizing handwriting input under control of the processor. The non-transitory computer readable medium may be configured to cause display of, on a display interface of a computing device, first digital ink in accordance with first handwriting input, allocate references to ink elements of the first digital ink, map the references to corresponding recognized elements of the first handwriting input, and determine and store, in the memory of a computing device, ink objects including the references and mapped recognized elements.

The ink elements may be at least part of strokes of the first digital ink and the recognized elements may include at least part of strokes of the first handwriting input. Further, the recognized elements may be recognition candidates of the first handwriting input.

The at least one non-transitory computer readable medium may be further configured to determine content of second handwriting input by referring to the ink objects, and cause display of, on the display interface, second digital ink in accordance with the determined content of the second handwriting input. The second digital ink may be an edited version of at least part of the first digital ink.

In another example, the present invention includes a method for providing interactive ink on computing devices. Each computing device may comprise a processor and at least one non-transitory computer readable medium for recognizing handwriting input under control of the processor. The method includes the steps of displaying, on a display interface of a computing device, first digital ink in accordance with first handwriting input, allocating references to ink elements of the first digital ink, map the references to corresponding recognized elements of the first handwriting input, and determining and storing, in the memory of a computing device, ink objects including the references and mapped recognized elements.

The ink elements may be at least part of strokes of the first digital ink and the recognized elements may include at least part of strokes of the first handwriting input. Further, the recognized elements may be recognition candidates of the first handwriting input.

The method may further include the steps of determining content of second handwriting input by referring to the ink objects, and displaying, on the display interface, second digital ink in accordance with the determined content of the second handwriting input. The second digital ink may be an edited version of at least part of the first digital ink.

In another example, the present invention includes a non-transitory computer readable medium having a computer readable program code embodied therein. The computer readable program code may be adapted to be executed to implement a method for providing interactive ink to a computing device. The computing device may comprise a processor and at least one system non-transitory computer readable medium for recognizing the handwriting input under control of the processor. The method may include the steps of displaying, on a display interface of a computing device, first digital ink in accordance with first handwriting input, allocating references to ink elements of the first digital ink, map the references to corresponding recognized elements of the first handwriting input, and determining and storing, in the memory of a computing device, ink objects including the references and mapped recognized elements.

The ink elements may be at least part of strokes of the first digital ink and the recognized elements may include at least part of strokes of the first handwriting input. Further, the recognized elements may be recognition candidates of the first handwriting input.

The method may further include the steps of determining content of second handwriting input by referring to the ink objects, and displaying, on the display interface, second digital ink in accordance with the determined content of the second handwriting input. The second digital ink may be an edited version of at least part of the first digital ink.

BRIEF DESCRIPTION OF THE DRAWINGS

The present system and method will be more fully understood from the following detailed description of the examples thereof, taken together with the drawings. In the drawings like reference numerals depict like elements. In the drawings:

FIG. **1** shows a block diagram of a computing device in accordance with an example of the present system and method;

FIG. **2** shows a block diagram of a system for handwriting recognition in accordance with an example of the present system and method;

FIG. **3** shows a block diagram illustrating detail of the handwriting recognition system of FIG. **2** in accordance with an example of the present system and method;

FIG. **4** shows a schematic view of an example visual rendering of an ink input area on a portion of an input surface of the computing device;

FIG. **5** shows example handwritten content input to the input area rendered as digital ink;

FIG. **6** shows the result of typesetting the handwritten content of FIG. **5**;

FIG. **7** shows the content of FIG. **5** with an example group of handwritten strokes highlighted;

FIG. **8** shows the content of FIG. **5** with an example group of partial handwritten strokes highlighted;

FIG. **9** shows the content of FIG. **5** with an example group of partial handwritten strokes highlighted;

FIG. **10** shows example handwritten content input to the input area rendered as digital ink;

FIG. **11**A shows the handwritten input of FIG. **10** with a representation of an example reference segmentation;

FIG. **11**B shows the reference segmentation isolated from the digital ink of the handwritten input;

FIG. **12**A shows the handwritten input of FIG. **10** with a representation of an example recognition;

FIG. **12**B shows the handwritten input of FIG. **10** with representations of several example segmentations of the recognition;

FIG. **13**A shows the digital ink of FIG. **10** with an example input editing gesture;

FIG. **13**B shows an example adjustment of the digital ink based on the editing gesture of FIG. **13**A;

FIG. **13**C shows the adjusted digital ink with an example adjusted recognition with representations of adjusted example segmentations;

FIG. **14**A shows the digital ink of FIG. **10** with an example input editing gesture;

FIG. **14**B shows an example adjustment of the digital ink based on the editing gesture of FIG. **14**A;

FIG. **14**C shows the adjusted digital ink with new example handwritten content input to the input area rendered as new digital ink;

FIG. **14**D shows adjusted and new digital ink with an example adjusted recognition with representations of adjusted example segmentations;

FIG. **15**A shows the digital ink of FIG. **5** with an example content portion highlighted as selected; and

FIG. **15**B shows the typeset ink of FIG. **6** with an example content portion highlighted as selected.

DETAILED DESCRIPTION

In the following detailed description, numerous specific details are set forth by way of examples in order to provide a thorough understanding of the relevant teachings. However, it should be apparent to those skilled in the art that the present teachings may be practiced without such details. In other instances, well known methods, procedures, components, and/or circuitry have been described at a relatively high-level, without detail, in order to avoid unnecessarily obscuring aspects of the present teachings.

The use of the term 'text' in the present description is understood as encompassing all alphanumeric characters, and strings thereof, in any written language and common place non-alphanumeric characters, e.g., symbols, used in written text. Further, the term 'non-text' in the present description is understood as encompassing freeform hand-written or hand-drawn content and rendered text and image data, as well as non-alphanumeric characters, and strings thereof, and alphanumeric characters, and strings thereof, which are used in non-text contexts. Furthermore, the examples shown in these drawings are in a left-to-right written language context, and therefore any reference to positions can be adapted for written languages having different directional formats.

The various technologies described herein generally relate to capture, processing and management of hand-drawn and handwritten content on portable and non-portable computing devices in a manner which retains the inputted style of the content while allowing conversion to a faithful typeset or beautified version of that content. The systems and methods described herein may utilize recognition of users' natural writing and drawing styles input to a computing device via an input surface, such as a touch sensitive screen, connected to, or of, the computing device or via an input device, such as a digital pen or mouse, connected to the computing device or via a physical or virtual surface monitored by a position detection system. Whilst the various examples are described with respect to recognition of handwriting input using so-called online recognition techniques, it is understood that application is possible to other forms of input for recognition, such as offline recognition in which images rather than digital ink are recognized. The terms hand-drawing and handwriting are used interchangeably herein to define the creation of digital content by users through use of their hands either directly onto a digital or digitally connected medium or via an input tool, such as a hand-held stylus. The term "hand" is used herein to provide concise description of the input techniques, however the use of other parts of a users' body for similar input is included in this definition, such as foot, mouth and eye.

FIG. **1** shows a block diagram of an example computing device **100**. The computing device may be a computer desktop, laptop computer, tablet computer, hybrid computers (2-in-1s), e-book reader, mobile phone, smartphone, wearable computer, digital watch, interactive whiteboard, global positioning system (GPS) unit, enterprise digital assistant (EDA), personal digital assistant (PDA), game console, or the like. The computing device **100** includes components of at least one processing element, some form of memory and input and/or output (I/O) devices. The components communicate with each other through inputs and outputs, such as connectors, lines, buses, cables, buffers, electromagnetic links, networks, modems, transducers, IR ports, antennas, or others known to those of ordinary skill in the art.

The illustrated example of the computing device **100** has at least one display **102** for outputting data from the computing device such as images, text, and video. The display **102** may use LCD, plasma, LED, iOLED, CRT, or any other appropriate technology that is or is not touch sensitive as known to those of ordinary skill in the art. At least some of the display **102** is co-located with at least one input surface **104**. The input surface **104** may employ technology such as resistive, surface acoustic wave, capacitive, infrared grid, infrared acrylic projection, optical imaging, dispersive signal technology, acoustic pulse recognition, or any other appropriate technology as known to those of ordinary skill in the art to receive user input. The input surface **104** may be bounded by a permanent or video-generated border that clearly identifies its boundaries. Instead of, or additional to, an on-board display, the computing device **100** may have a projected display capability.

The computing device **100** may include one or more additional I/O devices (or peripherals) that are communicatively coupled via a local interface. The additional I/O devices may include input devices such as a keyboard, mouse, scanner, microphone, touchpads, bar code readers, laser readers, radio-frequency device readers, or any other appropriate technology known to those of ordinary skill in the art. Further, the I/O devices may include output devices such as a printer, bar code printers, or any other appropriate technology known to those of ordinary skill in the art. Furthermore, the I/O devices may include communications devices that communicate both inputs and outputs such as a modulator/demodulator (modem; for accessing another device, system, or network), a radio frequency (RF) or other transceiver, a telephonic interface, a bridge, a router, or any other appropriate technology known to those of ordinary skill in the art. The local interface may have additional elements to enable communications, such as controllers, buffers (caches), drivers, repeaters, and receivers, which are omitted for simplicity but known to those of skill in the art. Further, the local interface may include address, control, and/or data connections to enable appropriate communications among the other computer components.

The computing device **100** also includes a processor **106**, which is a hardware device for executing software, particularly software stored in the memory **108**. The processor can be any custom made or commercially available general purpose processor, a central processing unit (CPU), commercially available microprocessors including a semiconductor based microprocessor (in the form of a microchip or chipset), microcontroller, digital signal processor (DSP), application specific integrated circuit (ASIC), field programmable gate array (FPGA) or other programmable logic device, discrete gate or transistor logic, discrete hardware components, state machine, or any combination thereof designed for executing software instructions known to those of ordinary skill in the art.

The memory **108** can include any one or a combination of volatile memory elements (e.g., random access memory (RAM, such as DRAM, SRAM, or SDRAM)) and nonvolatile memory elements (e.g., ROM, EPROM, flash PROM, EEPROM, hard drive, magnetic or optical tape, memory registers, CD-ROM, WORM, DVD, redundant array of inexpensive disks (RAID), another direct access storage device (DASD), or any other magnetic, resistive or phase-change nonvolatile memory). Moreover, the memory **108**

may incorporate electronic, magnetic, optical, and/or other types of storage media. The memory **108** can have a distributed architecture where various components are situated remote from one another but can also be accessed by the processor **106**. Further, the memory **108** may be remote from the device, such as at a server or cloud-based system, which is remotely accessible by the computing device **100**. The memory **108** is coupled to the processor **106**, so the processor **106** can read information from and write information to the memory **108**. In the alternative, the memory **108** may be integral to the processor **106**. In another example, the processor **106** and the memory **108** may both reside in a single ASIC or other integrated circuit.

The software in the memory **108** includes an operating system **110** and an ink management system **112**. The ink management system **112** optionally further includes a handwriting recognition (HWR) system **114** which may each include one or more separate computer programs. Each of these has an ordered listing of executable instructions for implementing logical functions. The operating system **110** controls the execution of the ink manager **112** (and the HWR system **114**). The operating system **110** may be any proprietary operating system or a commercially or freely available operating system, such as WEBOS, WINDOWS®, MAC and IPHONE OS®, LINUX, and ANDROID. It is understood that other operating systems may also be utilized. Alternatively, the ink management system **112** of the present system and method may be provided without use of an operating system.

The ink manager **112** includes one or more processing elements related to detection, management and treatment of user input (discussed in detail later). The software may also include one or more other applications related to handwriting recognition, different functions, or both. Some examples of other applications include a text editor, telephone dialer, contacts directory, instant messaging facility, computer-aided design (CAD) program, email program, word processing program, web browser, and camera. The ink manager **112**, and the other applications, include program(s) provided with the computing device **100** upon manufacture and may further include programs uploaded or downloaded into the computing device **100** after manufacture.

The HWR system **114**, with support and compliance capabilities, may be a source program, executable program (object code), script, application, or any other entity having a set of instructions to be performed. When a source program, the program needs to be translated via a compiler, assembler, interpreter, or the like, which may or may not be included within the memory, so as to operate properly in connection with the operating system. Furthermore, the handwriting recognition system with support and compliance capabilities can be written as (a) an object oriented programming language, which has classes of data and methods; (b) a procedure programming language, which has routines, subroutines, and/or functions, for example but not limited to C, C++, Pascal, Basic, Fortran, Cobol, Perl, Java, Objective C, Swift, and Ada; or (c) functional programming languages for example but no limited to Hope, Rex, Common Lisp, Scheme, Clojure, Racket, Erlang, OCaml, Haskell, Prolog, and F#.

Alternatively, the HWR system **114** may be a method or system for communication with a handwriting recognition system remote from the device, such as server or cloud-based system, but is remotely accessible by the computing device **100** through communications links using the aforementioned communications I/O devices of the computing device **100**. In FIG. **1**, the HWR system **114** is shown as being incorporated in the ink manager **112**, however it is possible the HWR system **114** is separate from and connected to the ink manager **112** either within the computing device **100** or remotely. Further still, the ink manager **112** and/or the HWR system **114** may be integrated within the operating system **110**.

Strokes entered on or via the input surface **104** are processed by the processor **106** as digital ink. Users may enter a stroke with a finger or some instrument such as a pen or stylus suitable for use with the input surface. The user may also enter a stroke by making a gesture above the input surface **104** if technology that senses or images motion in the vicinity of the input surface **104** is being used, or with a peripheral device of the computing device **100**, such as a mouse or joystick, or with a projected interface, e.g., image processing of a passive plane surface to determine the stroke and gesture signals. A stroke is characterized by at least the stroke initiation location, the stroke termination location, and the path connecting the stroke initiation and termination locations. Further information such as timing, pressure, angle at a number of sample points along the path may also be captured to provide deeper detail of the strokes. Because different users may naturally write the same object, e.g., a letter, a shape, a symbol, with slight variations, the HWR system accommodates a variety of ways in which each object may be entered whilst being recognized as the correct or intended object.

FIG. **2** is a schematic pictorial of an example of the HWR system **114**, in either its local (i.e., loaded on the device **100**) or remote (i.e., remotely accessible by the device **100**) forms. The HWR system **114** includes stages such as preprocessing **116**, recognition **118** and output **120**. The preprocessing stage **116** processes the digital ink to achieve greater accuracy and reducing processing time during the recognition stage **118**. This preprocessing may include normalizing of the path connecting the stroke initiation and termination locations by applying size normalization and/or methods such as B-spline approximation to smooth the input. The preprocessed strokes are then passed to the recognition stage **118** which processes the strokes to recognize the objects formed thereby. The recognized objects are then output **120** to the display **102** generally as a typesetted version of the handwritten elements/characters.

The recognition stage **118** may include different processing elements or experts. FIG. **3** is a schematic pictorial of the example of FIG. **2** showing schematic detail of the recognition stage **118**. Three experts, a segmentation expert **122**, a recognition expert **124**, and a language expert **126**, are illustrated which collaborate through dynamic programming to generate the output **120**.

The segmentation expert **122** defines the different ways to segment the input strokes into individual element hypotheses, e.g., alphanumeric characters and mathematical operators, text characters, individual shapes, or sub expression, in order to form expressions, e.g., words, mathematical equations, or groups of shapes. For example, the segmentation expert **122** may form the element hypotheses by grouping consecutive strokes of the original input to obtain a segmentation graph where each node corresponds to at least one element hypothesis and where adjacency constraints between elements are handled by the node connections. Alternatively, the segmentation expert **122** may employ separate experts for different input types, such as text, drawings, equations, and music notation.

The recognition expert **124** provides classification of the features extracted by a classifier **128** and outputs a list of element candidates with probabilities or recognition scores

for each node of the segmentation graph. Many types of classifiers exist that could be used to address this recognition task, e.g., Support Vector Machines, Hidden Markov Models, or Neural Networks such as Multilayer Perceptrons, Deep, Convolutional or Recurrent Neural Networks. The choice depends on the complexity, accuracy, and speed desired for the task.

The language expert **126** generates linguistic meaning for the different paths in the segmentation graph using language models (e.g., grammar or semantics). The expert **126** checks the candidates suggested by the other experts according to linguistic information **130**. The linguistic information **130** can include a lexicon, regular expressions, etc. and is the storage for all static data used by the language expert **126** to execute a language model. A language model can rely on statistical information on a given language. The linguistic information **130** is computed off-line, with or without adaption according to the results of recognition and user interactions, and provided to the linguistic expert **126**. The language expert **126** aims at finding the best recognition path. In one example, the language expert **126** does this by exploring a language model such as finite state automaton (FSA) representing the content of linguistic information **130**. In addition to the lexicon constraint, the language expert **126** may use a language model with statistical information modeling for how frequent a given sequence of elements appears in the specified language or is used by a specific user to evaluate the linguistic likelihood of the interpretation of a given path of the segmentation graph.

The present system and method make use of the HWR system **114** in order to recognize handwritten input to the device **100**. The ink management system **112** includes application(s) for handling the layout of the recognized user input. Such applications may be provided in an architecture with separate layers for handling different processing. One or more of these layers may be remote to the device **100** accessible via the communications channels mentioned earlier. The layers may include application wrapper(s), platform wrapper(s) and (platform specific) application user interface (s).

The ink management system **112** of the present system and method allows users to use natural writing to input content and to interact with the content to produce digital documents in sharable format on their portable or non-portable computing devices using natural and intuitive operations, such as gestures. Any person able to write is familiar with using handwriting to create and edit content. Any digital device user is already accustomed to gesturing on screen to write or edit content. Gesturing is a natural and intuitive pattern on touch and hover devices. These and other features of the present system and method are now described in detail.

FIG. **4** shows a schematic view of an example visual rendering of an ink input or capture area **400** on a portion of the input surface **104** of an example computing device **100**. The input area **400** is provided as a constraint-free canvas that allows users to create object blocks (blocks of text, drawings, etc.) anywhere without worrying about sizing or alignment. However, as can be seen an alignment structure in the form of a line pattern background **410** can be provided for guidance of user input and the alignment of digital and typeset ink objects. In any case, as users may input handwriting that is not closely aligned to the line pattern, or may desire to ignore the line pattern and write in a unconstrained manner, such as diagonally or haphazardly, the recognition of the handwriting input is performed by the HWR system **114** without regard to the line pattern. An example alignment pattern is described in United States Patent Application Publication No. 2017/0060819 titled "System and Method of Digital Note Taking" filed in the name of the present Applicant and Assignee, the entire content of which is incorporated by reference herein.

The input of handwritten content onto the input area **400** is performed through the use of gestures in relation to the input surface **104**, e.g., through touch, force and/or proximity depending on the screen technology of the device **100**. Gesture detection may be handled differently depending on the ability of the computing device **100** to differentiate a users' finger from a stylus or pen (by which the device defines passive and active styli) or the ability of a stylus to indicate or communicate to the device that it is being used for handwriting or the ability of users to provide such an indication. By default, and in devices which do not differentiate, any single-point touch or hover event within the input area **400** is to be considered as content input or content interaction. The present system and method through the HWR system **114** also provides a mechanism to digitize or convert the handwritten input through the typesetting or fontification of the digital ink into typeset ink. This function is particularly provided to allow the creation of at least near-final documents for communication and sharing, e.g., conversion of a handwritten note which has been edited and formatted in digital ink into a typeset document which could have been produced using keyboard (and associated gesturing devices) alone.

The handwritten input is detected by the ink management system **112** and rendered as digital ink on the surface **104** (or other display) as the input is received and is concurrently recognized by the HWR system **114** providing so-called 'on-the-fly' or incremental handwriting recognition. Incremental recognition is generally performed by parsing the (preprocessed) strokes to the recognizer as they are received and the recognizer processing groups of the strokes to output recognition results, even as input continues, where the results may be substantially immediately provided in the form of typesetting of the digital ink or displayed recognition candidates, or merely stored by the ink management system **112** for later use, e.g., by using the memory **108** of the device **100**. Re-recognition may occur upon the input of further strokes that relate to earlier input strokes in a spatial manner, such as diacritics. Alternatively, or additionally, handwriting recognition may be performed as a batch process, rather than incrementally. In such an example, display of the digital ink and the typeset ink may occur during input at specified times, e.g., system and/or user specified, or may occur at a time after input on the same or different device, for example.

FIG. **5** shows one (or two) multi-line paragraph(s) **500** input as handwritten content to the input area **400** generally on the lines of the line pattern **410** and rendered as digital ink on the interface surface of the device, for example. FIG. **6** shows the result of typesetting the paragraph(s) **500** into a typeset paragraph(s) **600**.

The ink management system **112** of the present system and method manages the handwritten input and digital ink representation thereof in a manner which is similar to the management of digital content, such as typed ink, conventionally employed by the operating system **110** and the components and applications thereof. That is, if the content **600** was input to the computing device **100** by typing using keyboard rather than handwritten, the keyboard or typing decoder of the computing device **100**, e.g., provided as an application in the memory **108**, as part of the operation system **110**, or in the present system and method as part of

the ink management system **112**, would interpret and encode each element of the content, being the paragraph **600** itself, the sentences contained in the paragraph, the words and symbols (e.g., grammatical marks, such as periods, commas) contained in the sentences and the characters or letters contained in the words, in the digital content. Similar management also applies if the content **600** was input via other means, such as optical character recognition (OCR) from digital or non-digital (e.g., paper ink) handwriting. This digital content is handled by the computing device **100** in a well understood manner, with each digital character treated as a digital object. In this way, functions such as input or editing cursor placement for interaction with the digital content can be made in relation to each digital object.

However, unlike the decoding process for typed input in which keystrokes on a keyboard are directly interpreted into 'digital objects' (e.g., the typed letter "a" as depicted in this text is a digital object), the recognition process for handwriting input generally takes a probabilistic approach to interpreting hand-drawn strokes on or to the interface surface to find the most probable candidates for characters. Conventional digital handwriting systems and methods using such probabilities generally provide the highest probability character and word candidates as recognized content, such that upon typesetting the recognized content is converted to the digital content at which time the recognized content, and as such the recognition result, is discarded. Any recognition errors can then be corrected through interaction with the digital content using the conventional digital content management techniques described above. The number of such errors carried through with the typeset conversion can be minimized through certain feedback mechanisms during input, such as display of candidate lists, auto-correction based on the language model(s), or use of substantially real-time recognition result display, such as described in United States Patent Application Publication No. 2017/0060406 titled "System and Method of Guiding Handwriting Input" filed in the name of the present Applicant and Assignee, the entire content of which is incorporated by reference herein.

Digital content management only takes account of the digital objects through reference to the metadata of the digital objects. This metadata provides information on the object type, font, styling, relative position, embedded data or hyperlinks, etc. of the digital content, but does not provide any further information such as the word to which each character belongs, the context of the character, object candidates. Accordingly, any subsequent interaction with the digital objects is made independent of the proximate digital objects. The handwriting recognition process however provides a vast amount of information with respect to each recognition object, e.g., each recognized character, which can be used to guide and augment subsequent interaction with the content. The present system and method provide ways to leverage this recognition information to provide levels of interaction with handwritten and digital content which enhance creativity, usability and productivity for content creation, editing, sharing and collaboration, as described below.

The present system and method references or links the 'raw' or input ink (e.g., the handwritten strokes) to the digital ink (e.g., the displayed ink). By this link, the relationship between the actual input, as recognized by the HWR system **114**, and the displayed input is known by the system, such that user interaction with the digital ink, for example, to edit the content, is performed in relation to the underlying recognition. This 'marrying' of the recognized ink and digital ink forms 'ink objects'. Each ink object has metadata including information similar to that of the digital objects but also further information related to the recognition processing. An example reference system and method is now described with reference to FIGS. **5** and **7** to **12**.

In FIG. **5**, the paragraph(s) **500** is the digital ink representation of the underlying raw ink input of a paragraph(s) of handwritten text. The raw ink includes a number of ink strokes written in temporal and spatial order. In the context of the left-to-right and top-to-bottom language depicted in the example of FIG. **5**, the first stroke is stroke **501** (represented as the digital ink character 'W') at the beginning of the paragraph **500** and the final stroke is stroke **502** (represented as the digital ink character ".") at the end of the paragraph **500**. As the ink input is received, the present system and method sequentially indexes each element of the input, such as the strokes characterized by the ink between each detected subsequent stroke initiation (or 'pen down') and termination (or 'pen up') locations, in time order, by allocating a reference to each element and its features, for example.

For example, for the paragraph **500**, the stroke **501** may be indexed as stroke [0] (zero) and the stroke **502** may be indexed as stroke [244]. The indexation in time order means that strokes that are positionally out of order, such as diacritics, strokes added later, etc., have later indexing than proximate strokes, e.g., stroke **503** (represented as the digital ink character "i" without the dot ".") may be indexed as stroke [1] (one) and stroke **504** (represented as the dot ".") may be indexed as stroke [4] as it is written after strokes **505** and **506**, for example. Alternatively or additionally, the strokes may be re-indexed to take account of spatial order rather than time order. Further, the indexing may occur in either or both fashions for bulk input, such as OCR input.

This stroke index provides a coarse indexation of the input which can be used to define features of the input, such as character and word boundaries. For example, user interaction with the digital ink character **501**, represented as stroke index [0], for example, with a gesture on the digital ink, such as a single or multi-point gesture (e.g., a tap, long press or double tap), may cause selection of the character and/or provision of an input cursor at the character boundary, similar to interaction with digital objects. However, unlike digital objects, due to the stroke index, this interaction also references the underlying recognition result(s), "character[0]", which provides features as discussed in more detail later. Briefly here however, the indexing to the stroke index, e.g., index [0], remains stable even if the stroke is transformed, e.g., partially erased, erased, omitted through an 'undo' action, fontified or typeset, and any new strokes for this transformation are allocated or assigned new stroke indices which still internally reference the previous stroke index, e.g., index [0], thereby retaining the link to the recognition results. Further, in FIG. **7** a group of strokes **700** (represented as the digital ink word "handwriting") of the paragraph **500** (highlighted in the display of FIG. **7** for illustration purposes) includes the stroke indexes of stroke [172] to stroke [183], represented as stroke index [172, 183], for example. User interaction with the digital ink word **700** with a gesture on the digital ink, such as a single or multi-point gesture (e.g., a tap, long press or double tap), anywhere within or near the word **700** may cause selection of the entire word and/or provision of a cursor at the word boundary, similar to interaction with digital objects. However, unlike digital objects, due to the stroke index, this

interaction also relates to the underlying recognition result (s), "word[172, 183]", which provides features as discussed in more detail later.

As discussed earlier, in the recognition process the strokes are segmented in order to determine probable character candidates. A similar (or the same or actual) segmentation approach is taken by the present system and method to refine the indexation of the input. This segmentation involves segmenting each indexed stroke into a number of stroke segments and indexing each stroke segment at the point between each stroke segment. For example, the stroke **501** may be indexed with a series of 50 stroke segments from point 0 (zero) to point 50, say, represented as stroke and point ("reference") index [0:0, 0:50], for example. The position of each point may be set by segmenting the strokes in a regular manner, e.g., each stroke segment has the same length (such as number of pixels with respect to the digital ink representation), in accordance with the stroke characteristics, or in accordance with the device sampling and any possible pre-sampling applied by the system.

In the latter case, the raw input samples (as given by the computing or capturing device, as stored in the memory **108** for example) of each input event, such as pen down, pen move, direction change, pen up, are used by the ink management system **112** as a new point or sample in the input stroke(s). Further, instead of the raw input samples, smoothing units could be used by the ink management system **112** in which the raw input samples are processed and output as a different number and position of samples. In this case, the segmentation regime or segment length itself is immaterial to the indexing applied, rather it is the 'smoothness' of the digital ink as segmented which provides optimized indexing. It is noted however that such smoothing of the input may invalidate biometric processing of the digital ink, such as for signature authentication. Further, smoothing suitable for display may be different from smoothing suitable for recognition. As such, smoothing may be performed by the ink management system **112** itself at different stages than smoothing at input and rather than by any processing of the device **100** itself.

In any case, the number of stroke segments set the number of points in each index. As discussed above, user interaction with the digital ink character **501** with a gesture on the digital ink may cause selection of the character and/or provision of a digital cursor at the character boundary, such that the cursor is provided at the reference or stroke:point index [0:50]. Accordingly, the cursor position is defined by i:j, where 'i' is the stroke index and 'j' is the point index. Due to the point index, this interaction also relates to the underlying recognition result(s), "character[0:0, 0:50]", which provides features as discussed in more detail later.

The point index provides a fine indexation of the input which can be used to define further features of the input, such as character boundaries within strokes. As can be seen, each stroke of the raw (and digital) ink may belong to more than one text character (e.g., stroke **507** is represented as the digital ink characters "h" and "e") or be a part of one character (e.g., the strokes **503** and **504** combine to form the character "i"). Similar stroke attribution can also occur with non-text objects, such as shapes. As discussed above, the segmentation process of the recognition system segments each stroke into many segmentation possibilities represented by the segmentation graph. By this process, the characters "h" and "e" of the stroke **507** are correctly recognized, for example. The fine indexation of the present system and method provides a similar segmentation result. For example, the stroke **507** may be indexed as stroke [9] and with a series of 60 stroke segments from point 0 (zero) to point 60, say, represented as reference index [9:0, 9:60], for example. The character boundary between the characters "h" and "e" may, for example, be at point 30, such that the character "h" has reference index [9:0, 9:30] and the character "e" has reference index [9:30, 9:60]. Alternatively, contiguous characters may not share indexing, such that index [9:30] is not shared, for example. Accordingly, user interaction with the digital ink characters **507** with a gesture on the digital ink at or near the character boundary may cause provision of a digital cursor at the character boundary, such that the cursor is provided at the reference index [9:30]. Due to the stroke and point index, this interaction also relates to the underlying recognition result(s), "characters[9:0, 9:60]", which provides features as discussed in more detail later.

The fine indexation of the point index can be used to define further features of the input, such as partial strokes. For example, in FIG. **8** a group of strokes **800** (represented as the digital ink word "application") of the paragraph **500** has stroke **801** (represented as the digital ink character "a"), stroke **802** (represented as the digital ink characters "t" and "i" without the bar "-" and dot "·", respectively) and stroke **803** (represented as the bar "-"). The stroke **801** may be indexed as stroke [158] and with points 0 (zero) to 50, say. The stroke **802** may be indexed as stroke [159] and with points 0 (zero) to 40, say. The stroke **803** may be indexed as stroke [161] and with points 0 (zero) to 14, say (e.g., the stroke represented as the following digital ink characters "o" and "n" is input before the stroke **802** and therefore provide stroke [160]). User interaction with the digital ink word **800** with a gesture on the digital ink, such as a single or multi-point gesture (e.g., a tap, long press or double tap), at parts of the strokes of the digital ink may cause selection of only those parts of the strokes. For example, in FIG. **8** interaction (highlighted in the display of FIG. **8** for illustration purposes) with parts of the strokes **801** and **802** includes the reference index [158:44, 159:20], and with all of the stroke **803** includes the reference index [161:0, 161:14], for example. Due to the stroke and point indexes, this selection also relates to the underlying recognition result(s) of the partial characters, which provides features as discussed in more detail later.

Further, in FIG. **9** a group of strokes **900** (represented as the digital ink words "layout and formatting") of the paragraph **500** has stroke **901** (represented as the digital ink character "y" in the word "layout"), stroke **902** (represented as the digital ink character "f" in the word "formatting") and stroke **903** (represented as the digital ink characters "n" and "g" in the word "formatting"). The stroke **901** may be indexed as stroke [215] and with points 0 (zero) to 60, say. The stroke **902** may be indexed as stroke [233] and with points 0 (zero) to 60, say. The stroke **903** may be indexed as stroke [243] and with points 0 (zero) to 54, say. User interaction with the digital ink **900** with a gesture on the digital ink, such as a single or multi-point gesture (e.g., a press and swipe), at parts of the strokes of the digital ink may cause selection of only those parts of the strokes. For example, in FIG. **9** interaction (highlighted in the display of FIG. **9** for illustration purposes) with part of the stroke **901** (the descender of "y") includes the reference index [215:36, 215:48], with part of the stroke **902** (the underwrite or descender of "f") includes the reference index [233:35, 233:54] and with part of the stroke **903** (the descender of "g") includes the reference index [243:37, 243:54], for example. Due to the stroke and point indexes, this selection

also relates to the underlying recognition result(s) of the partial characters, which provides features as discussed in more detail later.

Further still, increased precision for such user interaction can be provided by further segmenting between the points of the point index. For example, regular or stroke defined sub-points can be defined between each pair of consecutive points to provide a decimal part of the point index. The fractional precision can be, for example, about $\frac{1}{50}^{th}$ to about $\frac{1}{300}^{th}$ of each stroke segment. However, less or more precision can be provided. For example, in FIG. **9**, $\frac{1}{200}^{th}$ sub-points are provided such that the displayed interaction with the part of the stroke **901** includes the reference index [215:36.665, 215:48.265], with the part of the stroke **902** includes the reference index [233:35.585, 233:54.71] and with the part of the stroke **903** includes the reference index [243:37.495, 243:54], for example. Due to the stroke and decimal-point indexes, this selection also relates to the underlying recognition result(s) of the partial characters with greater precision, which provides features as discussed in more detail later.

The above described reference indexing system employed by the present system and method provides a reference between the displayed digital ink and the recognition result (s) from the raw ink. This reference once determined by the ink management system **112** may be stored, for example, in the memory **108** of the computing device **100**, for later use upon detection of user interaction with the digital ink, for example. As described above, the reference index is included as metadata with the digital content represented by the raw, recognized and digital ink as ink objects. Accordingly, the ink objects are transportable, so that the digital content can be exported, imported and otherwise transferred or transformed between different applications and operating systems. That is, like any other digital content, the reference metadata is maintained with the ink object content itself. Because the reference index remains with the digital ink it is not necessary that the raw ink itself, e.g., the original handwritten input, be preserved, thereby reducing the amount of data to be stored and increasing the portability of the ink management system **112**, as it is not necessary for a database of the raw ink to be addressed when interpreting ink object interaction. Full capability of the reference system is retained however even if the raw ink itself is not retained. This and the above-discussed features are now described in detail.

FIG. **10** shows a phrase **1000** containing several words input as handwritten content to the input area **400** generally on the lines of the line pattern **410** and rendered as digital ink on the interface surface of the device, for example. The raw (and digital) ink **1000** includes a number of strokes **1001** to **1009** representing text characters which form the phrase "cleaning the keyboard". As discussed earlier, the ink management system **112** segments the strokes to determine the reference index of the input, and therefore the digital ink. FIG. **11**A shows the handwritten input **1000** with a representation of an example reference segmentation **1100** of the phrase "cleaning the keyboard" as determined by the ink management system **112** and/or the HWR system **114**. The reference segmentation **1100** is illustrated as a series of boxes r_m designating the general digital ink bounds or extent of each reference indexed portion of the digital ink **1000**. FIG. **11**B shows these boxes r_m isolated from the digital ink **1000** with the stroke index points shown by dashed lines.

The stroke index may be for example: the stroke **1001** as stroke [0] (zero), the stroke **1002** as stroke [1], the stroke **1003** as stroke [2], the stroke **1004** as stroke [3], the stroke **1005** as stroke [4], the stroke **1006** as stroke [5], the stroke **1007** as stroke [6], the stroke **1008** as stroke [7], and the stroke **1009** as stroke [8]. For example, the stroke **1001** is long and may therefore be segmented to have 300 points, say, such that reference index r_1 corresponds to [0:0, 0:35], reference index r_2 corresponds to [0:35, 0:90], reference index r_3 corresponds to [0:90, 0:125], reference index r_4 corresponds to [0:125, 0:170], reference index r_5 corresponds to [0:170, 0:210], reference index r_6 corresponds to [0:210, 0:225], reference index r_8 corresponds to [0:225, 0:265], and reference index r_9 corresponds to [0:265, 0:300]. As described above, extra precision can be provided by also referring to sub-points. Similar reference indexes are determined for the other strokes **1002** to **1009**.

As discussed earlier, in the recognition process the strokes **1001** to **1009** are segmented in order to determine probable character candidates which lead to the recognition of the words of the phrase **1000**. The ink management system **112** utilizes the reference segmentation **1102** which may be independent of the segmentation produced by the HWR system **114** or the reference segmentation **1102** may be that provided by the HWR system **114** (or at least produced using a similar process). In either case, the reference segmentation of the strokes of the input/digital ink provides a map to the underlying recognition results as follows, for example. FIG. **12**A shows the handwritten input **1000** with a representation of an example recognition **1200** of the phrase "cleaning the keyboard". It is noted that the example recognition is not itself displayed as depicted. As can be seen, the recognition **1200** includes a segmentation **1202** of the strokes **1001** to **1009** of the input **1000** (shown by dashed lines) to produce a series of recognized characters C_n. The recognized characters C_n include "c" C_1, "l" C_2, "e" C_3, "a" C_4, "n" C_5, "i" C_6 including the dot ".", "n" C_7, "g" C_8, "t" C_9 including the bar "-", "h" C_{10}, "e" C_{11}, "k" C_{12}, "e" C_{13}, "y" C_{14}, "b" C_{15}, "o" C_{16}, "a" C_{17}, "r" C_{18}, and "d" C_{19}. It is noted that in the recognition processing, the "n" C_5 may also be recognized as an "r" to provide the word candidate "clearing", instead of "cleaning" with similar or different probabilities based on the language model of the HWR system **114**. Other alternative character recognitions are also possible. Further, the numbering order of the candidates is only for descriptive purposes and does not necessarily reflect the manner in which the HWR system **114** identifies recognized characters, if this is done. Further still, the use of boxes to designate the recognized characters in the drawings is for illustrative purposes only.

As can be seen the recognition segmentation **1202** corresponds generally with the reference segmentation **1102**. Accordingly, the ink management system **112** maps the reference indexes r_m to the candidate characters C_n. For example, the stroke **1001** contains several characters, namely characters C_1 to C_8, such that character C_1 corresponds to reference index r_1, character C_2 corresponds to reference index r_2, character C_3 corresponds to reference index r_3, character C_4 corresponds to reference index r_4, character C_5 corresponds to reference index r_5, character C_6 corresponds to reference indexes r_6 and r_7 combined (e.g., the complete character "i"), character C_7 corresponds to reference index r_8, and character C_8 corresponds to reference index r_9. Similar mapping of the candidate characters and reference indexes are determined for the other strokes **1002** to **1009**. Accordingly, the underlying recognition **1202** of the digital ink **1000** may be determined via the reference indexing as word[0:0, 1:10] corresponding to "cleaning", word [2:0, 3:10] corresponding to "the", and word[4:0, 8:40] corresponding to "keyboard", for example. This reference or

link between the recognized elements, such as characters and words, and the corresponding digital ink elements provides the ink objects, where a first ink object corresponds to the reference index [0:0, 1:10], a second ink object corresponds to the reference index [2:0, 3:10] and a third ink object corresponds to the reference index [4:0, 8:40], in this present example.

In this way, the ink management system **112** links or maps the digital ink strokes with the underlying recognition result provided by the handwriting recognition process. This mapping is provided regardless of the manner in which the digital ink is rendered. That is, the digital ink, being a displayed version of the raw ink input, may be rendered with a different size, style, font, etc. than the original handwriting through transposition or so-called 'beautification'. This is particularly the case if the handwriting is input in a different area than the display itself, in a restricted area, or in a relatively free manner without respect to layout elements, such as the line pattern, in which cases the digital ink is rendered relatively independently of the input raw ink. The mapping of such transposed or beautified digital ink to the recognized ink is possible however so long as the transposition or beautification characteristics are known, such as described in, for example, U.S. patent application Ser. No. 14/978,081 titled "System and Method for Beautifying Digital Ink" filed in the name of the present Applicant and Assignee, the entire content of which is incorporated by reference herein.

The digital to recognized ink map supports user interactivity with the digital ink, and therefore the ink objects, as interactions with the digital ink (for example, in a manner as described earlier), are referred to interactions with the underlying recognized ink through this link, and therefore editing of the digital ink, such as changes to existing content, input of new content, are managed by the ink management system **112** with recourse to existing and new recognition results. This is now described in detail.

In the probabilistic approach to handwriting recognition taken by the HWR system **114**, many potential object candidates, such as character candidates for text input, shape candidates for shape input, notation candidates for music input, are tested via the segmentation graph and the most probable candidates are returned to provide the recognition result. However, upon outputting the recognition result, the HWR system **114** does not discard (all) the other candidates or at least does not discard the top candidates, such as the **5**, **10**, etc. most likely candidates, for example. This is, at least in part, because the recognition result, while the most likely based on the various experts and models employed by the HWR system **114** at the time, may not be correct and users may therefore be provided with the other candidates for selection, via interactive UI menus or the like, in order to correct the recognition result, for example, such as described in the afore-incorporated by reference United States Patent Application Publication No. 2017/0060406. Each one of these character candidates maps onto the digital ink in a similar manner as illustrated in the example of FIG. **12**A. That is, the candidates correspond to certain segmentations of the raw ink, and therefore also of the digital ink. FIG. **12**B shows the handwritten input **1000** with representations of several segmentations **1204**, **1206** and **1208** of the example recognition **1200** of the phrase **1000**.

The segmentation **1204** of the strokes **1001** to **1009** is similar to the segmentation **1202** however in this segmentation the stroke **1006** is segmented to treat the apparent space between the strokes **1006** and **1007** as a space, such that the character candidate C_{14} corresponding to "y" in the segmentation **1202** is not present and instead character candidate C_{14}' results corresponding to "n" for the candidate recognition "cleaning the ken board" (or "clearing the ken board as discussed earlier). It is noted that in the recognition processing, the "n" C_{14}' may also be recognized as a "y" to provide the word candidate "key", instead of "ken" with similar or different probabilities based on the language model of the HWR system **114**.

The segmentation **1206** of the strokes **1001** to **1009** is similar to the segmentation **1102** however in this segmentation the stroke **1001** is segmented such that the character candidates C_1, C_2, C_3 and C_4 corresponding to "c", "l", "e" and "a", respectively, in the segmentation **1202** are not present and instead character candidates C_1' and C_3' result corresponding to "d" and "u", respectively, for the candidate recognition "during the keyboard".

The segmentation **1208** of the strokes **1001** to **1009** is a combination of the segmentations **1204** and **1206** to produce the candidate recognition "during the ken board" (or "during the key board" as described above). These different segmentations may correspond to the most likely recognition candidates, e.g., those returning the highest probability scores. While four most likely recognition candidates are shown, it is understood that more or less may be provided by the present system and method. Further, the order of the candidates **1202** to **1208** is only illustrative and does not imply the order of candidate likelihood. Examples of the manner in which this mapping of candidate recognitions to the digital ink supports interactivity is now described.

FIG. **13**A shows the digital ink **1000** with an editing gesture **1300** detected by the ink management system **112** and/or the HWR system **114** as input as a bottom-to-top vertical line in a single stroke by a user swiping their finger or stylus tip from the bottom to the top in the displayed position shown, for example. That is, the stroke start position is below the stroke end position, in time order, as depicted by the gradual thickening of the stroke from the bottom to the top. The illustrated characteristic of the editing stroke is an example, and other characteristics are possible, such as scratch-out, as are other forms of editing commands (described in detail later). For example, as with digital objects, editing of the ink objects can also be achieved through other actions and tools, such as keyboard, UI buttons or menus, combined with a cursor or selection of ink as editable content. The editing stroke **1300** is detected as passing between the strokes **1006** and **1007**, and as such the ink management system **112** knows the relative location of the editing stroke **1300** not only from the digital ink **1000** but also from each of the candidate recognitions, e.g., **1202** to **1208**, of the underlying recognized ink **1200**. Consideration of this relative location of the editing gesture **1300** and its bottom-to-top writing direction characteristic may result in a determination that the editing gesture **1300** is intended by the user to delete a space at the detected location, particularly with respect to the recognition candidates **1202** and **1208** (described in detail below). With this determination, the ink management system **112** causes the digital ink **1000** to be re-rendered as edited digital ink **1000**' as shown in FIG. **13**B, in which the strokes **1006** and **1007** are joined together. This editing of the digital ink is accompanied by any suitable further recognition of the digital content represented by the ink objects, as described below.

In the determination or interpretation of the editing stroke **1300**, the ink management system **112** refers to the recognition candidates referenced to the strokes in the proximity of the editing stroke location. The range of this proximity may be pre-set and/or user settable (e.g., via a UI menu) and

should generally take into account processing burden and timing, which may affect user experience. For example, the location of the editing stroke **1300** between the strokes **1006** and **1007** substantially coincides with the space candidate identified in the recognition candidates **1204** and **1208** between the alternative character candidate C_{14}' and the character candidate C_{15}. Accordingly, the ink management system **112** determines, or causes the HWR system **114** to determine (e.g., through provision of a suitable probability weighting), that the editing gesture **1300** is a delete space gesture. This determination may at least in part be performed, for example, with the gesture detection and recognition described in U.S. patent application Ser. No. 14/989,217 titled "System and Method for Note Taking with Gestures" filed in the name of the present Applicant and Assignee, the entire content of which is incorporated by reference herein.

Similar delete actions can be taken on the content itself, such as on the text characters, words, sentences, paragraphs, etc., and non-text elements. Further, due to the fine reference indexing (particularly the precision indexing using stroke sub-points) partial deletion of characters, etc., is possible since only those parts of the digital ink will be omitted and the recognition adjusted accordingly to revise the ink objects. For example, the gesture which selected the descenders of the characters in the strokes **901**, **902** and **903** of the paragraph **500** in FIG. **9** may be a deletion gesture for these element components, such that only the digital ink portions corresponding to the reference indexes of these parts are omitted. Alternatively or additionally, partial deletions which are actually intended as full deletions (e.g., a scratch-out or strike-through gesture made by a user only partially selected or misses one or more characters at the end of word, or one or more words at the end of a sentence), can be properly handled by the ink management system **112** to output full deletions as the partial character reference indexes belong to full character reference indexes, character reference indexes belong to word reference indexes, word reference indexes belong to sentence reference indexes, and so on, such that the digital ink of the greater reference indexes are deleted.

The ink management system **112** may be configured to retain the digital ink of any deleted reference index ink, either permanently or for a certain amount of time or number of subsequent operations, for example, by storing the digital ink in the memory **108** of the device **100**, so that deletions can be undone, through UI selection or the like. Alternatively or additionally, only the deleted reference indexes may be retained along with the underlying recognition results (e.g., the partial or entire ink objects) as metadata in the ink objects and/or the memory **108** so that upon an undo action, the restored digital ink can be generated based on the recognition results and appropriate handwriting like fonts, such as personal user fonts produced by sampling a user's handwriting. Accordingly, the ink objects of the present system and method are managed similar to digital objects with similar editing functionality but also provide enhanced functionality as the interactions with the ink objects are augmented by the underlying handwriting recognition and allow interaction with partial content elements, which cannot be performed with digital objects.

Returning to the example of FIG. **13**, the delete space determination causes a change in the recognition candidates. For example, the recognition candidates **1202** and **1206** may be omitted from the ink objects or downgraded from the most likely recognition candidates list through provision of a suitable probability weighting by the ink management system **112**, and the recognition candidates **1204** and **1208** are revised and may be elevated or promoted in the most likely recognition candidates list through provision of a suitable probability weighting by the ink management system **112**. FIG. **13**C shows an example revised or edited recognition **1200**' having the representations of revised segmentations **1204**' and **1208**' of the edited phrase **1000**'. In the revised segmentations **1204**' and **1208**' the recognition processing may further cause the character candidate C_{14}' to be recognized as a "y" omitting or downgrading the "n" candidate, since the joined strokes **1006** and **1007** now likely form the word "keyboard" which based on the language model the HWR system **114** may determine as more likely. This increased likelihood may also be used by the ink management system **112** in the determination of the editing gesture **1300**, as may the entire original and edited phrases **1000** and **1000**'.

As described earlier, the range of proximate strokes to an editing stroke or action takes into account the efficiency of any further recognition processing necessary. The allowed range is relatively generous however since the efficiency is enhanced by the ink objects due to already or existing recognized elements of the input/digital ink not needing to be re-recognized. That is, as described above, only new input is recognized and previous recognition result(s) are retained, revised or omitted in accordance with the referencing made thereto. It is understood however that in some cases previous recognition result(s) may not assist the recognition of the new input, or may require re-recognition together with the new input, such that re-recognition may occur albeit limited to that necessary for recognition of the new input.

The new 'input' of the example of FIG. **13** represented an editing or control action on the content, and not content itself. The present system and method is also applicable to new content input, for example as follows. FIG. **14**A shows the digital ink **1000** with an editing gesture **1400** detected by the ink management system **112** and/or the HWR system **114** as input as a top-to-bottom vertical line in a single stroke by a user swiping their finger or stylus tip from the top to the bottom in the displayed position shown, for example. That is, the stroke start position is above the stroke end position, in time order, as depicted by the gradual thickening of the stroke from the top to the bottom. The illustrated characteristic of the editing stroke is an example, and other characteristics are possible, such as a wedge, as are other forms of editing commands (described in detail later). The editing stroke **1400** is detected as passing between the strokes **1001** (and **1002**) and **1003** (and **1004**), and as such the ink management system **112** knows the relative location of the editing stroke **1400** not only to the digital ink **1000** but also to each of the candidate recognitions, e.g., **1202** to **1208**, of the underlying recognized ink.

Consideration of this relative location of the editing gesture **1400** and its top-to-bottom writing direction may result in a determination that the editing gesture **1400** is intended by the user to break or split the digital ink **1000** at the detected location. With this determination, the ink management system **112** causes the digital ink **1000** to be re-rendered as edited digital ink **1000"** as shown in FIG. **14**B, in which the strokes **1001** (and **1002**) and **1003** (and **1004**) are provided with a space therebetween. The size of this space may be pre-set and/or user settable (e.g., via a UI menu) and is generally sized to allow subsequent input to be provided in that space. However, just a normal (based on the text template or the like) sized single space may also be provided. This editing of the digital ink is accompanied by

any suitable further recognition of the digital content represented by the ink objects. In the determination or interpretation of the editing stroke **1400**, the ink management system **112** refers to the recognition candidates referenced to the strokes in the proximity of the editing stroke location, as in the example of FIG. **13**. For example, the location of the editing stroke **1400** between the strokes **1001** (and **1002**) and **1003** (and **1004**) substantially coincides with a space identified in all of the (most likely) recognition candidates **1202** to **1208**. Accordingly, the ink management system **112** determines, or causes the HWR system **114** to determine (e.g., through provision of a suitable probability weighting), that the editing gesture **1400** is an add space gesture. This determination may at least in part be performed, for example, with the gesture detection and recognition described in the afore-incorporated by reference U.S. patent application Ser. No. 14/989,217, for example.

Content insertion could also be made by overwriting on existing content. Such actions may be interpreted by the ink management system **112**, for example, as deletion of the overwritten content and insertion of the new input, or the addition of the new input to the existing content (such as turning an "l" into a "t" by addition of the bar "-"). As discussed above, this includes partial overwriting/deletion of characters, etc., and provision of undo actions through retention of the digital ink and/or reference indexes.

The ink management system **112** may be configured to display the input editing stroke(s) as digital ink, either with a shape as, for example, shown in FIGS. **13**A and **14**A, or some other appropriate rendering until the recognized editing operation is performed or may just perform the editing operation without such display. The display of the detected editing gesture may be used to provide recognition feedback to users with respect to recognition of the gesture itself and the relative location thereof. Otherwise, the eventual editing operation, e.g., the modified display shown in FIGS. **13**B and **14**B, may be used for such feedback.

FIG. **14**C shows the edited digital ink **1000"** with new input **1402** detected by the ink management system **112** and/or the HWR system **114** as being within the space between the strokes **1001** (and **1002**) and **1003** (and **1004**), thereby forming the edited phrase **1000"** of "cleaning of the keyboard". The raw (and digital) ink **1402** includes a stroke representing text characters which form the word "of". As discussed earlier, the ink management system **112** segments the stroke to produce a reference segmentation used to determine the reference index of the input, and therefore the digital ink. For example, the stroke index of the new stroke **1402** may be stroke [9], i.e., sequentially carrying on from the stroke **1009** which is stroke [8], and may be segmented to have 60 points, say, such that, for example reference index r_{22} corresponds to [9:0, 9:30] and reference index r_{23} corresponds to [9:30, 9:60]. As described above, extra precision can be provided by also referring to sub-points. In this way, the reference indexes of the previously indexed content are not disturbed and the new input is referenced in combination with the existing content.

As with non-content input, like the editing gesture **1400**, consideration of the relative location of the new input **1402**, with reference to the recognition candidates referenced to the strokes in the proximity of the new stroke location as described earlier, may result in a determination that the new input is intended by the user as new content to the digital ink **1000'** at the detected location, particularly with respect to the recognition candidates **1202** and **1204** (described in detail below). With this determination, the ink management system **112** may cause further re-rendering of the edited digital ink **1000'** to provide additional space for content insertion and then reduce this space once the new content has been input, for example. Alternatively, users may be required to insert further space to allow for this. This editing of the digital ink is accompanied by any suitable further recognition of the digital content represented by the ink objects, for example, by the ink management system **112** determining, or causing the HWR system **114** to determine (e.g., through provision of a suitable probability weighting), that the stroke **1402** is new content, such that the recognition processing of the new content may be carried out faster since the type of input has already been determined or pre-processed. Accordingly, in the recognition process the new stroke **1402** is segmented in order to determine probable character candidates which lead to the recognition of the word "of" of the new input **1402**, where this recognition is performed with reference to the existing recognition candidates but without disrupting those candidates unless this is determined to be necessary (such as characters being added to a word, for example).

In the present example, this recognition causes a change in the recognition candidates. For example, FIG. **14**D shows an example revised or edited recognition **1200"** having the representations of revised segmentations **1202'** and **1204"** of the edited phrase **1000"**. In each of the revised segmentations **1202'** and **1204"** the stroke **1402** is segmented to produce additional recognized characters of the series of recognized characters C_n. That is, the segmentation of the strokes **1001** to **1009** is unchanged. The additional recognized characters C_1, include "o" C_{20} and "f" C_{21}. As in the examples before, each of the revised recognition segmentations **1202'** and **1204"** correspond generally with the reference segmentation such that the ink management system **112** maps the expanded reference indexes r_m to the expanded candidate characters C_n. That is, the new character C_{20} corresponds to reference index r_{21}, the new character C_{21} corresponds to reference index r_{22} and the mapping of the existing candidate characters and reference indexes for the other strokes **1002** to **1009** is maintained. Accordingly, the underlying recognition **1202'** and **1204"** of the digital ink **1000** may be determined via the reference indexing as word[0:0, 1:10] corresponding to "cleaning", word[9:0, 9:60] corresponding to "of", word[2:0, 3:10] corresponding to "the", and word[4:0, 8:40] corresponding to "keyboard", for example, where a fourth ink object corresponds to the reference index [9:0, 9:60].

In this example, the recognition candidates **1206** and **1208** are omitted from the ink objects or downgraded from the most likely recognition candidates list through provision of a suitable probability weighting by the ink management system **112**, and the recognition candidates **1202** and **1204** are revised and may be elevated or promoted in the most likely recognition candidates list through provision of a suitable probability weighting by the ink management system **112**. This is because, in both of the recognition candidates **1206** and **1208** the strokes **1001** and **1002** correspond to the candidate recognition "during" and based on the language model the HWR system **114** determines the likelihood of the phrases "during of the keyboard" or "during of the ken board", for example, to be relatively reduced from the recognition of the original input **1000**. On the other hand, the relative likelihoods of the revised recognition candidates **1202'** and **1204"** of "cleaning of the keyboard" and "cleaning of the ken board", respectively, are most likely about the same as for the original recognition candidates **1202** and **1204**.

Accordingly, the ink management system **112** manages interactions to edit the digital ink by adjusting the ink objects

corresponding to each of the digital ink characters, words, etc., that are interacted with through re-rendering of the digital ink and revision or adjustment of the underlying recognition candidates corresponding thereto. As such, the ink objects are updated with any editions made so that upon conversion of the digital ink content into typeset or fontified content (e.g., upon finalization of a document), the typesetting is performed based on the updated ink objects so that the most current and applicable recognition results are used in the conversion.

Further, as with the digital ink and/or reference indexes of the deletion and insertion actions described above, the ink objects may be retained by the ink management system **112** after typesetting is performed so that undo actions can be similarly taken to revert back to the digital ink. The ink objects may be retained upon conversion to digital objects, either permanently or for a certain amount of time or number of subsequent operations, for example, by storing the ink objects in the memory **108** of the device **100**. Alternatively, the retention of the ink objects through typesetting may be performed by merely re-rendering the digital ink as typeset ink, so that the digital content remains characterized by the ink objects, rather than converted to digital objects. In this way, interactions with the typeset ink are treated by the ink management system **112** in similar fashion to the interactions with the digital ink. That is, reference is made to the related underlying recognition results such that similar functionality is provided. This is particularly useful in providing the ability to input mixed content, e.g., handwriting revisions or annotations on typeset content, where the handwritten input is interpreted with reference to the existing ink objects to provide seamless integration of the handwriting into the typeset content. The reverse is also possible, with typed content being added to handwritten digital ink. In this case, the decoded keystrokes may be treated by the ink management system **112** as recognition candidates so that the typed content is formulated as ink objects. Similar possibilities apply for mixed non-text content which is input with handwriting or with digital input tools.

As mentioned earlier, editing commands other than the handwritten editing gestures may also be used to edit the ink objects. These editing commands include traditional editing commands such as those provided via UI menus, keyboard (e.g., backspace) and mouse (e.g., select). That is, as discussed earlier, the reference indexing allows substantially precise placement of an input cursor with respect to the digital ink and also provides precise knowledge of the extent of content elements, such as text characters. Accordingly, regardless of whether the rendered ink is digital ink or typeset ink, a cursor can be positioned at element boundaries, such as at character boundaries, and the traditional editing commands can therefore be used similar to digital objects. For example, FIG. **15**A shows the digital ink paragraph **500** of FIG. **5** with a portion **1500** of the paragraph selected and FIG. **15**B shows the corresponding typeset ink paragraph **600** of FIG. **6** with a portion **1502** of the paragraph selected. In both drawings the selections are of the same content and are shown as a highlighted selection region as commonly provided in document processing applications. The selection regions **1500** and **1502** are rendered at the character boundaries of the respective digital and typeset ink characters, that is with the cursor positions. Accordingly, in both cases the ink objects corresponding to the reference indexes of the selected content are selected for any subsequent interaction.

Use of handwriting on computing devices for the creation of digital content is advantageous because these devices do not have the physical limitations of paper, such as size, number of pages. However, creating digital content with handwriting is not currently as easy or as effective as typing because of the way digital ink is traditionally managed. Conventionally, digital content is made of typed text objects or digitally drawn non-text objects, such as shapes and equations. Current digital ink management captures handwritten objects as pixels or strokes, which does not fit with the digital content model. This leads to creators having to re-enter content by typing with keyboard or using complex tools for creation and editing of non-text content.

Handwriting recognition technology can avoid this re-entry paradigm. However, as handwriting is generally unstructured with text of various size and orientation and combined with non-text, performing recognition on the whole content at once, as it is conventionally done, does not provide the best use of this technology. This leads to creators having to make many post-recognition edits to create publishable digital content, significantly limiting the productivity gain that handwriting recognition could provide.

The present system and method dramatically improves the productivity of digital ink through the ink management system. The ink management system provides interactive ink that is as easily manageable as typed digital content through the combination of the digital ink with "on-the-fly" recognition, for text and non-text. This combination of the digital ink and the recognition results transforms the digital ink into ink objects which are interactive like typed objects and can be manipulated with intuitive gestures akin to those of keyboard, like backspace and enter, and mouse, like select and drag'n'drop. Interaction with the digital ink provides an intuitive user experience, even for non-tech savvy users.

While the foregoing has described what is considered to be the best mode and/or other examples, it is understood that various modifications may be made therein and that the subject matter disclosed herein may be implemented in various forms and examples, and that they may be applied in numerous other applications, combinations, and environments, only some of which have been described herein. Those of ordinary skill in that art will recognize that the disclosed aspects may be altered or amended without departing from the true spirit and scope of the subject matter. Therefore, the subject matter is not limited to the specific details, exhibits, and illustrated examples in this description. It is intended to protect any and all modifications and variations that fall within the true scope of the advantageous concepts disclosed herein.

We claim:

1. A system for providing interactive ink on a computing device, the computing device comprising a processor, a memory and at least one non-transitory computer readable medium for recognizing handwriting input under control of the processor, the at least one non-transitory computer readable medium configured to:

cause display of, on a display interface of the computing device, first digital ink in accordance with first handwriting input;

allocate spatial or temporal references to ink elements of the first digital ink;

map the spatial or temporal references to corresponding recognized elements of the first handwriting input; and

determine and store, in the memory of the computing device, ink objects including the spatial or temporal references and mapped recognized elements.

2. The system according to claim **1**, wherein the ink elements are at least part of strokes of the first digital ink.

3. The system according to claim 1, wherein the recognized elements include at least part of strokes of the first handwriting input.

4. The system according to claim 3, wherein the recognized elements are recognition candidates of the first handwriting input.

5. The system according to claim 1, the at least one non-transitory computer readable medium configured to:

determine content of second handwriting input by referring to the ink objects; and

cause display of, on the display interface, second digital ink in accordance with the determined content of the second handwriting input.

6. The system according to claim 5, wherein the second digital ink is an edited version of at least part of the first digital ink.

7. A method for providing interactive ink on a computing device, the computing device comprising a processor and at least one non-transitory computer readable medium for recognizing handwriting input under control of the processor, the method comprising:

displaying, on a display interface of the computing device, first digital ink in accordance with first handwriting input;

allocating spatial or temporal references to ink elements of the first digital ink;

map the spatial or temporal references to corresponding recognized elements of the first handwriting input; and

determining and storing, in the memory of the computing device, ink objects including the spatial or temporal references and mapped recognized elements.

8. The method according to claim 7, wherein the ink elements are at least part of strokes of the first digital ink.

9. The method according to claim 7, wherein the recognized elements include at least part of strokes of the first handwriting input.

10. The method according to claim 9, wherein the recognized elements are recognition candidates of the first handwriting input.

11. The method according to claim 7, further comprising:

determining content of second handwriting input by referring to the ink objects; and

displaying, on the display interface, second digital ink in accordance with the determined content of the second handwriting input.

12. The method according to claim 11, wherein the second digital ink is an edited version of at least part of the first digital ink.

13. A non-transitory computer readable medium having a computer readable program code embodied therein, said computer readable program code adapted to be executed to implement a method for providing interactive ink to a computing device, the computing device comprising a processor and at least one system non-transitory computer readable medium for recognizing handwriting input under control of the processor, the method comprising:

displaying, on a display interface of the computing device, first digital ink in accordance with first handwriting input;

allocating spatial or temporal references to ink elements of the first digital ink;

map the spatial or temporal references to corresponding recognized elements of the first handwriting input; and

determining and storing, in the memory of the computing device, ink objects including the spatial or temporal references and mapped recognized elements.

14. The non-transitory computer readable medium according to claim 13, wherein the ink elements are at least part of strokes of the first digital ink.

15. The non-transitory computer readable medium according to claim 13, wherein the recognized elements include at least part of strokes of the first handwriting input.

16. The non-transitory computer readable medium according to claim 15, wherein the recognized elements are recognition candidates of the first handwriting input.

17. The non-transitory computer readable medium according to claim 13, further comprising:

determining content of second handwriting input by referring to the ink objects; and

displaying, on the display interface, second digital ink in accordance with the determined content of the second handwriting input.

18. The non-transitory computer readable medium according to claim 17, wherein the second digital ink is an edited version of at least part of the first digital ink.

* * * * *

EXHIBIT J

US011402991B2

(12) **United States Patent**
Rucine et al.

(10) **Patent No.: US 11,402,991 B2**
(45) **Date of Patent: Aug. 2, 2022**

(54) **SYSTEM AND METHOD FOR NOTE TAKING WITH GESTURES**

(71) Applicant: **MyScript**, Nantes (FR)

(72) Inventors: **Nicolas Rucine**, Nantes (FR); **Thomas Penin**, Nantes (FR); **Nathalie Delbecque**, Nantes (FR); **Jean-Nicolas Guiavarc'h**, Nantes (FR); **Julien Vergne**, Nantes (FR); **Gildas Lemarie**, Nantes (FR); **Alicja Wojtczuk**, Nantes (FR); **Louise Bonnamy**, Nantes (FR); **Claire Sidoli**, Nantes (FR); **Joanne Ho**, Nantes (FR); **Arnoud Boekhoorn**, Nantes (FR)

(73) Assignee: **MyScript**, Nantes (FR)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 97 days.

(21) Appl. No.: **14/989,217**

(22) Filed: **Jan. 6, 2016**

(65) **Prior Publication Data**

US 2017/0153806 A1 Jun. 1, 2017

(30) **Foreign Application Priority Data**

Dec. 1, 2015 (EP) 15290296

(51) **Int. Cl.**
G06F 3/01 (2006.01)
G06F 3/0483 (2013.01)
(Continued)

(52) **U.S. Cl.**
CPC ***G06F 3/04883*** (2013.01); ***G06F 3/017*** (2013.01); ***G06F 40/166*** (2020.01);
(Continued)

(58) **Field of Classification Search**
CPC G06F 3/0488; G06F 17/242; G06F 17/211; G06F 3/04883; G06F 17/241; G06F 17/24; G06F 3/017; G06K 9/00402
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,220,649 A 6/1993 Forcier
5,784,504 A * 7/1998 Anderson G06F 3/04883 382/309
(Continued)

FOREIGN PATENT DOCUMENTS

JP 2003-208569 A 7/2003
JP 2013-218592 A 10/2013
JP 2013-246633 A 12/2013

OTHER PUBLICATIONS

International Search Report for Related Application No. PCT/EP2016/002032, dated Feb. 15, 2017, EPO-Internal, WPI Data.
(Continued)

Primary Examiner — Shahid K Khan
Assistant Examiner — Ahamed I Nazar
(74) *Attorney, Agent, or Firm* — McDermott Will and Emery LLP

(57) **ABSTRACT**

A system, method and computer program product for use in editing digital documents with handwriting input to a computing device are provided. The computing device is connected to an input device in the form of an input surface. A user is able to provide input by applying pressure to or gesturing above the input surface using either his or her finger or an instrument such as a stylus or pen. The present system and method monitors the input strokes. The computing device further has a processor and at least one application for recognizing the handwriting input under control of the processor. The at least one system application is configured to cause display of, on a display interface of a
(Continued)



computing device, first digital ink in accordance with document content, detect handwriting input as one or more editing gestures with respect to the document content, and, cause display of, on the display interface, second digital ink in accordance with handwriting input.

19 Claims, 52 Drawing Sheets

(51) **Int. Cl.**

G06K 9/00	(2022.01)
G06F 3/04883	(2022.01)
G06F 40/166	(2020.01)
G06F 40/169	(2020.01)
G06F 40/171	(2020.01)
G06V 30/32	(2022.01)

(52) **U.S. Cl.**
CPC ***G06F 40/169*** (2020.01); ***G06F 40/171*** (2020.01); ***G06V 30/32*** (2022.01)

(56) **References Cited**

U.S. PATENT DOCUMENTS

6,499,043	B1 *	12/2002	Forcier	G06F 3/0488 715/273
7,259,752	B1 *	8/2007	Simmons	G06F 3/04817 345/173
7,751,623	B1 *	7/2010	Simmons	G06F 17/242 345/173
8,245,130	B1 *	8/2012	Lloyd	G06F 17/30905 715/234
8,250,486	B2 *	8/2012	Lentz	G06F 3/04812 715/711
8,448,083	B1 *	5/2013	Migos	G06F 3/04883 345/671
8,634,645	B2 *	1/2014	Xing	G06F 17/245 382/181
9,164,777	B2 *	10/2015	Prasad	G06F 9/4443
2002/0059350	A1 *	5/2002	Iwema	G06F 3/04883 715/234
2002/0064308	A1 *	5/2002	Altman	G06F 3/04883 382/187
2004/0070573	A1 *	4/2004	Graham	G06F 3/04883 345/179
2005/0111736	A1 *	5/2005	Hullender	G06F 3/04883 382/188
2006/0001656	A1 *	1/2006	LaViola	G06F 3/04883 345/179
2006/0018546	A1 *	1/2006	Lagardere	G06K 9/222 382/186
2006/0045337	A1 *	3/2006	Shilman	G06K 9/00402 382/181
2006/0093219	A1 *	5/2006	Gounares	G06F 3/0481 382/187
2008/0019591	A1 *	1/2008	Iwayama	G06F 3/04883 382/187
2010/0066691	A1 *	3/2010	Li	G06F 40/171 345/173
2014/0079317	A1 *	3/2014	Hirabayashi	G06F 3/0488 382/165
2014/0223382	A1 *	8/2014	Hicks	G06F 3/04883 715/863
2014/0300609	A1 *	10/2014	Kang	G06F 3/04883 345/467
2015/0169975	A1 *	6/2015	Kienzle	G06T 1/0007 382/189
2015/0286886	A1 *	10/2015	Wimmer	G06K 9/52 382/173
2017/0060819	A1	3/2017	Rucine et al.	

OTHER PUBLICATIONS

JPO—Official Notice of Reasons for Rejection (Examination Report) for related Japanese Patent Application No. 2018-528680 (translations in English and Japanese), dated Dec. 15, 2020, 8 pgs.

* cited by examiner

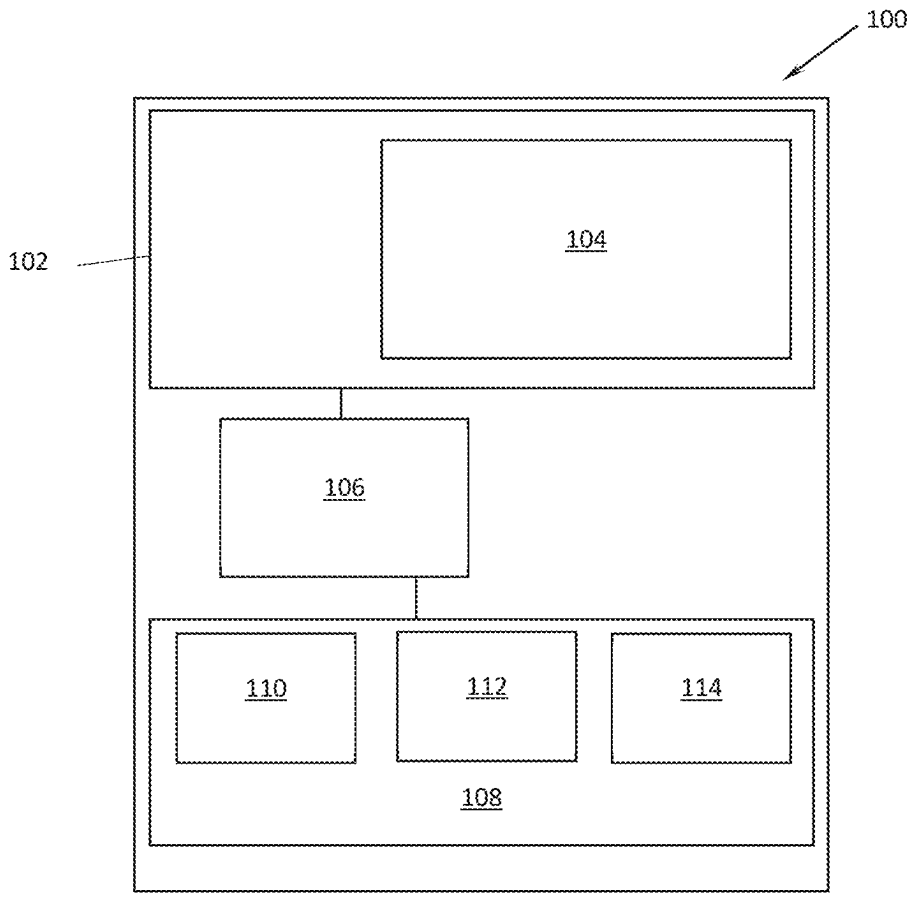


FIG. 1

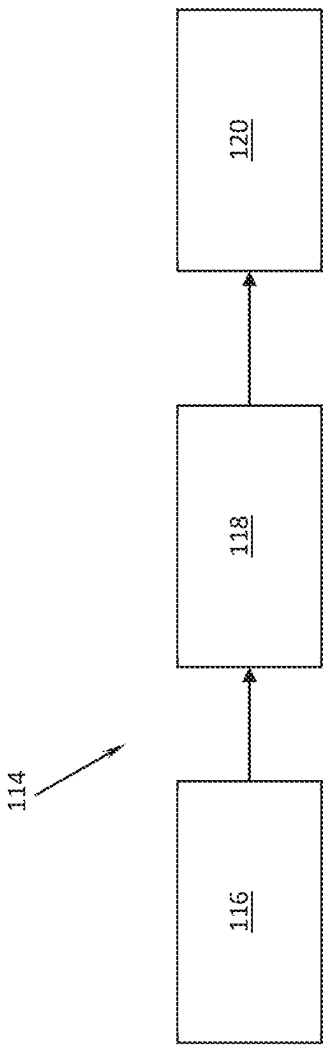


FIG. 2

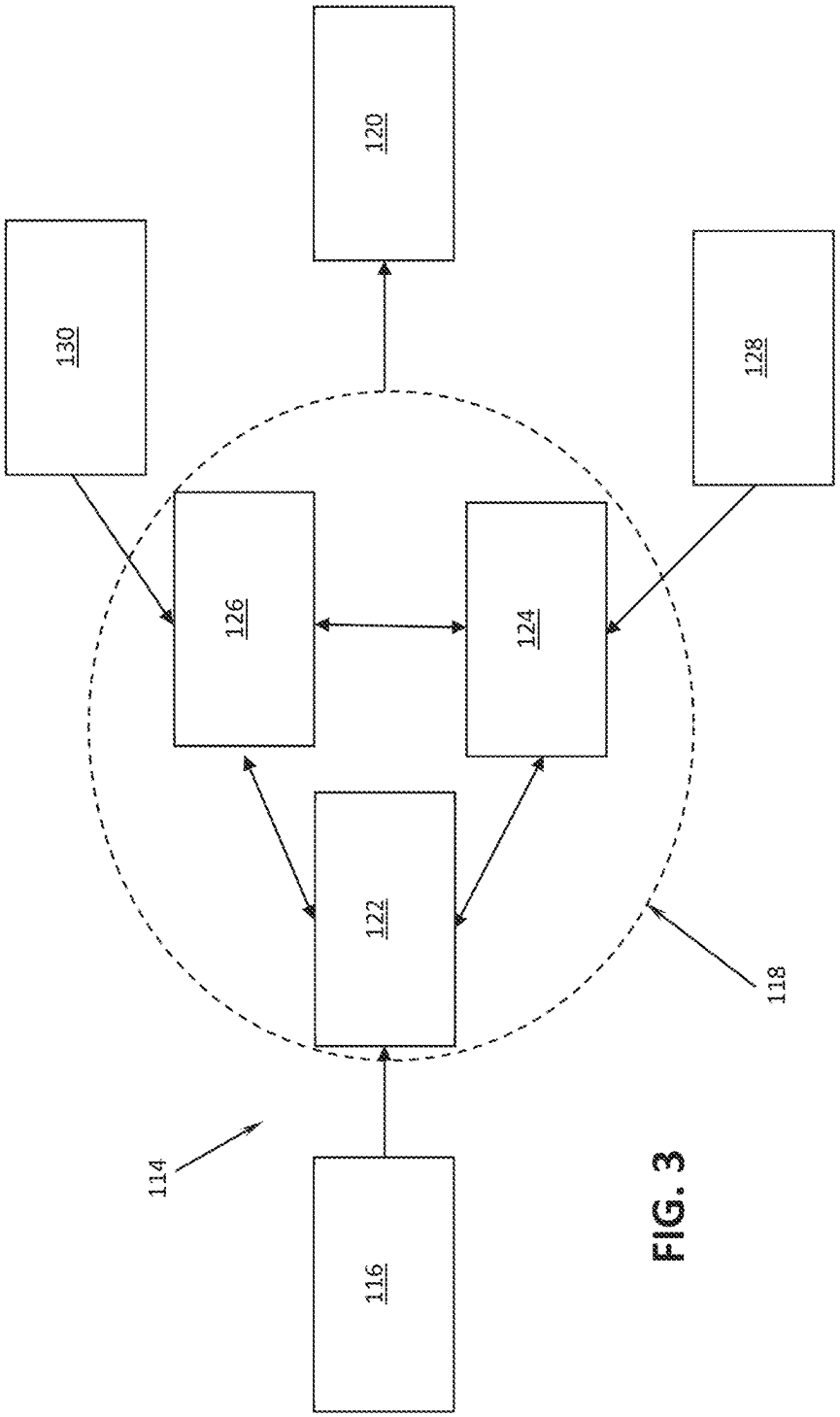


FIG. 3

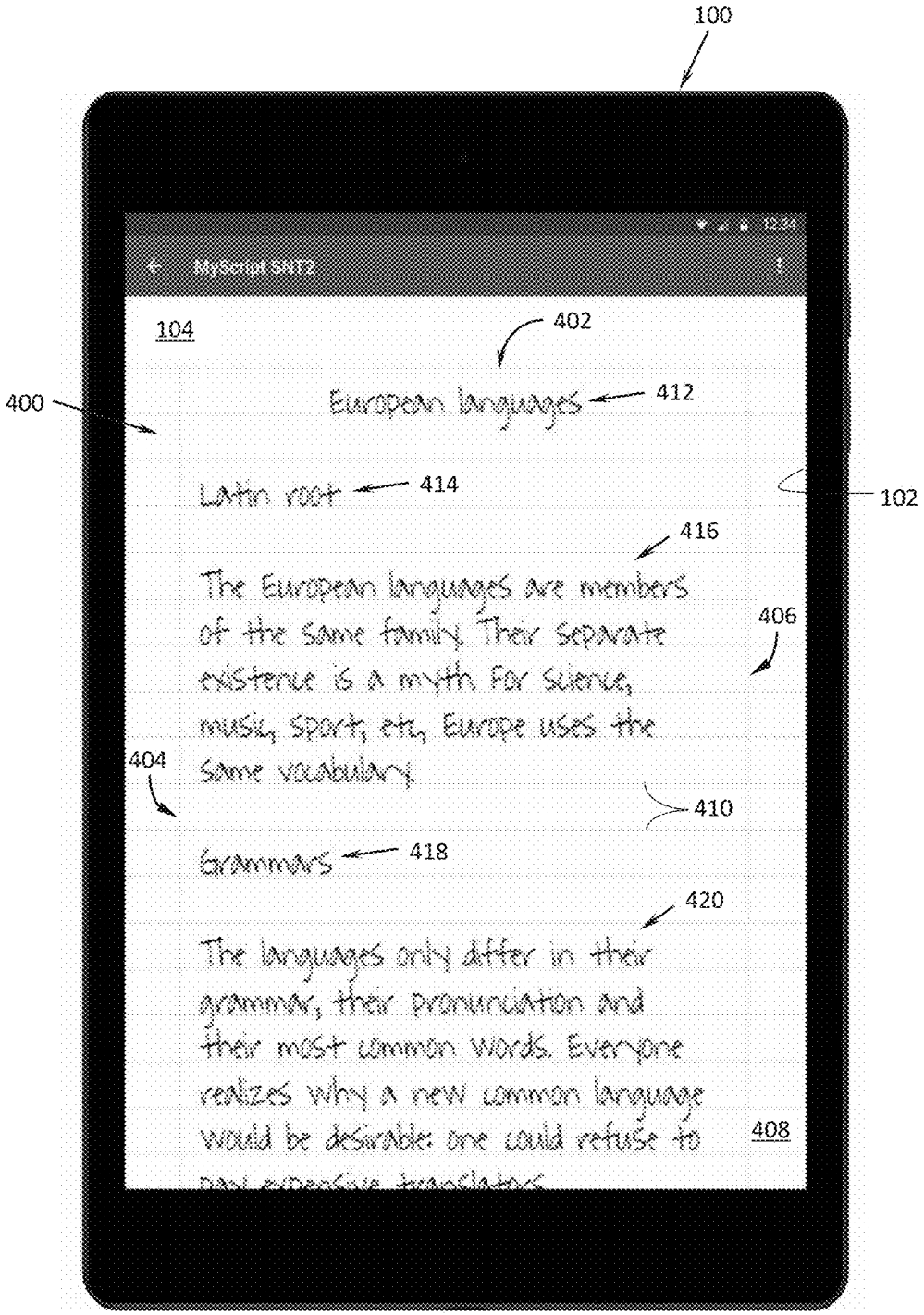

100
MyScript Stilo
104
402
400
European languages
412
Latin root
414
102
416
The European languages are members of the same family. Their separate existence is a myth. For science, music, sport, etc, Europe uses the same vocabulary.
406
404
410
Grammars
418
420
The languages only differ in their grammar, their pronunciation and their most common words. Everyone realizes why a new common language would be desirable: one could refuse to
408

FIG. 4

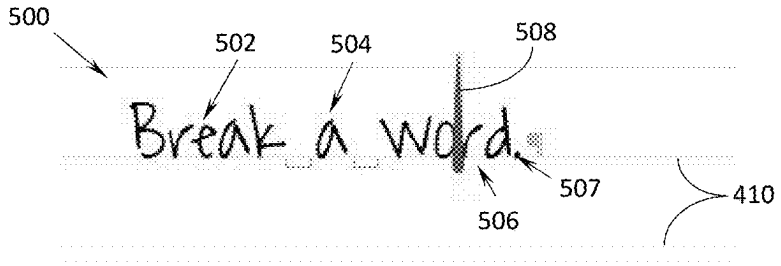


FIG. 5A

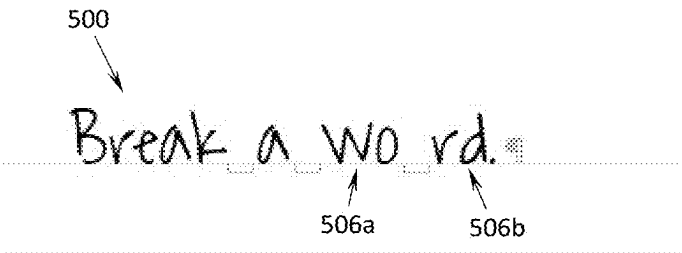


FIG. 5B

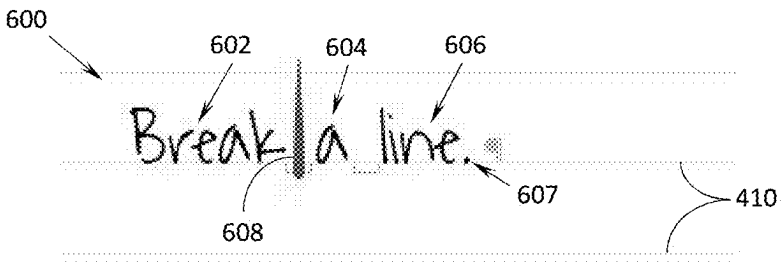


FIG. 6A



FIG. 6B

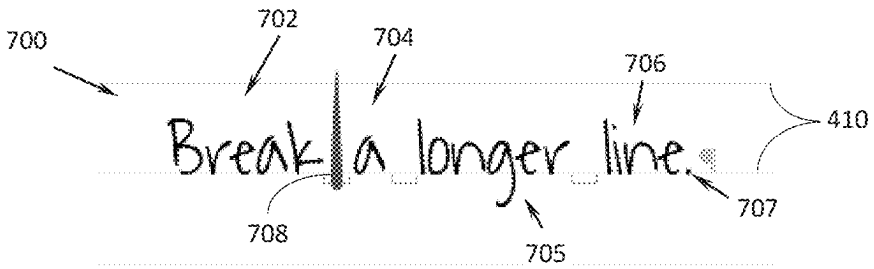

700
702
704
706
410
Break a longer line.
707
708
705

FIG. 7A

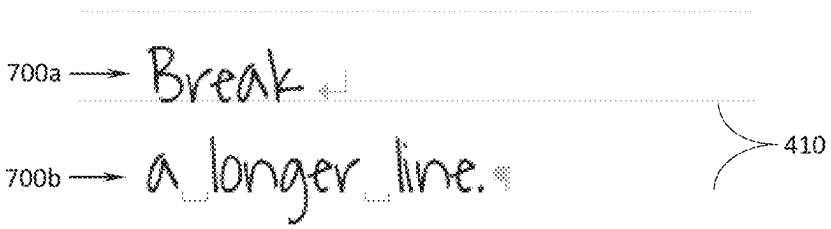

700a
Break
700b
a longer line.
410

FIG. 7B

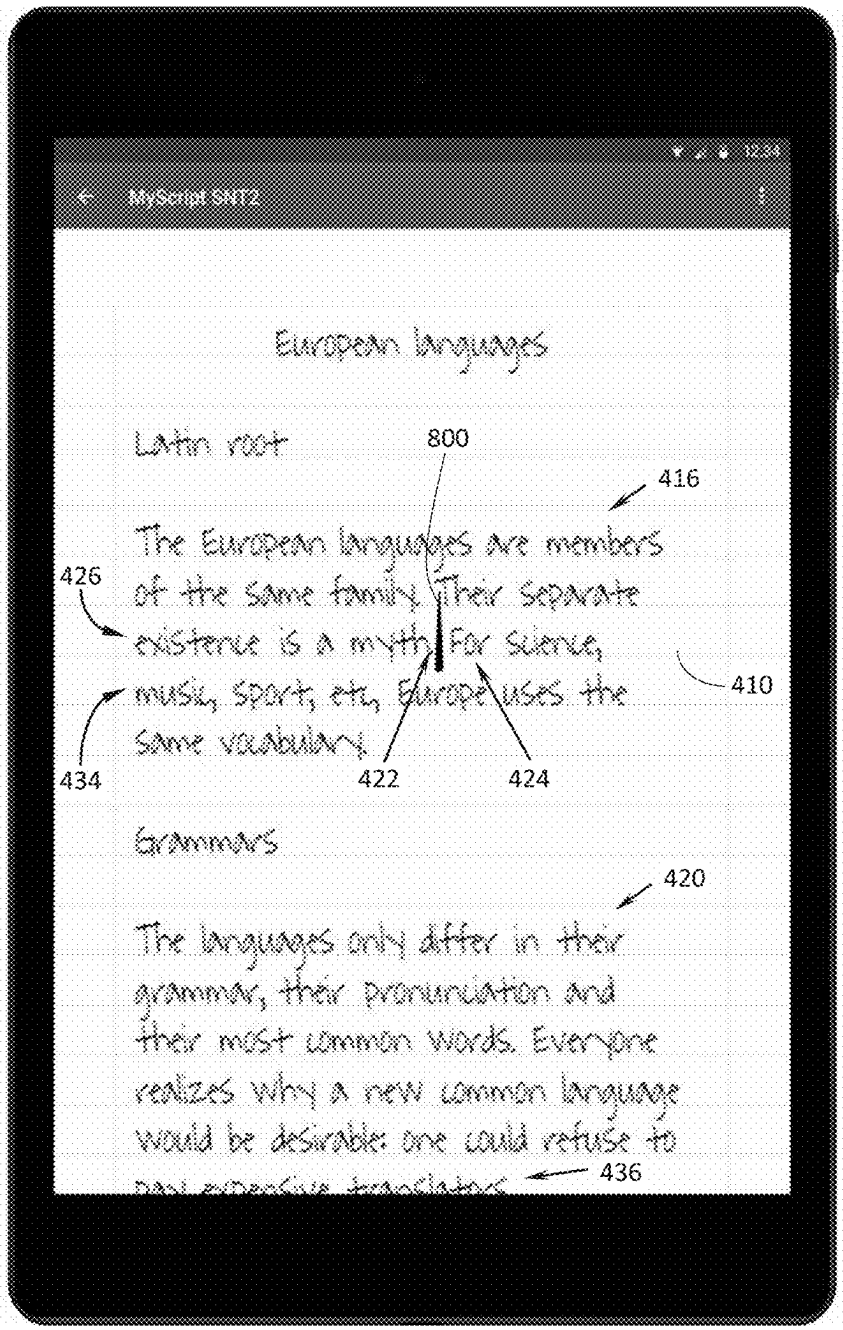

MyScript SMT2
European languages
Latin root
800
416
The European languages are members of the same family. Their separate existence is a myth. For science, music, sport, etc, Europe uses the same vocabulary.
426
410
434
422
424
Grammars
420
The languages only differ in their grammar, their pronunciation and their most common words. Everyone realizes why a new common language would be desirable: one could refuse to
436

FIG. 8A

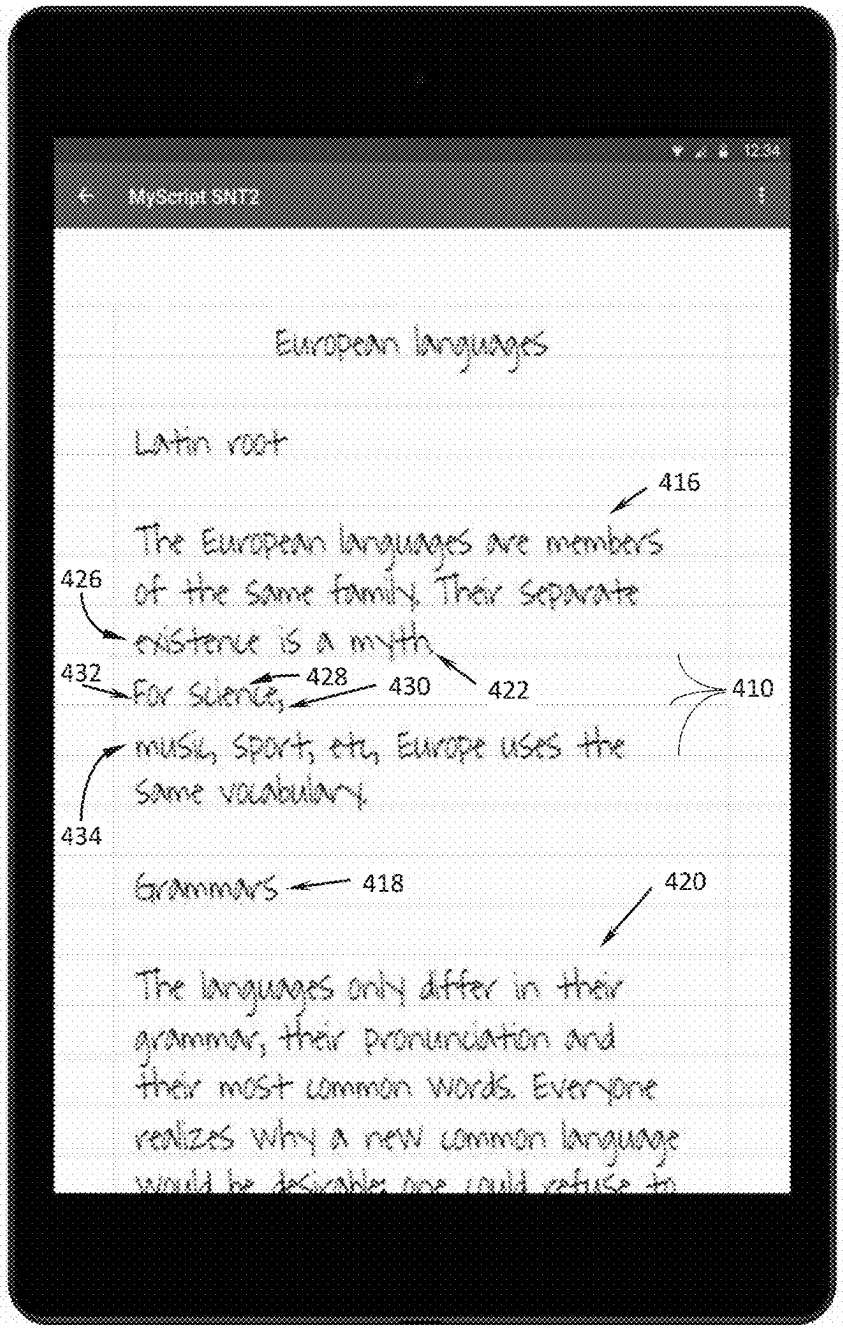

MyScript SNT2
European languages
Latin root
416
The European languages are members
of the same family. Their separate
426
existence is a myth.
For science,
432
428
430
422
410
music, sport, etc, Europe uses the
same vocabulary.
434
Grammars
418
420
The languages only differ in their
grammar, their pronunciation and
their most common words. Everyone
realizes why a new common language

FIG. 8B

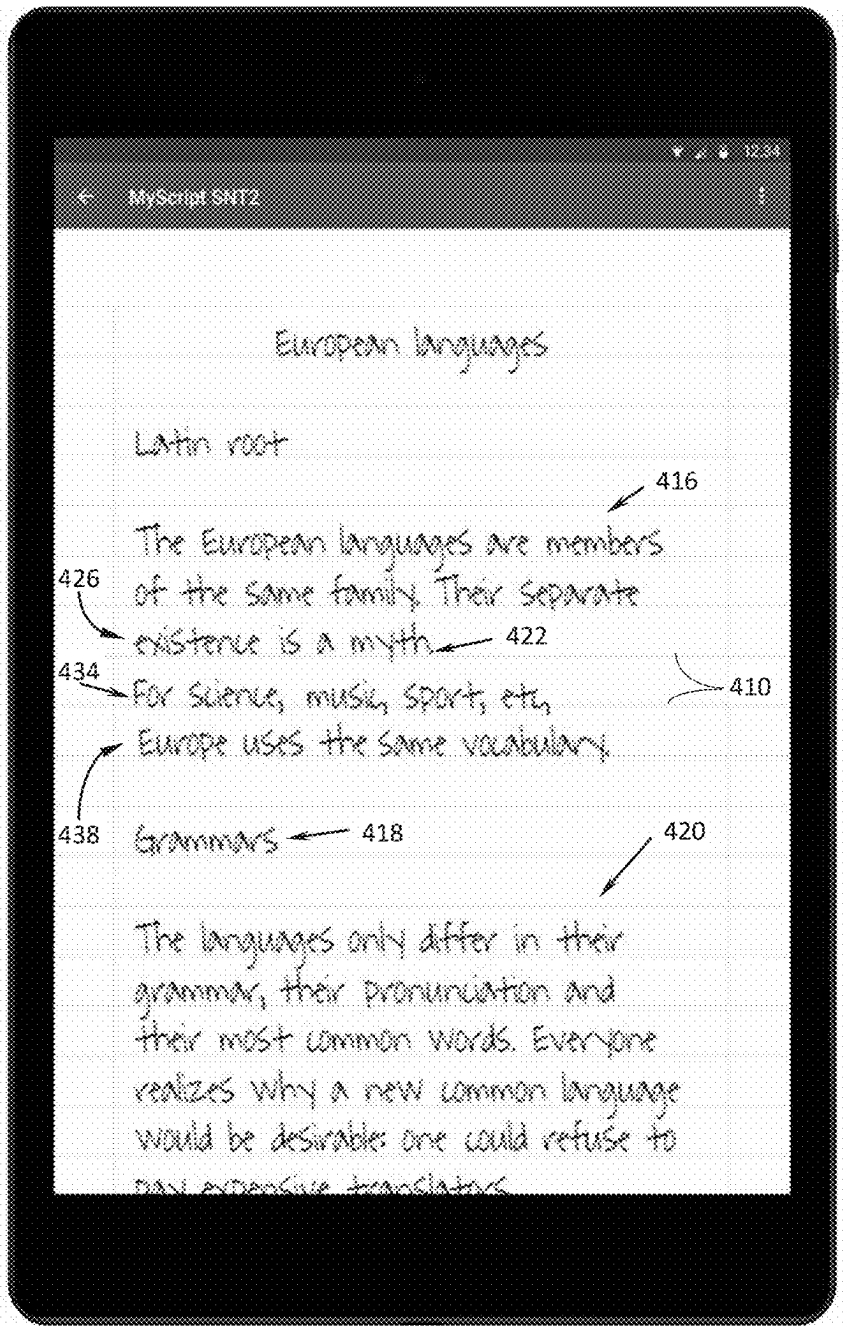

MyScript SMT2
European languages
Latin root
416
The European languages are members
of the same family. Their separate
426
existence is a myth
422
434
For science, music, sport, etc,
410
Europe uses the same vocabulary.
438
Grammars
418
420
The languages only differ in their
grammar, their pronunciation and
their most common words. Everyone
realizes why a new common language
would be desirable: one could refuse to

FIG. 8C

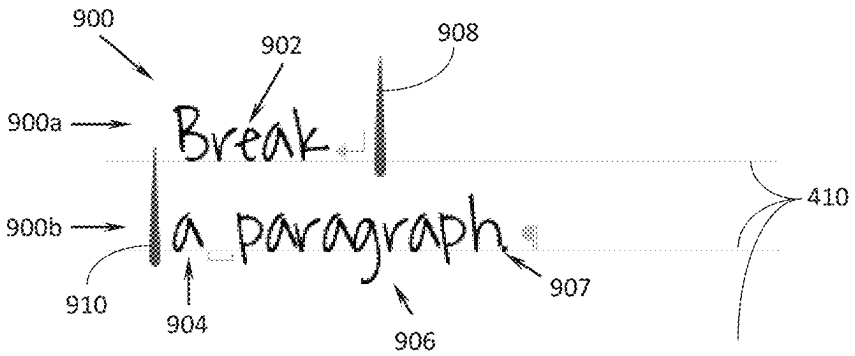

900
902
908
900a
Break
900b
a paragraph
410
910
904
906
907

FIG. 9A

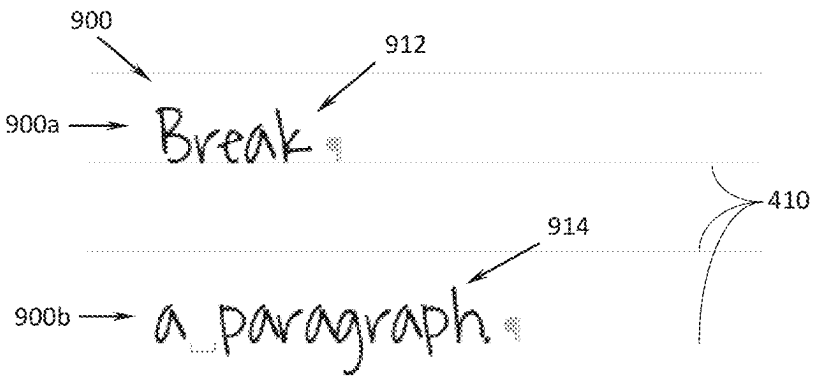

900
912
900a
Break
410
914
900b
a paragraph

FIG. 9B

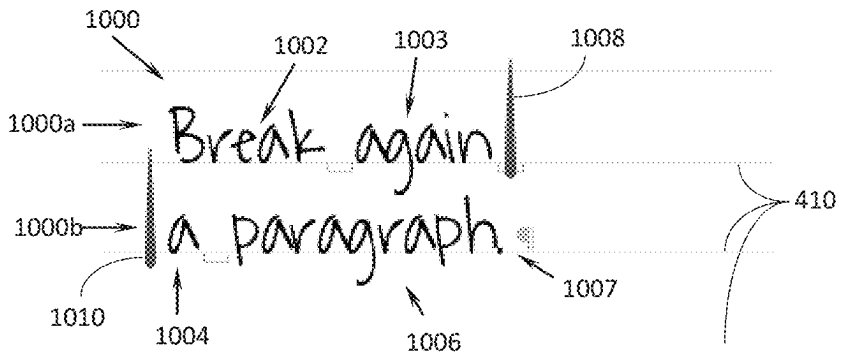


FIG. 10A

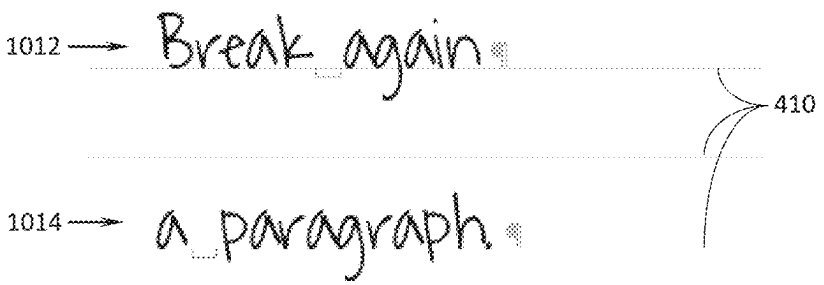


FIG. 10B

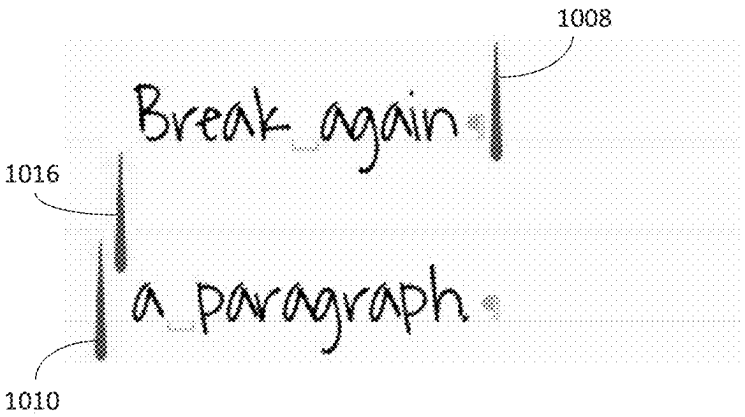


FIG. 10C

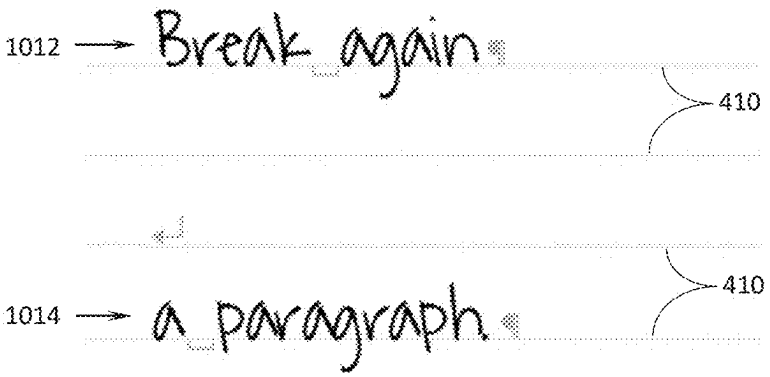


FIG. 10D

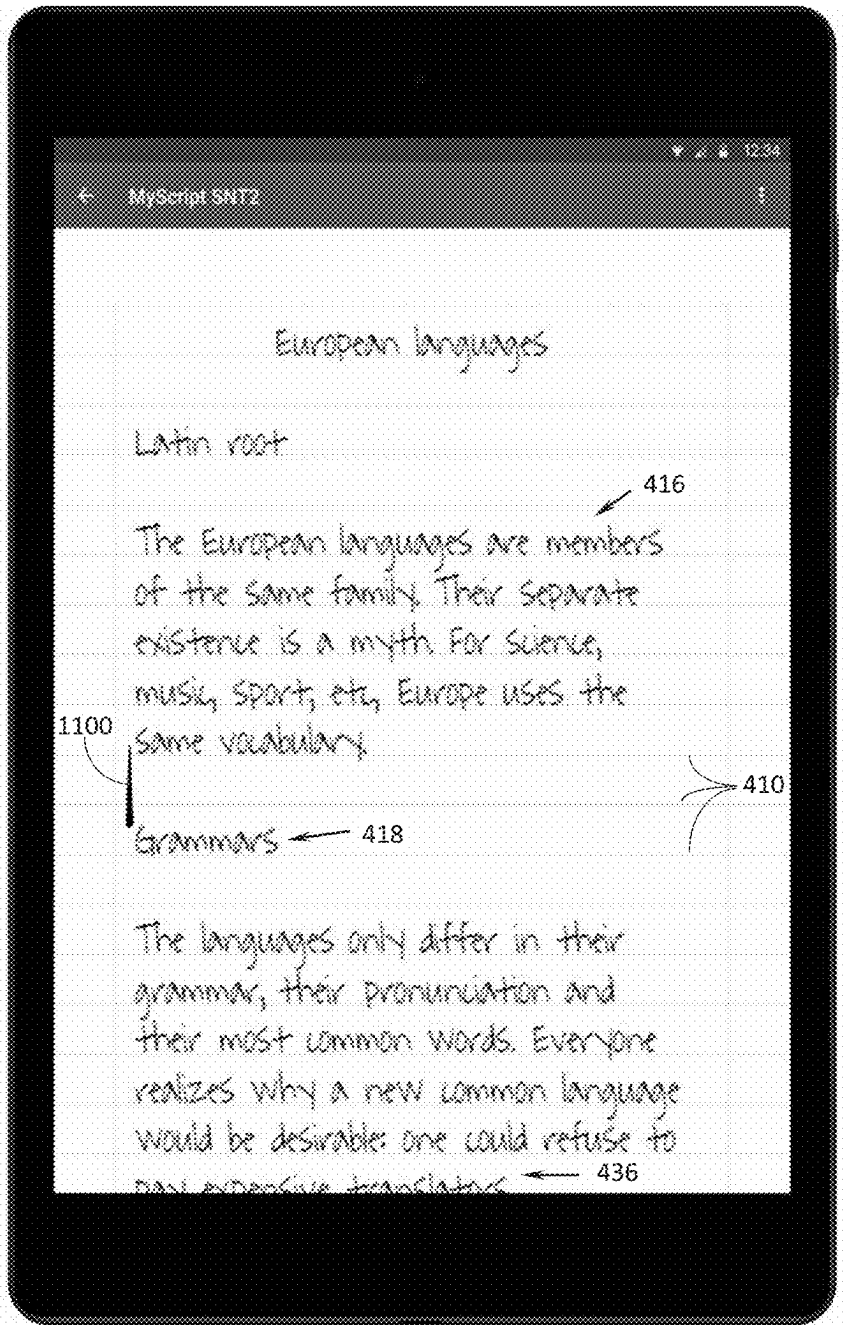

MyScript SNT2
European languages
Latin root
416
The European languages are members of the same family. Their separate existence is a myth. For science, music, sport, etc, Europe uses the same vocabulary.
1100
410
Grammars
418
The languages only differ in their grammar, their pronunciation and their most common words. Everyone realizes why a new common language would be desirable: one could refuse to
436

FIG. 11A

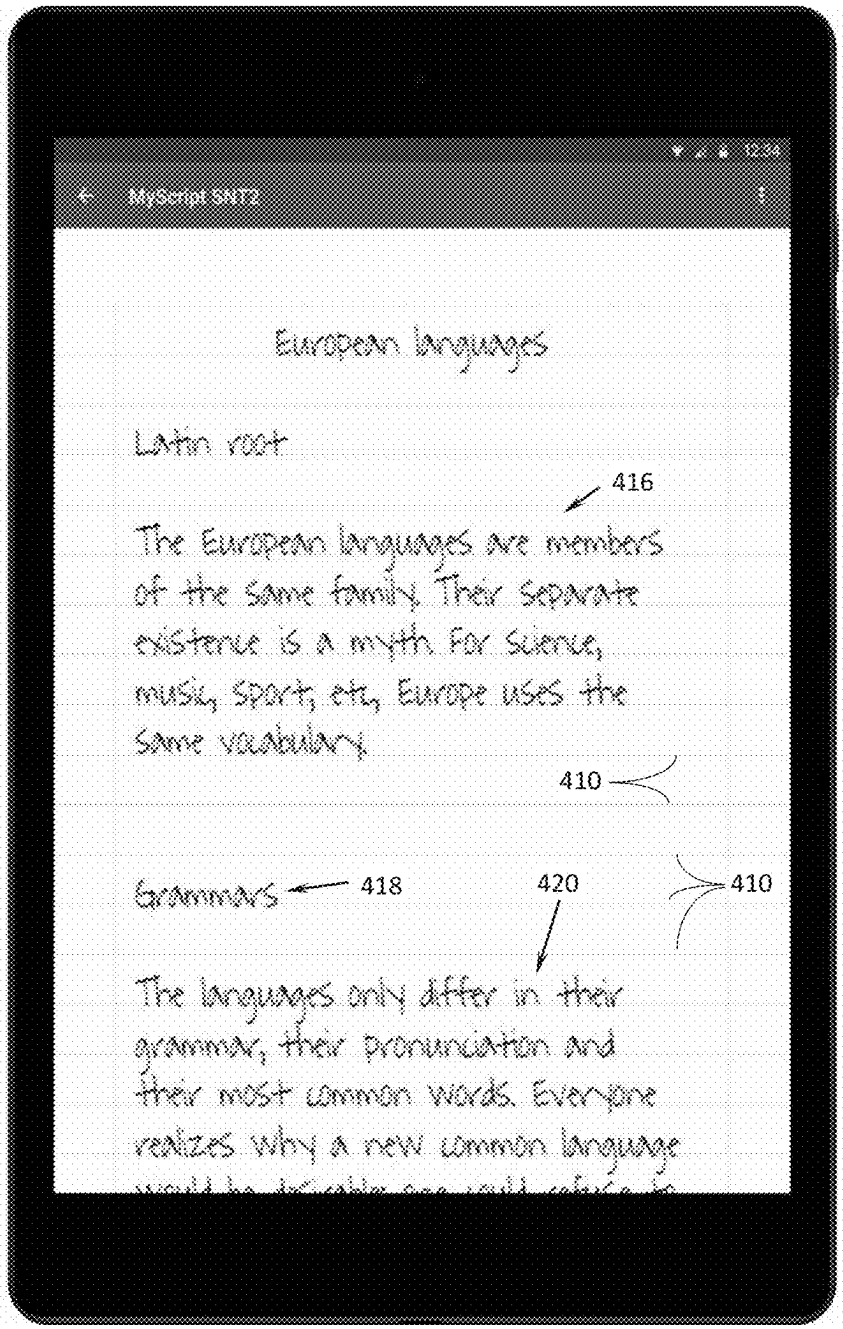

MyScript SNT2
European languages
Latin root
416
The European languages are members of the same family. Their separate existence is a myth. For science, music, sport, etc, Europe uses the same vocabulary.
410
Grammars
418
420
410
The languages only differ in their grammar, their pronunciation and their most common words. Everyone realizes why a new common language

FIG. 11B

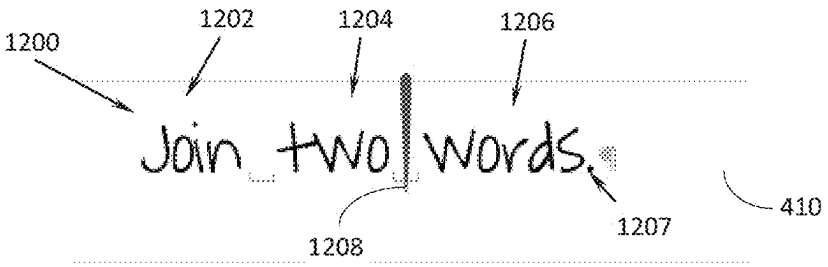


FIG. 12A

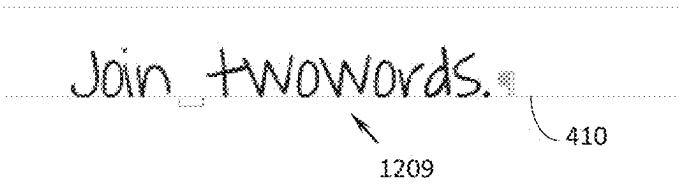


FIG. 12B

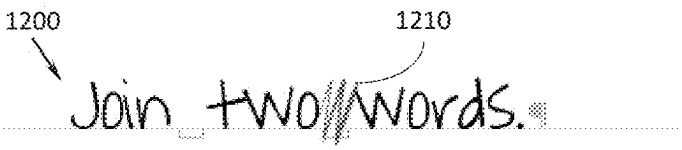


FIG. 12C

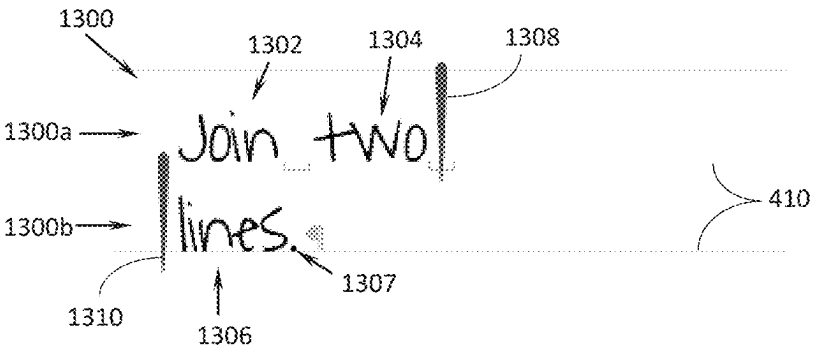


FIG. 13A

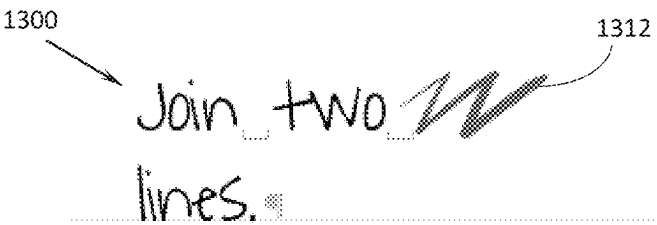


FIG. 13B

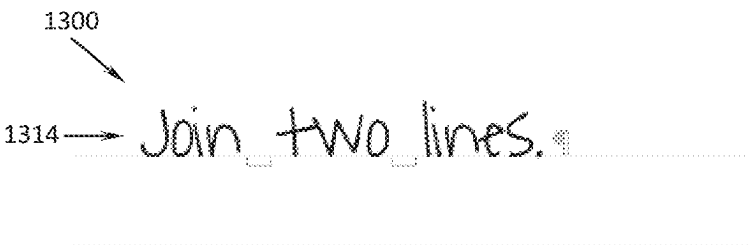


FIG. 13C

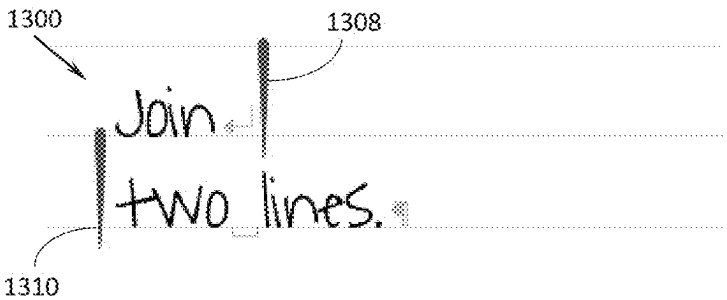


FIG. 13D

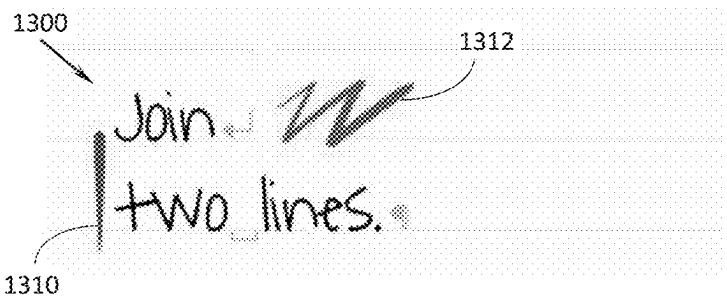


FIG. 13E

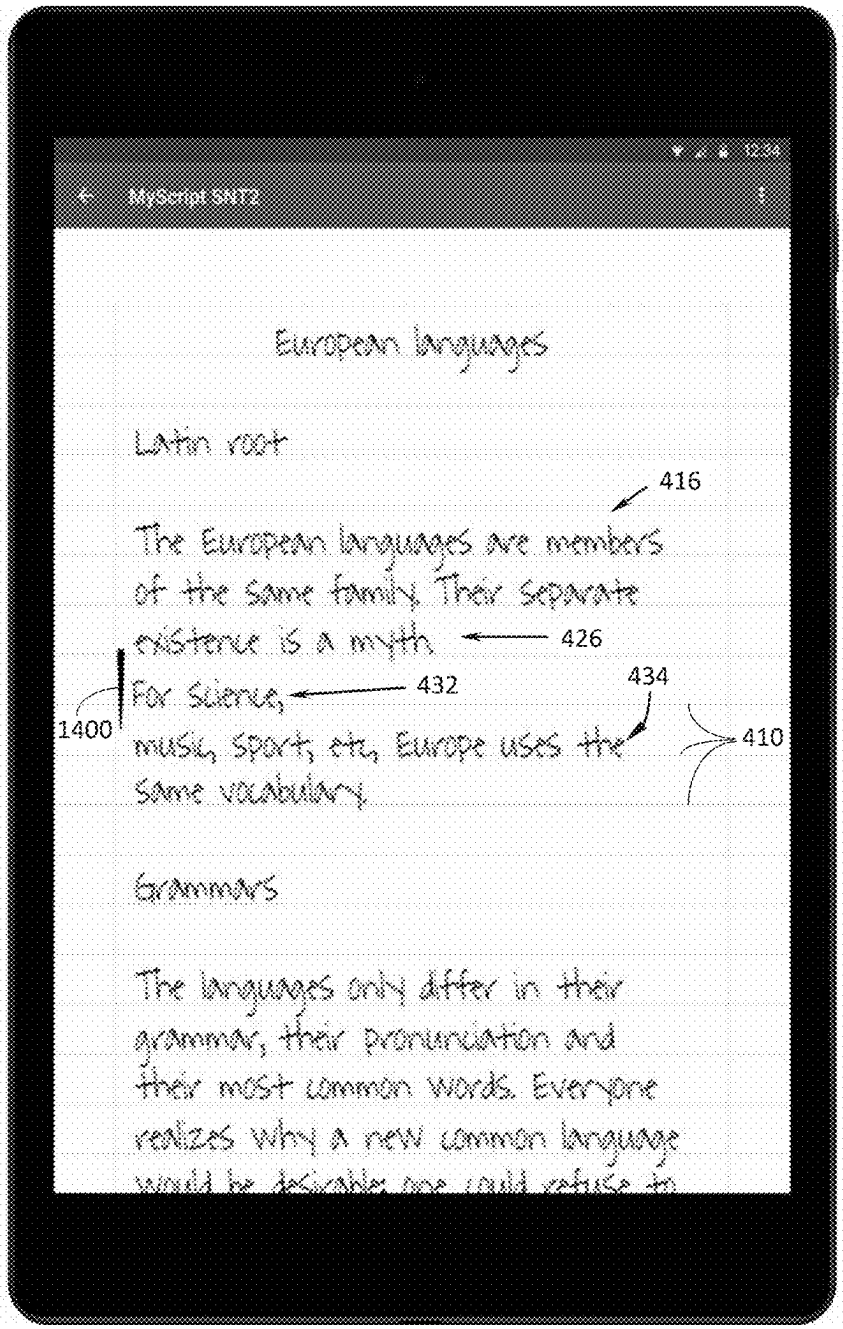

MyScript SNT2
European languages
Latin root
416
The European languages are members of the same family. Their separate existence is a myth
426
For Science,
432
434
music, sport, etc, Europe uses the same vocabulary
1400
410
Grammars
The languages only differ in their grammar, their pronunciation and their most common words. Everyone realizes why a new common language

FIG. 14

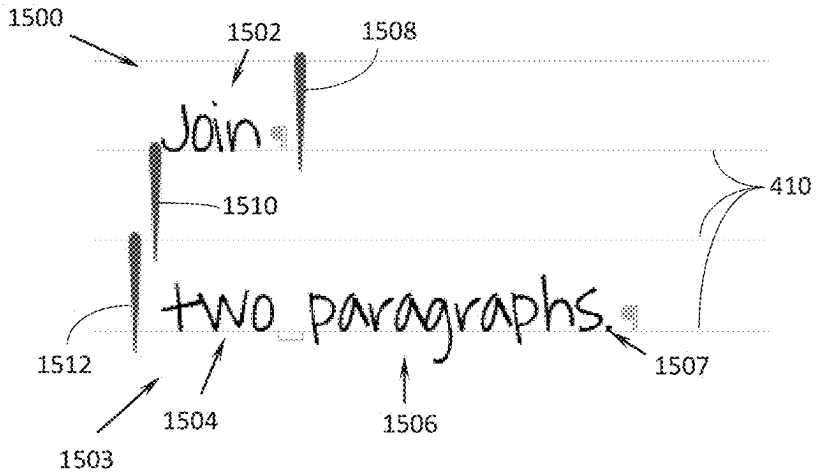


FIG. 15A

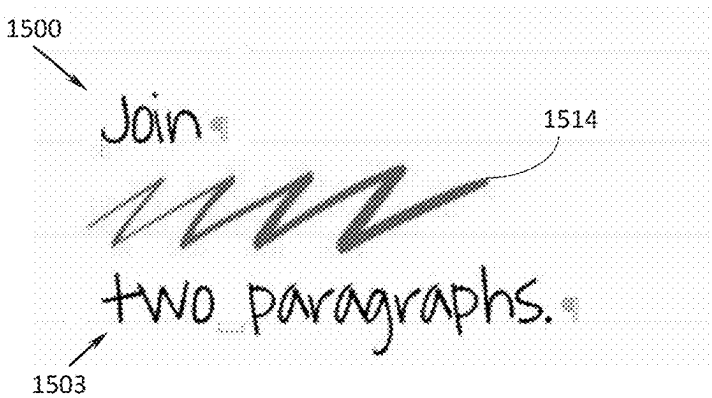


FIG. 15B

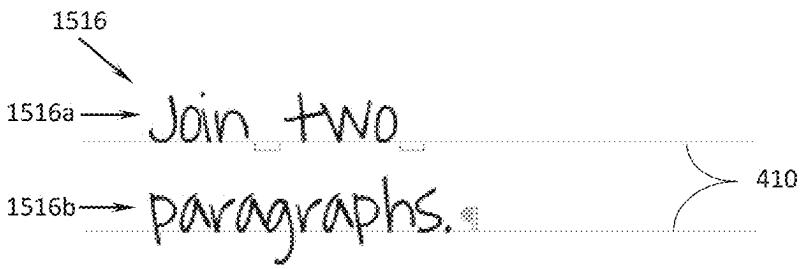


FIG. 15C

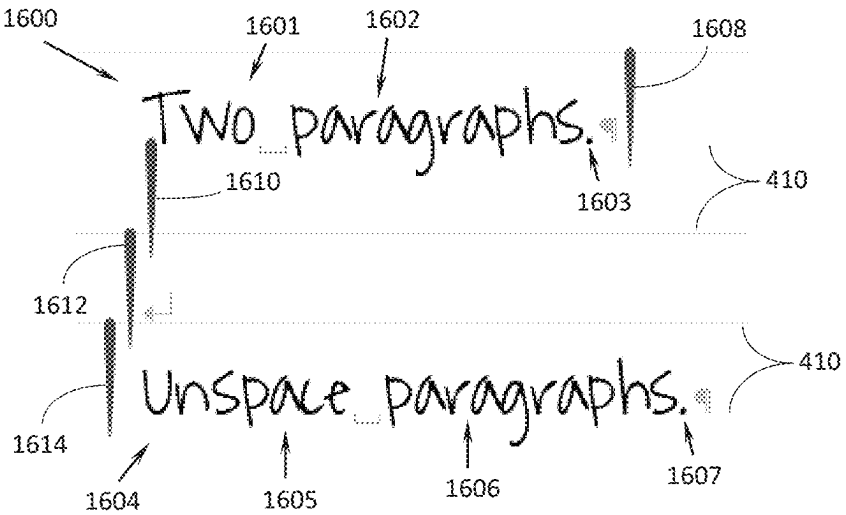


FIG. 16A

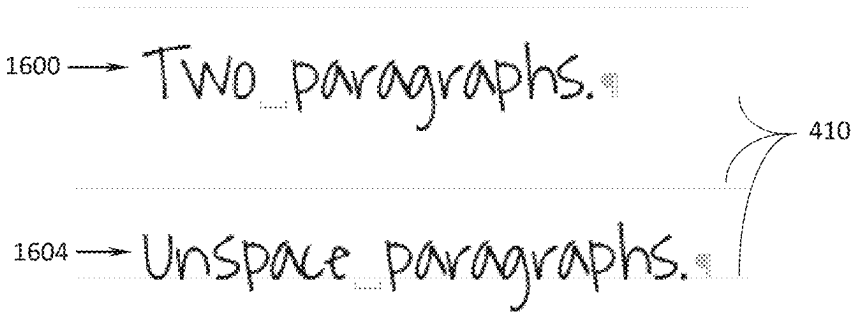


FIG. 16B

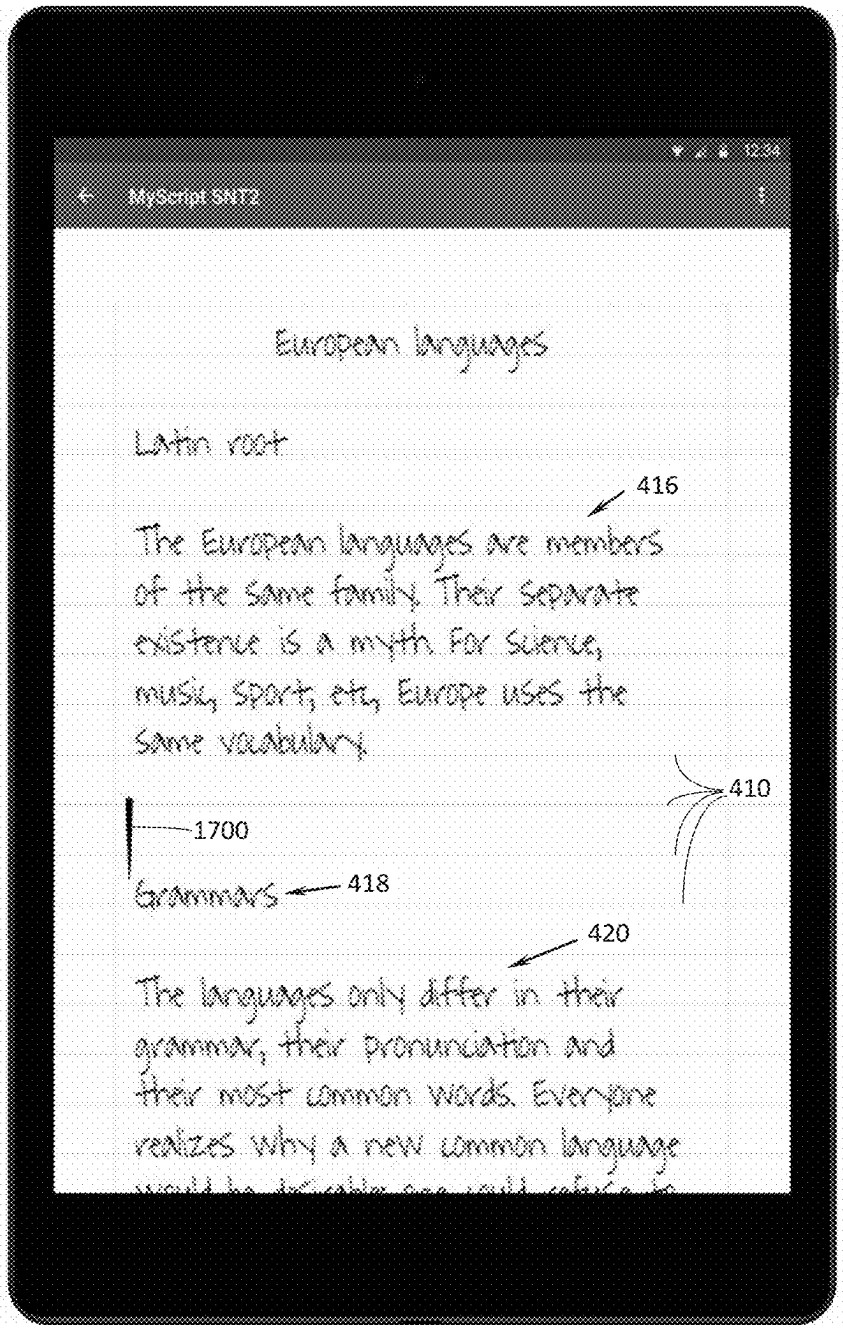

MyScript SNT2
European languages
Latin root
416
The European languages are members of the same family. Their separate existence is a myth. For science, music, sport, etc, Europe uses the same vocabulary.
410
1700
Grammars
418
420
The languages only differ in their grammar, their pronunciation and their most common words. Everyone realizes why a new common language

FIG. 17A

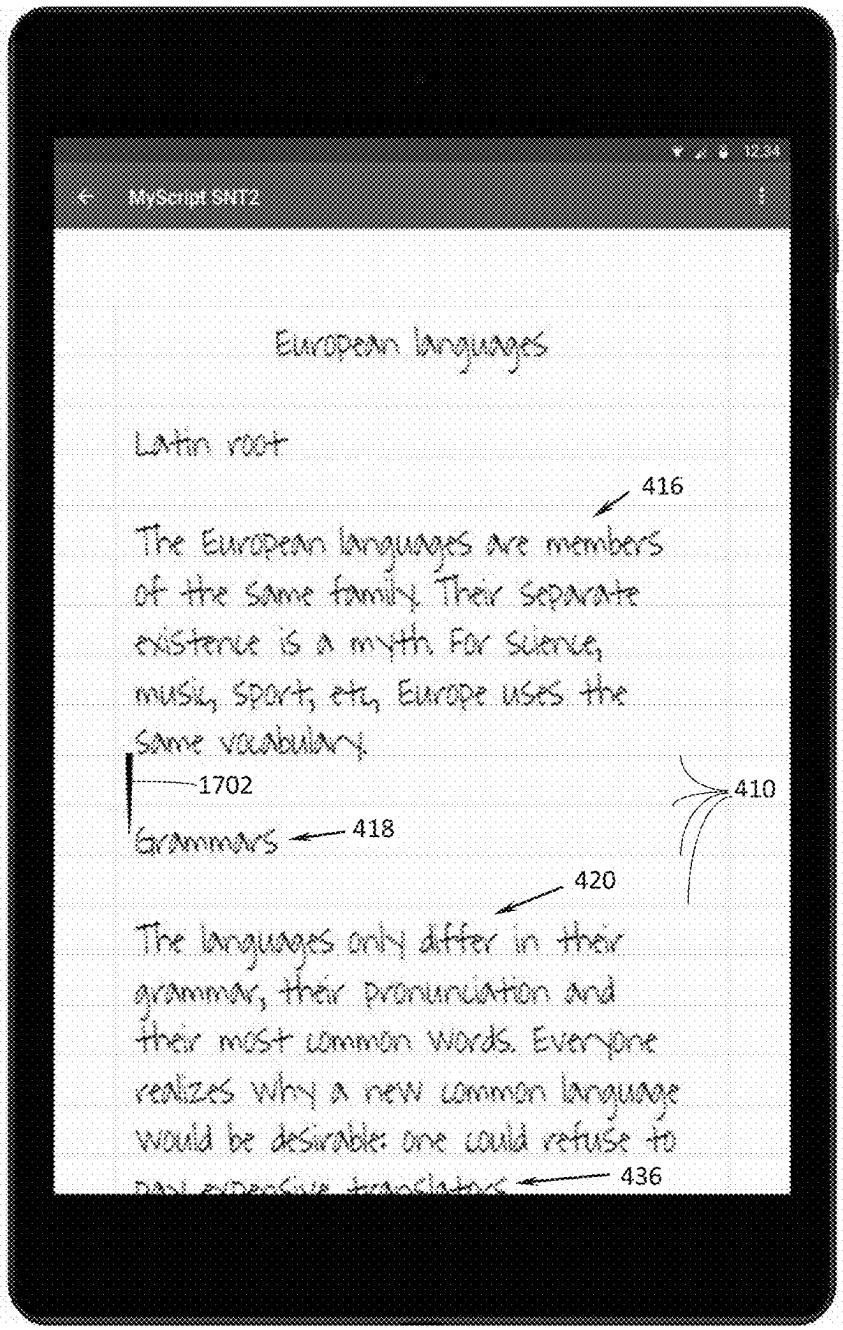

MyScript SMT2
European languages
Latin root
416
The European languages are members of the same family. Their separate existence is a myth. For science, music, sport, etc, Europe uses the same vocabulary.
1702
410
Grammars
418
420
The languages only differ in their grammar, their pronunciation and their most common words. Everyone realizes why a new common language would be desirable: one could refuse to
436

FIG. 17B

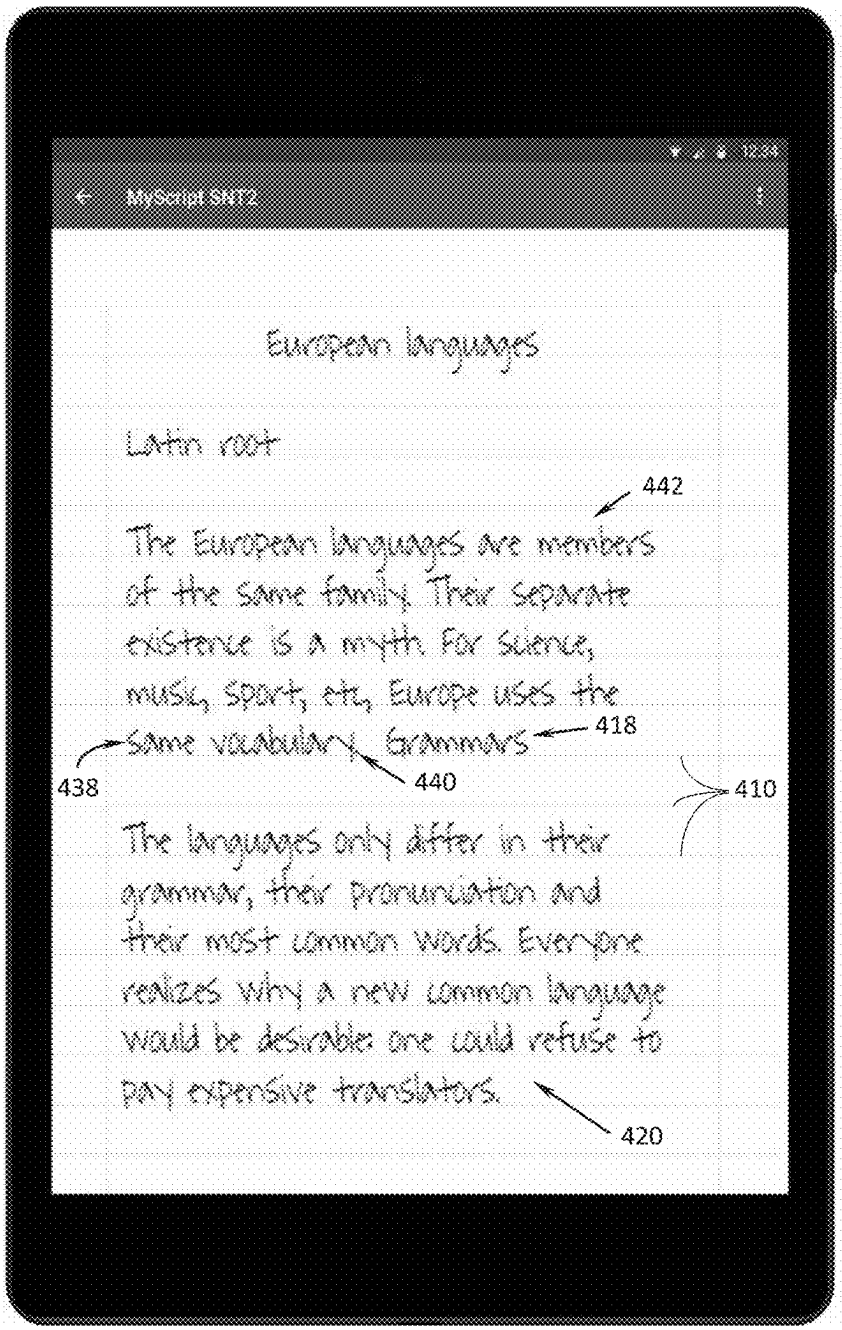

MyScript SMT2
European languages
Latin root
442
The European languages are members of the same family. Their separate existence is a myth. For science, music, sport, etc, Europe uses the same vocabulary. Grammars
418
438
440
410
The languages only differ in their grammar, their pronunciation and their most common words. Everyone realizes why a new common language would be desirable: one could refuse to pay expensive translators.
420

FIG. 17C

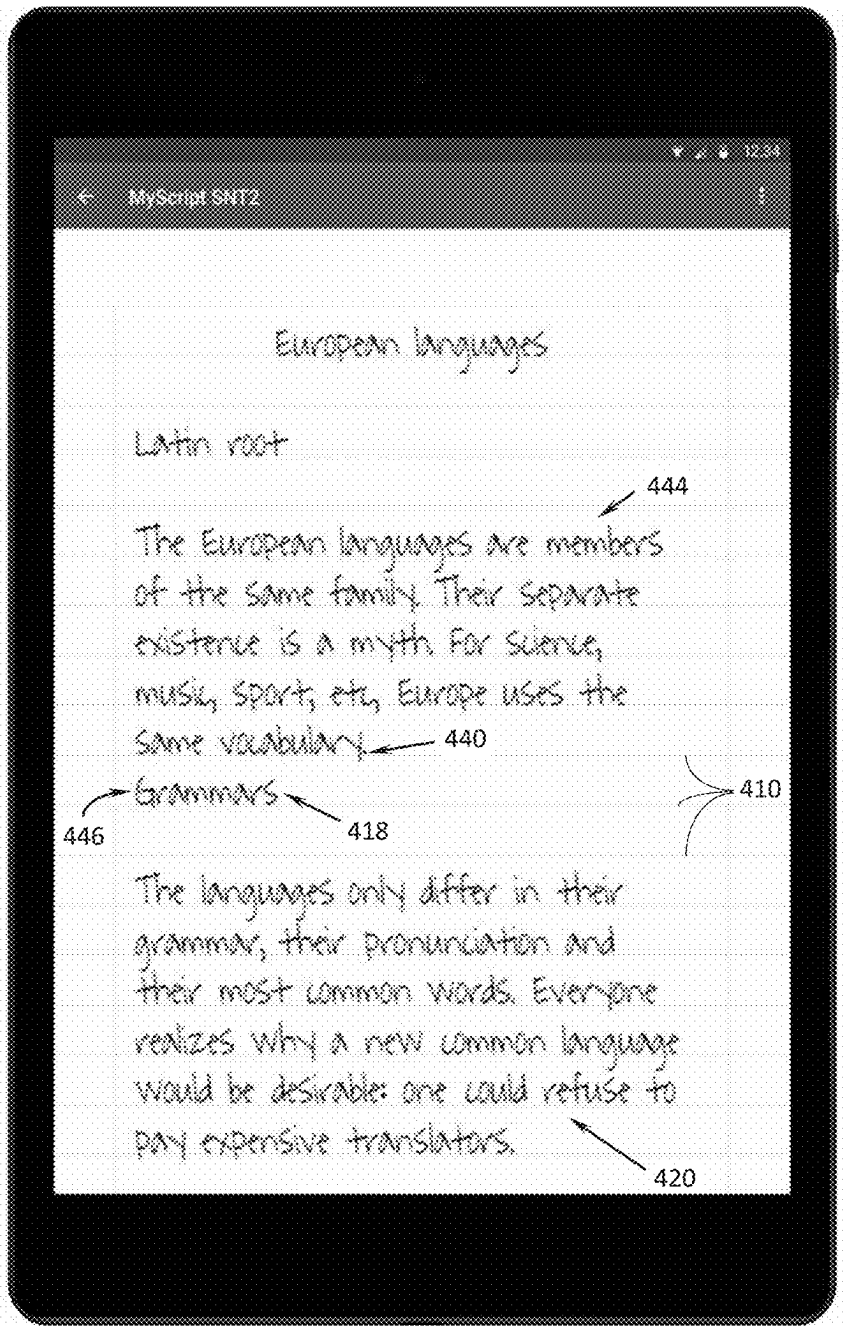

MyScript SMT2
European languages
Latin root
444
The European languages are members of the same family. Their separate existence is a myth. For Science, music, sport, etc, Europe uses the same vocabulary.
440
grammars
446
418
410
The languages only differ in their grammar, their pronunciation and their most common words. Everyone realizes why a new common language would be desirable: one could refuse to pay expensive translators.
420

FIG. 17D

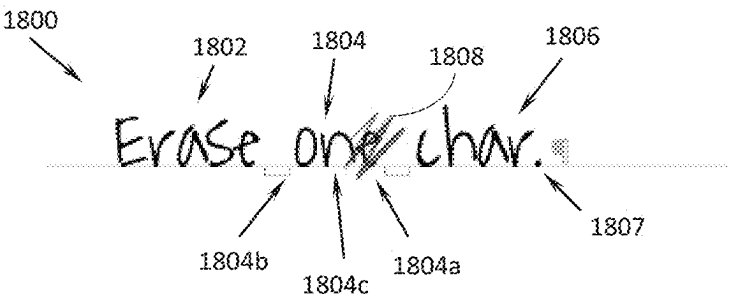


FIG. 18A

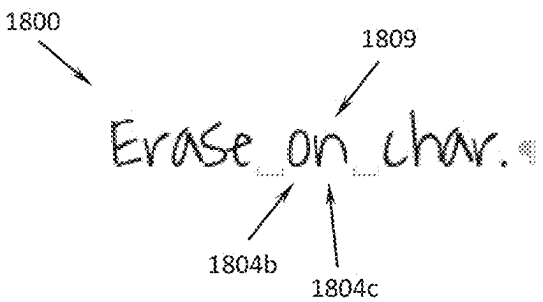


FIG. 18B

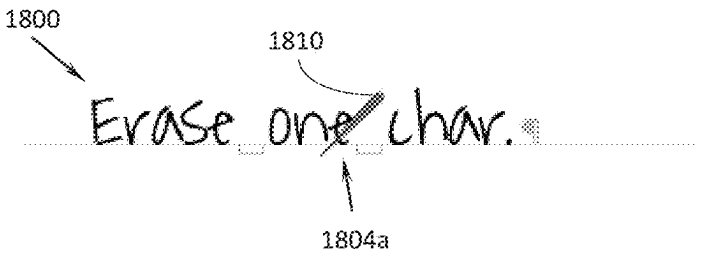


FIG. 18C

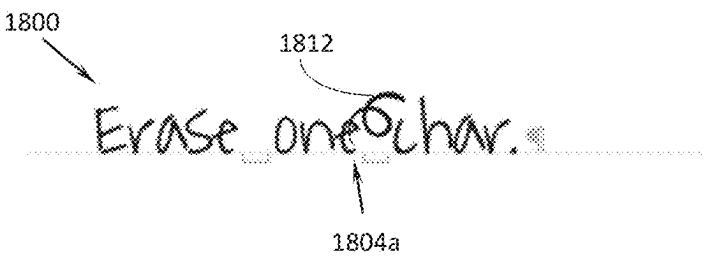


FIG. 18D

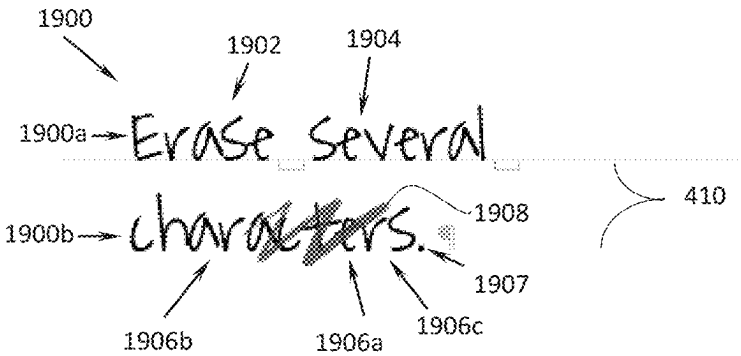


FIG. 19A

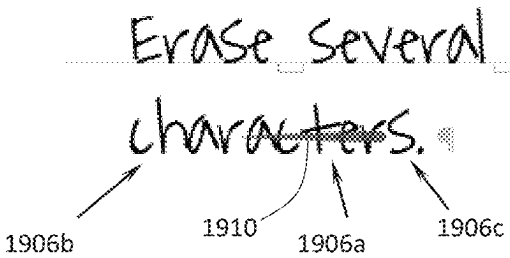


FIG. 19B

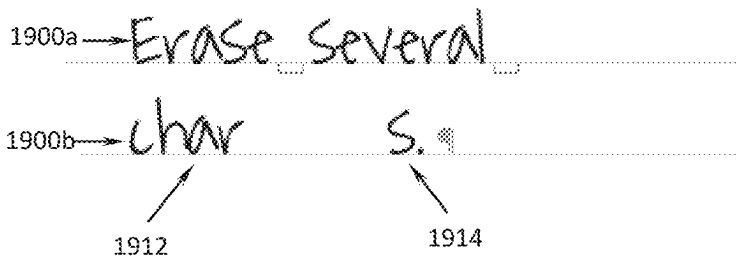


FIG. 19C

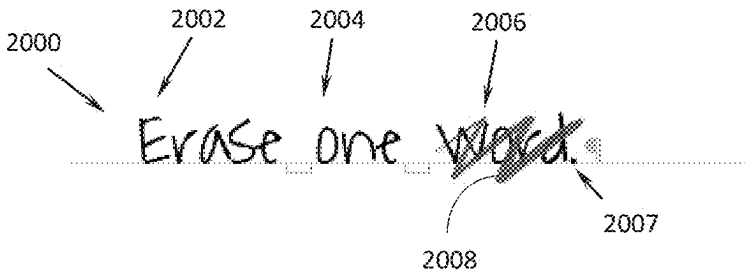


FIG. 20A

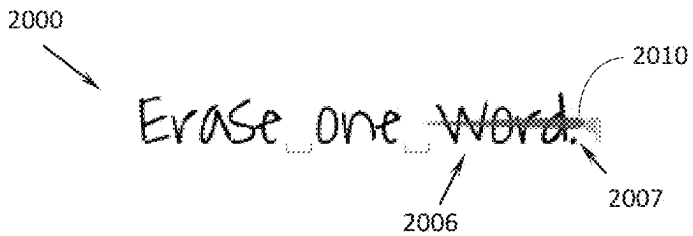


FIG. 20B



FIG. 20C

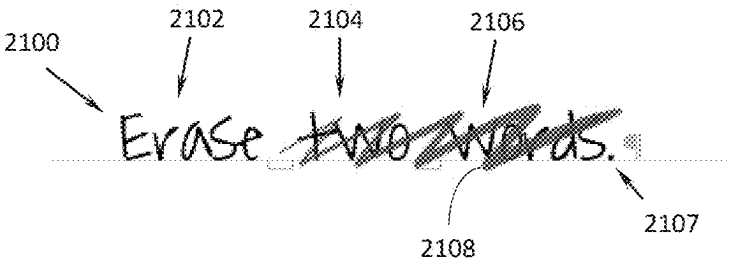


FIG. 21A

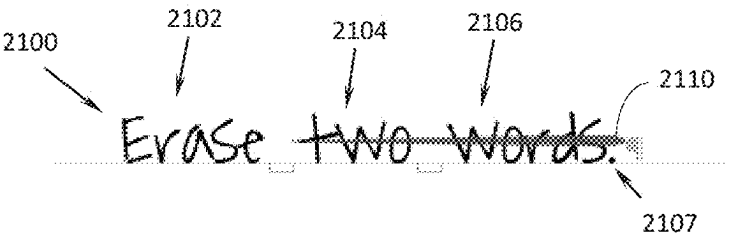


FIG. 21B

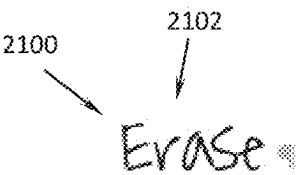


FIG. 21C

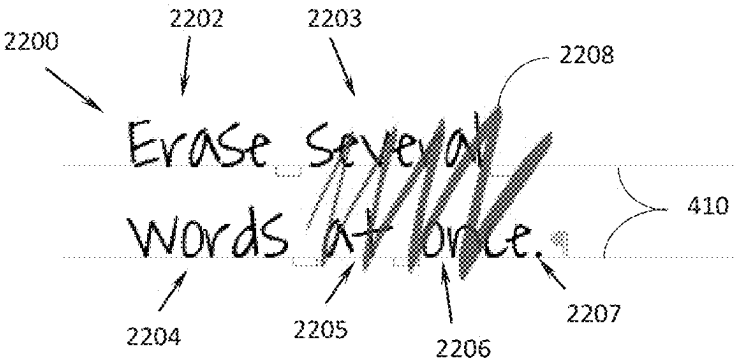


FIG. 22A

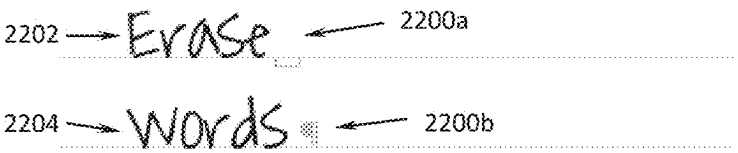


FIG. 22B

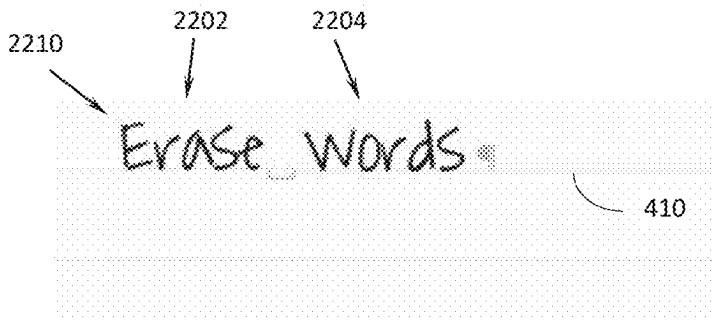


FIG. 22C

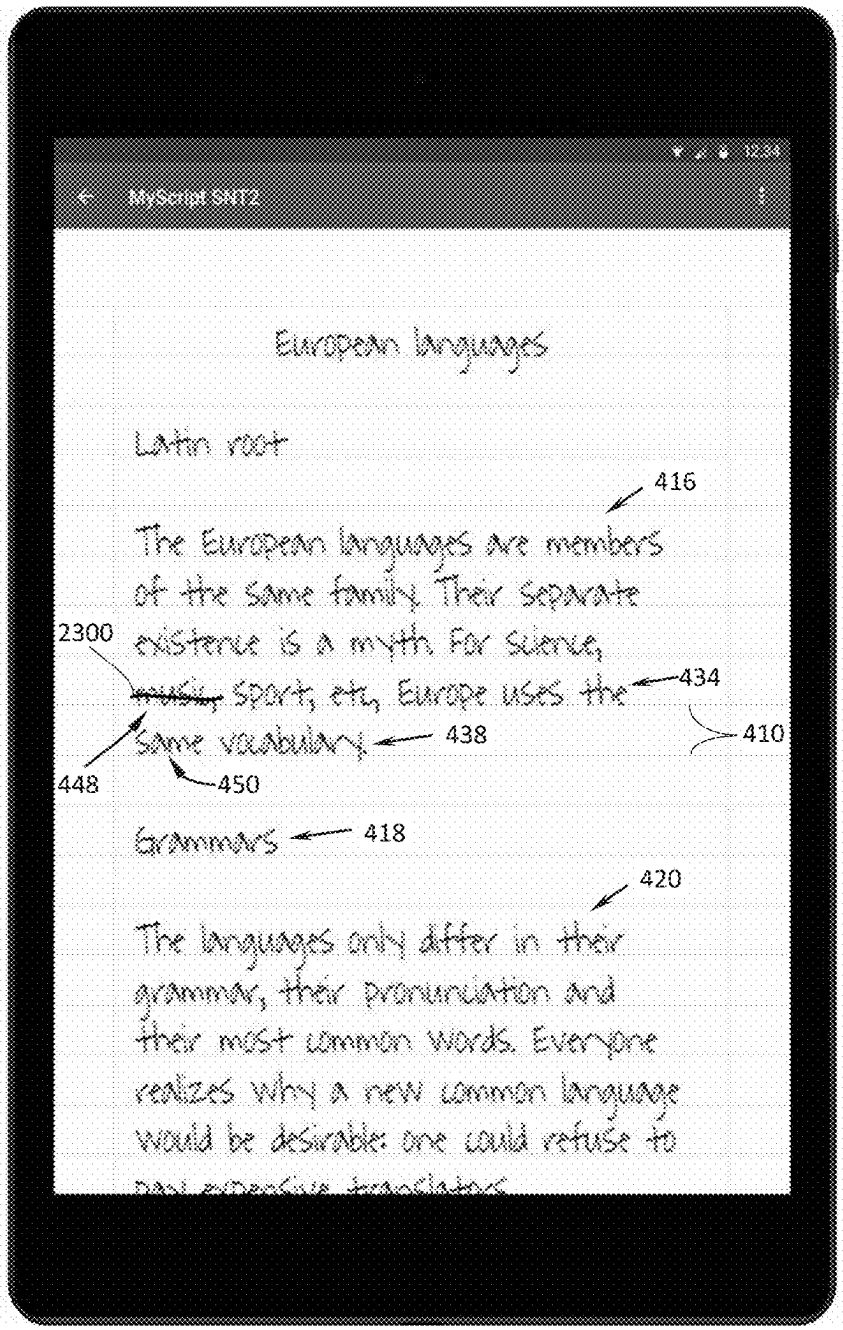

MyScript SMT2
European languages
Latin root
416
The European languages are members of the same family. Their separate existence is a myth. For science, music, sport, etc, Europe uses the same vocabulary.
2300
434
438
410
448
450
Grammars
418
420
The languages only differ in their grammar, their pronunciation and their most common words. Everyone realizes why a new common language would be desirable: one could refuse to

FIG. 23A

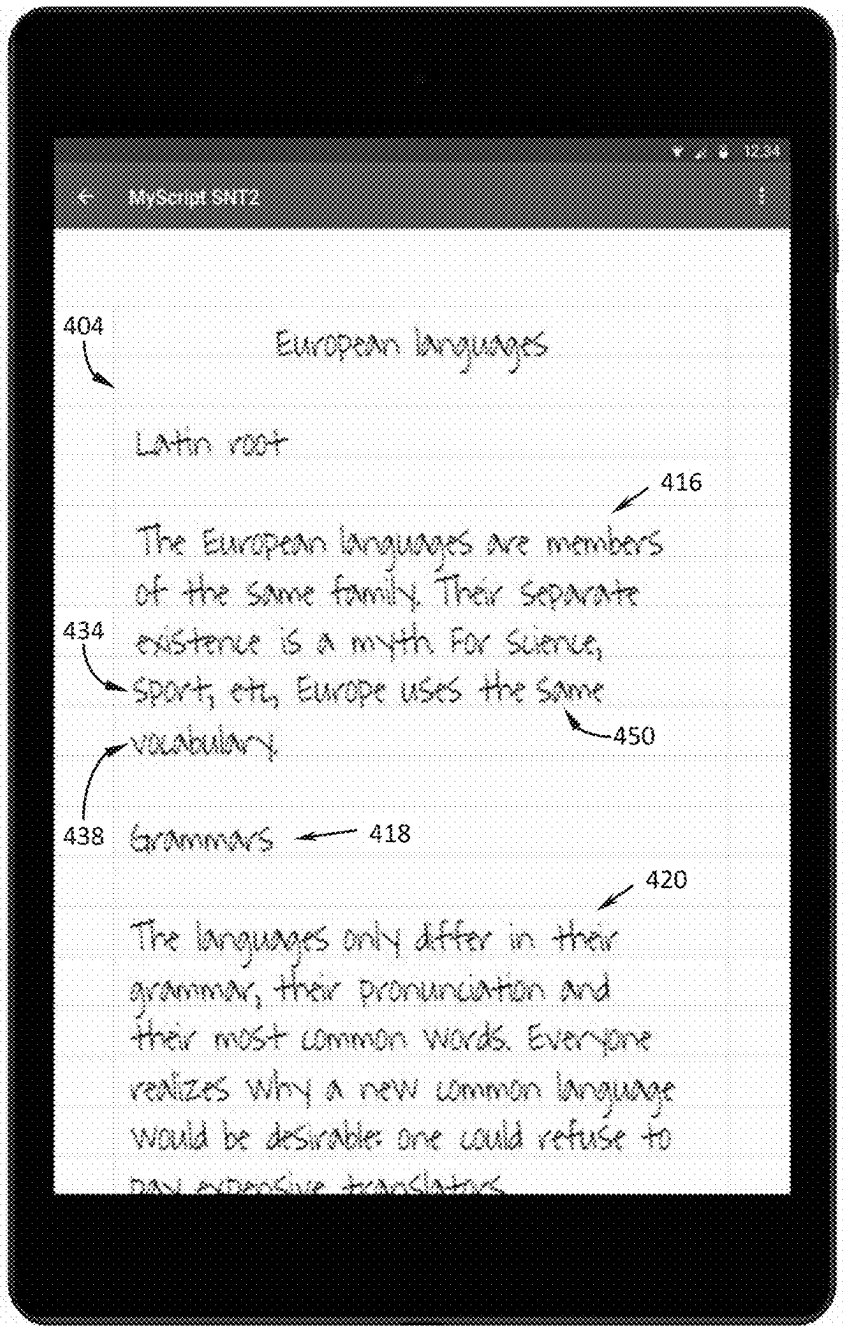

MyScript SMT2
404
European languages
Latin root
416
The European languages are members of the same family. Their separate existence is a myth. For science, sport, etc, Europe uses the same vocabulary.
434
450
438
Grammars
418
420
The languages only differ in their grammar, their pronunciation and their most common words. Everyone realizes why a new common language would be desirable: one could refuse to

FIG. 23B

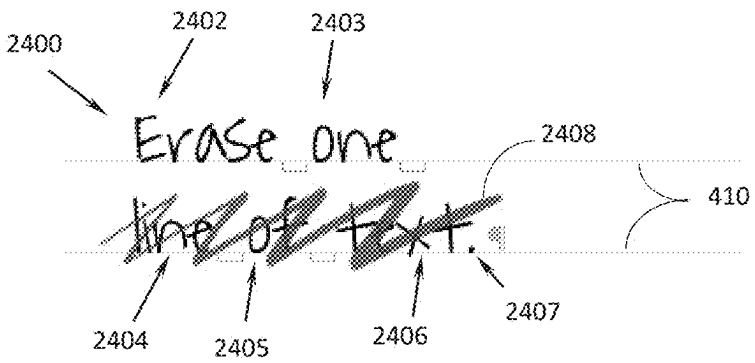


FIG. 24A

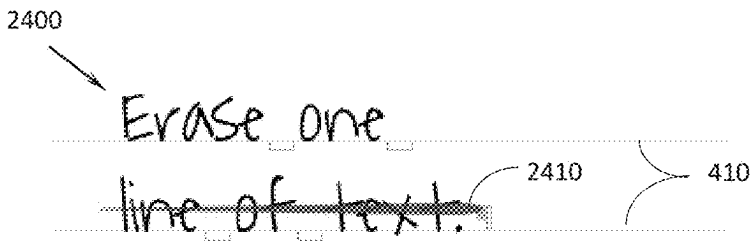


FIG. 24B

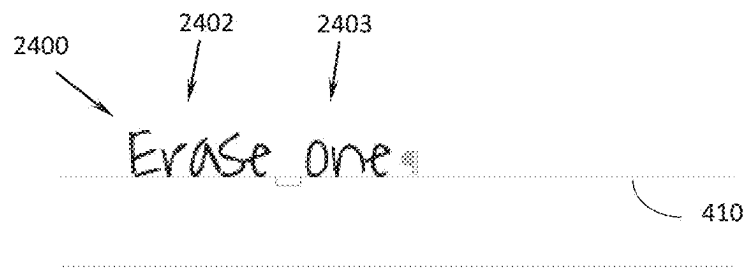


FIG. 24C

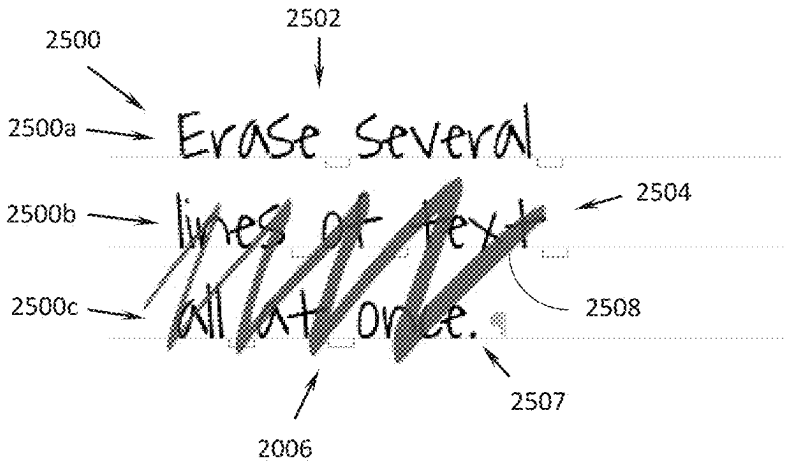


FIG. 25A

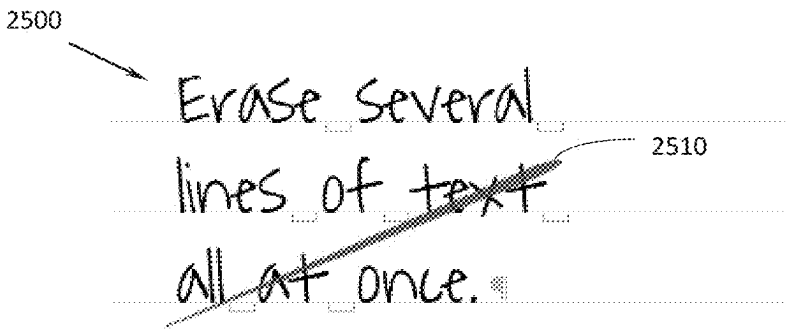


FIG. 25B

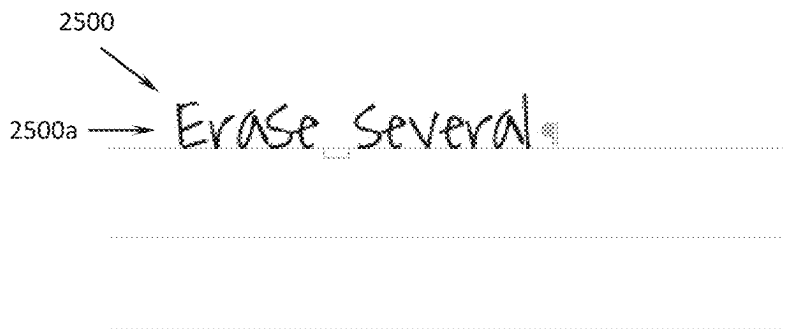


FIG. 25C

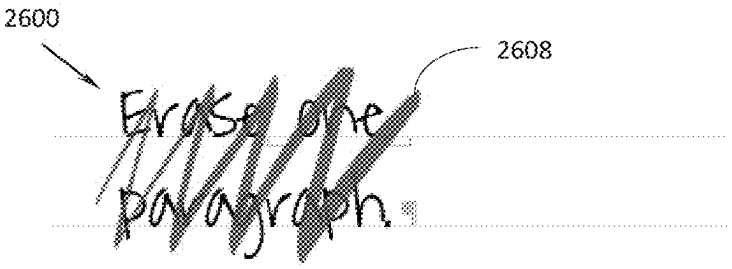


FIG. 26A

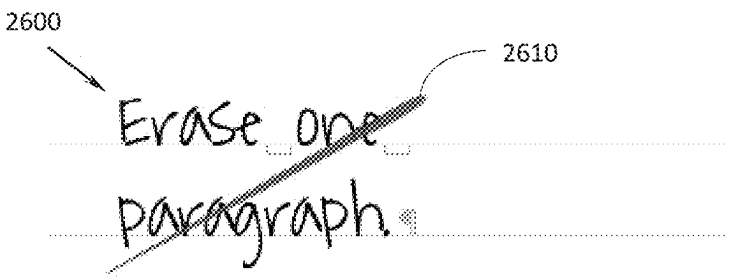


FIG. 26B

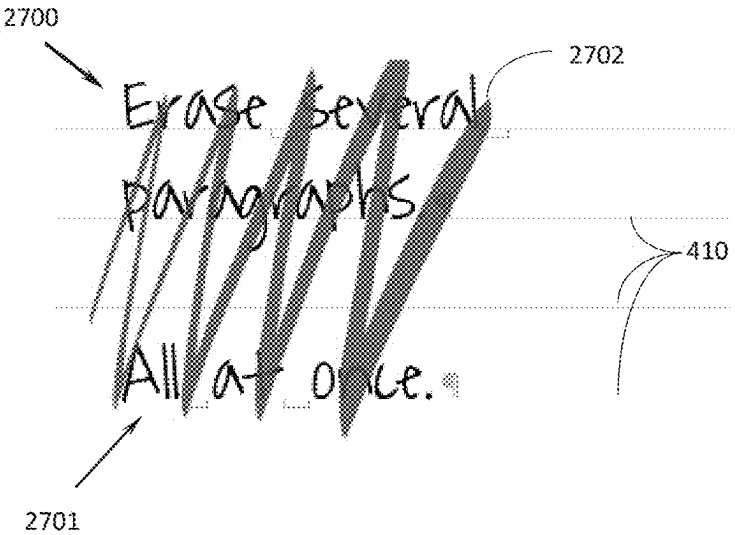


FIG. 27A

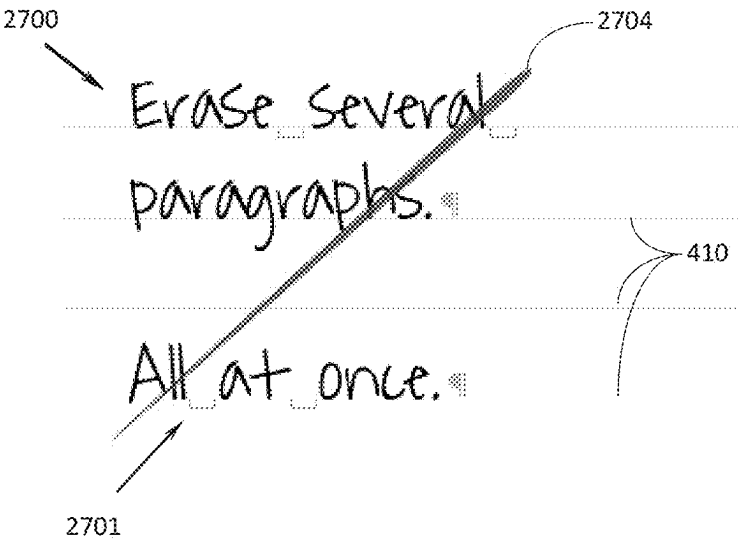


FIG. 27B

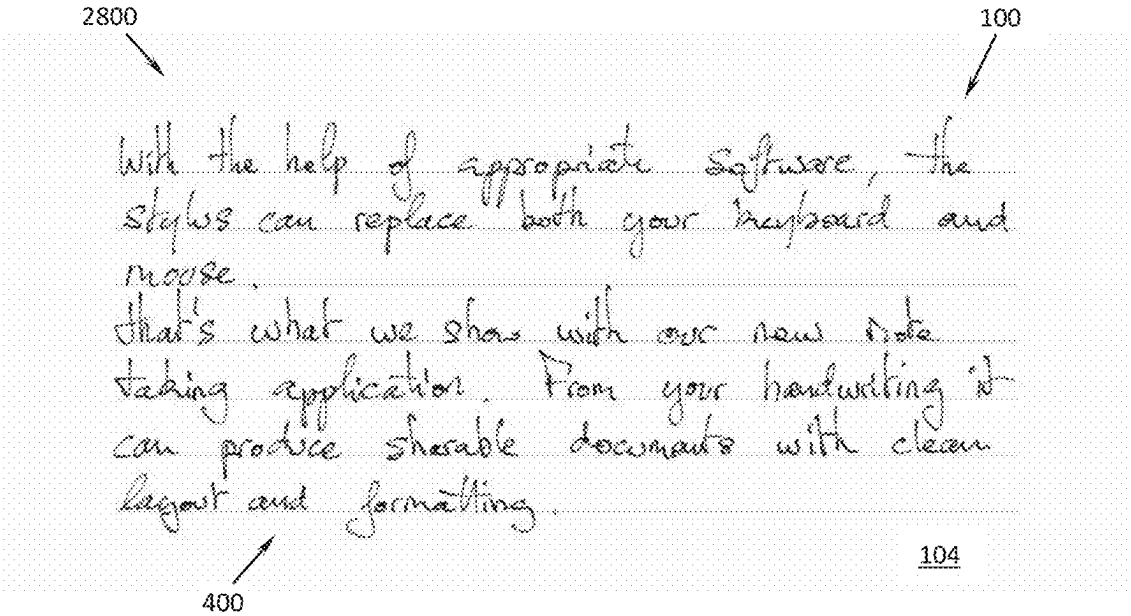

2800
100
With the help of appropriate Software, the
Stylus can replace both your keyboard and
mouse.
That's what we show with our new Note
taking application. From your handwriting it
can produce shareable documents with clean
layout and formatting.
104
400

FIG. 28

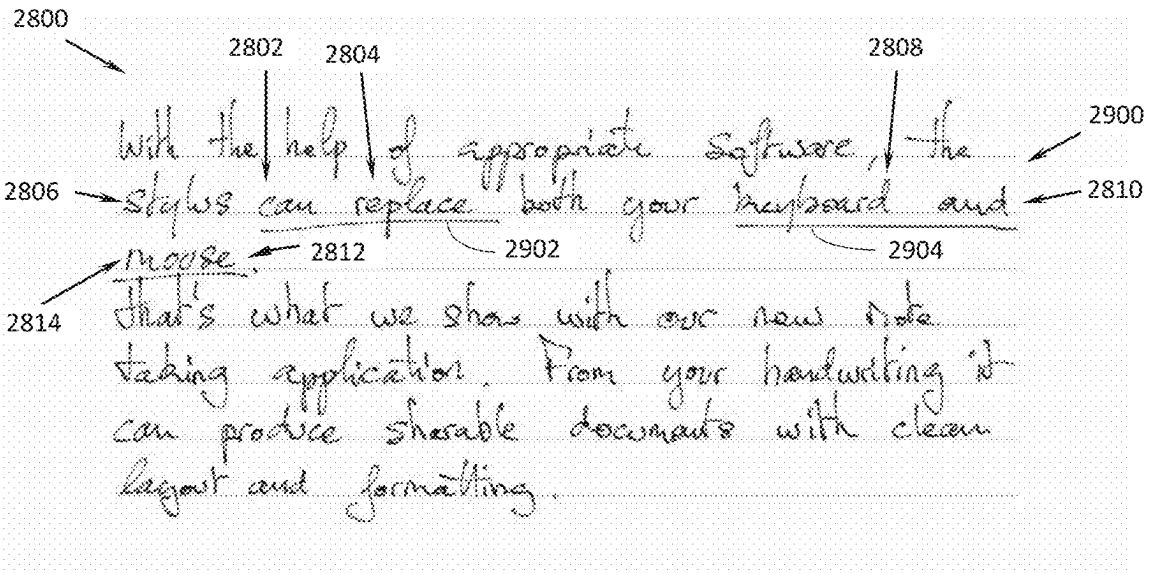

2800
2802
2804
2808
2900
2806
2810
With the help of appropriate Software, the
Stylus can replace both your keyboard and
mouse.
2812
2902
2904
2814
That's what we show with our new Note
taking application. From your handwriting it
can produce shareable documents with clean
layout and formatting.

FIG. 29

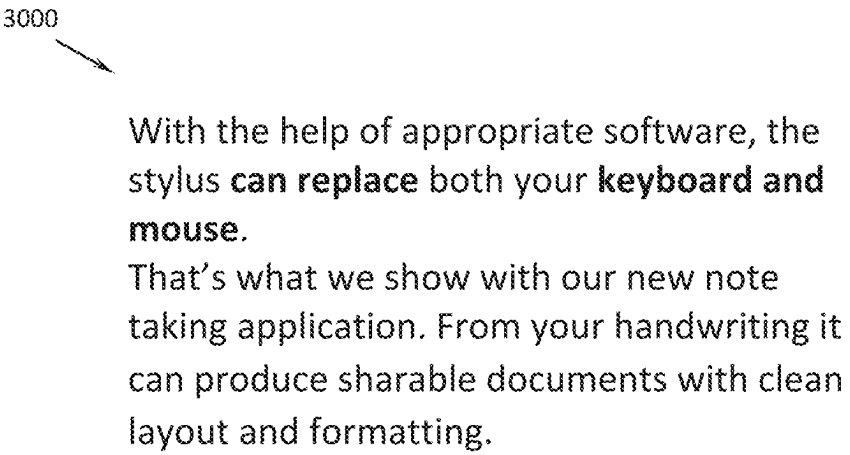

3000
With the help of appropriate software, the stylus **can replace** both your **keyboard and mouse**.
That's what we show with our new note taking application. From your handwriting it can produce sharable documents with clean layout and formatting.

FIG. 30

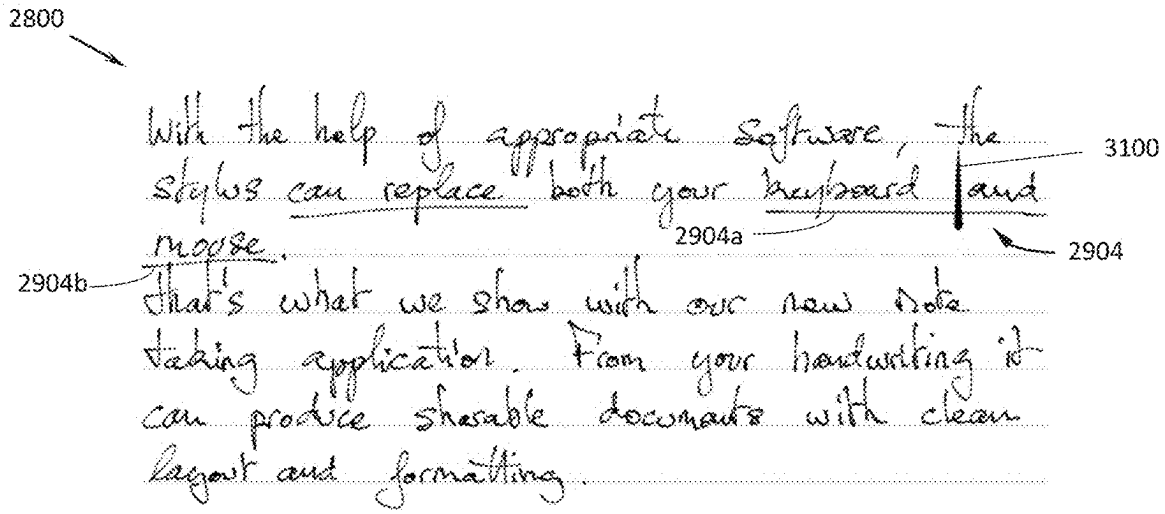


FIG. 31A

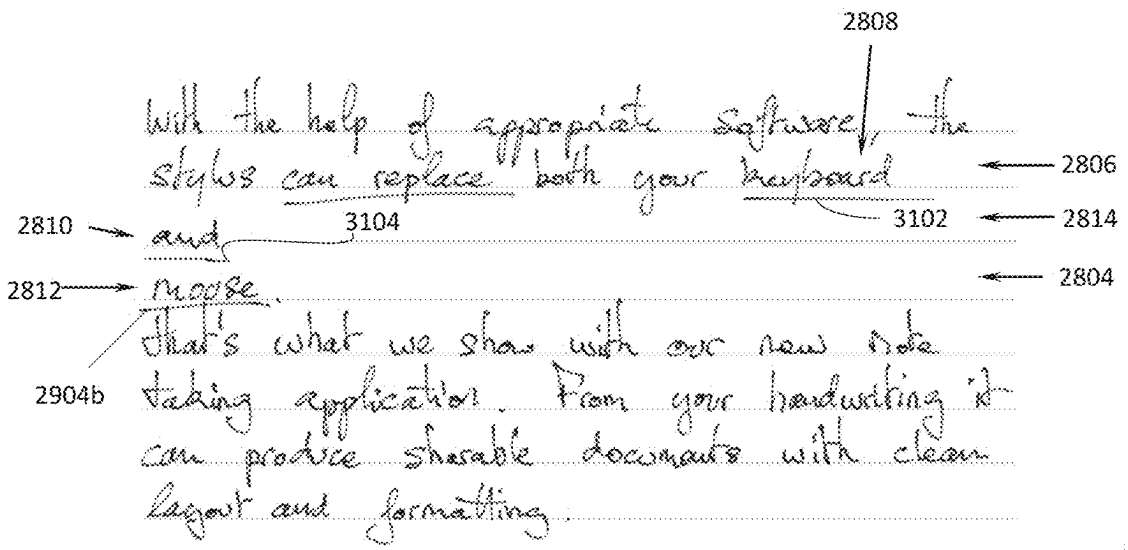


FIG. 31B

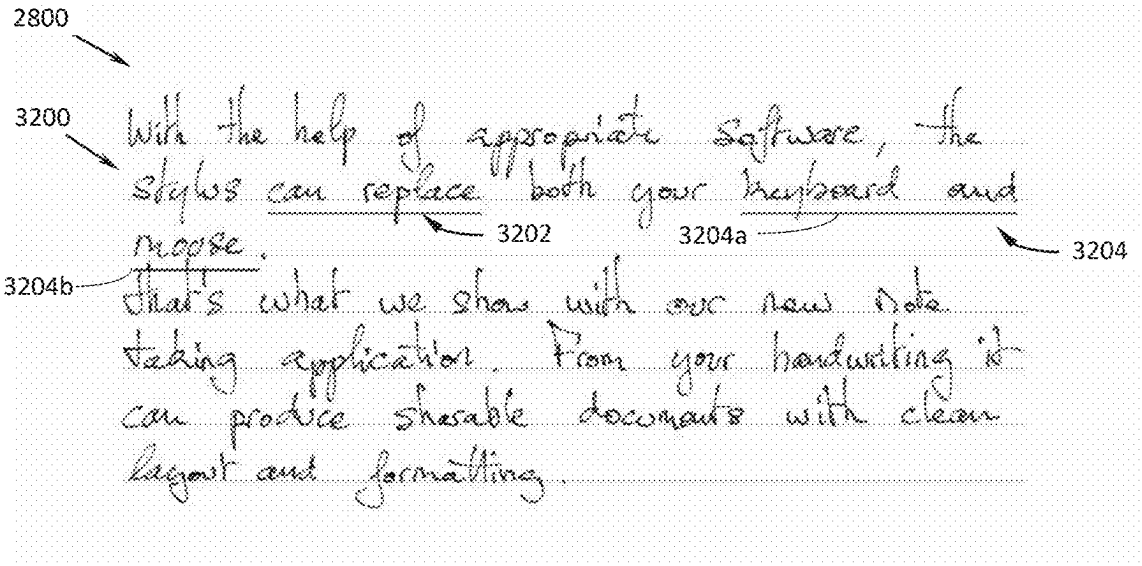


FIG. 32

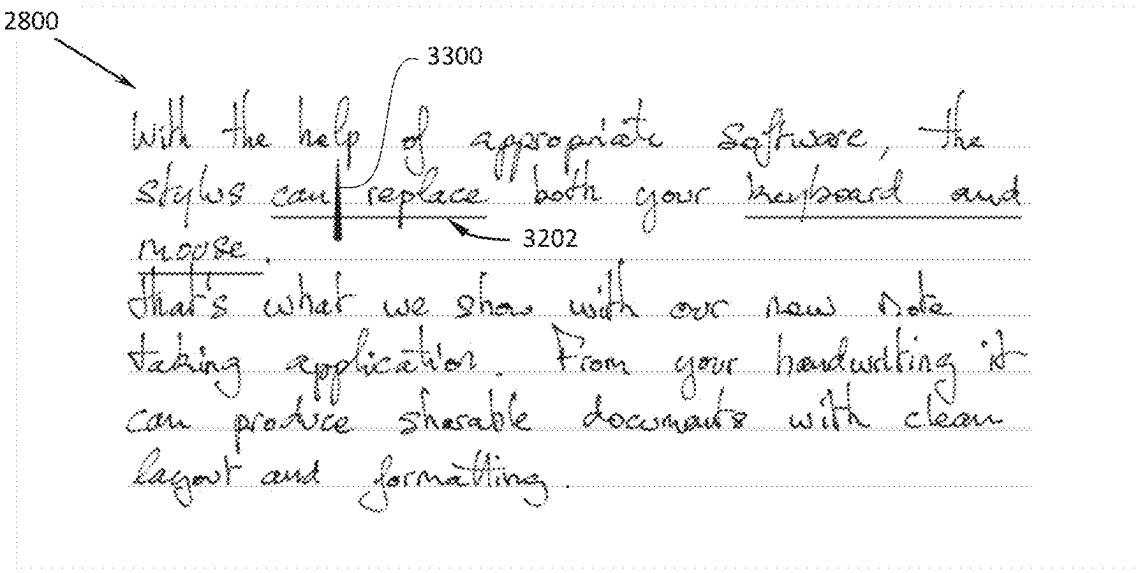


FIG. 33A

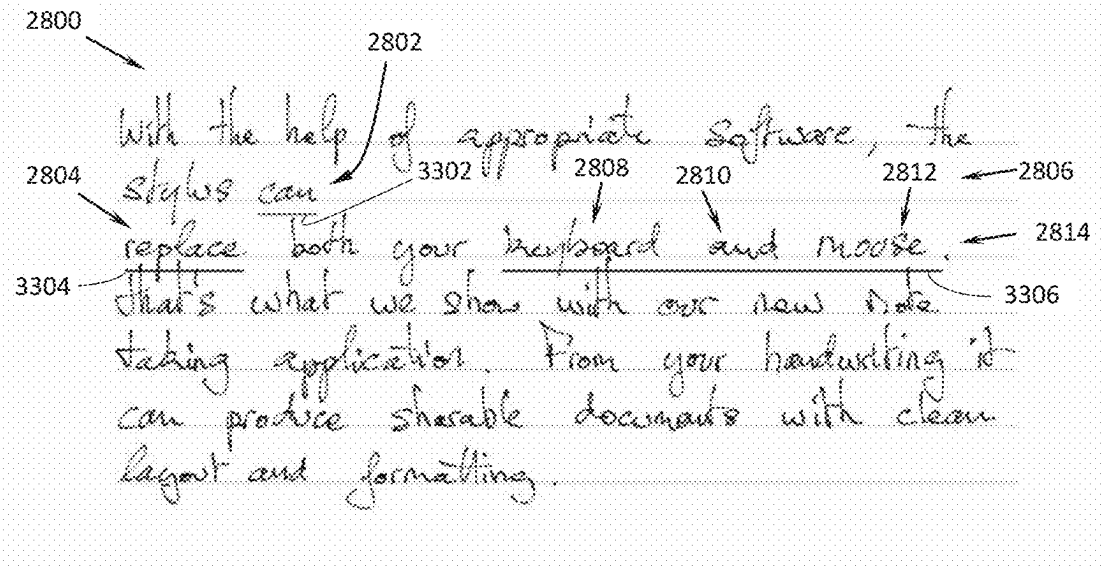


FIG. 33B

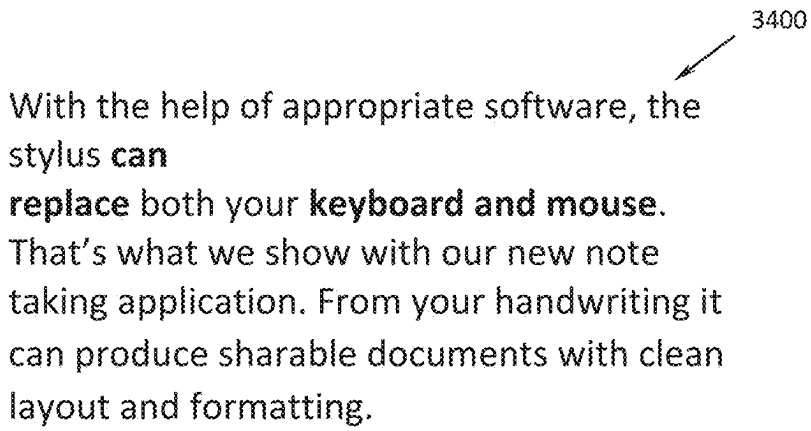


FIG. 34

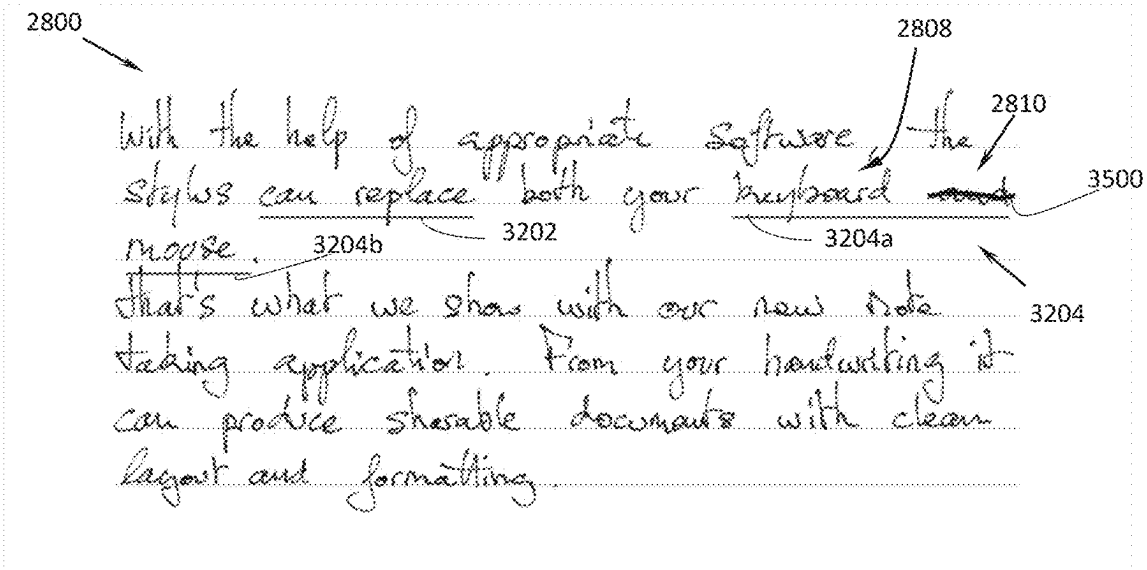


FIG. 35A

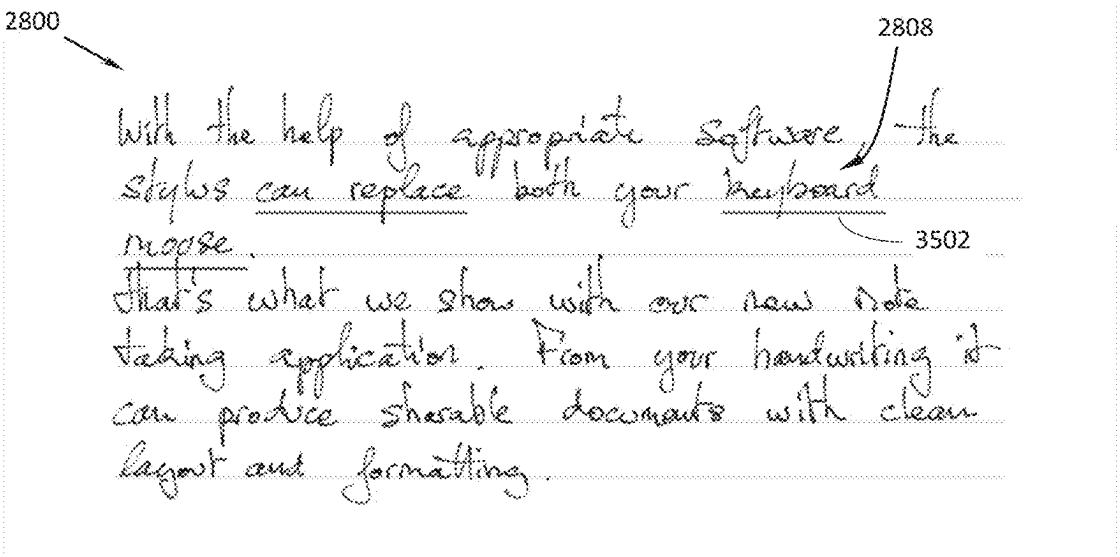


FIG. 35B

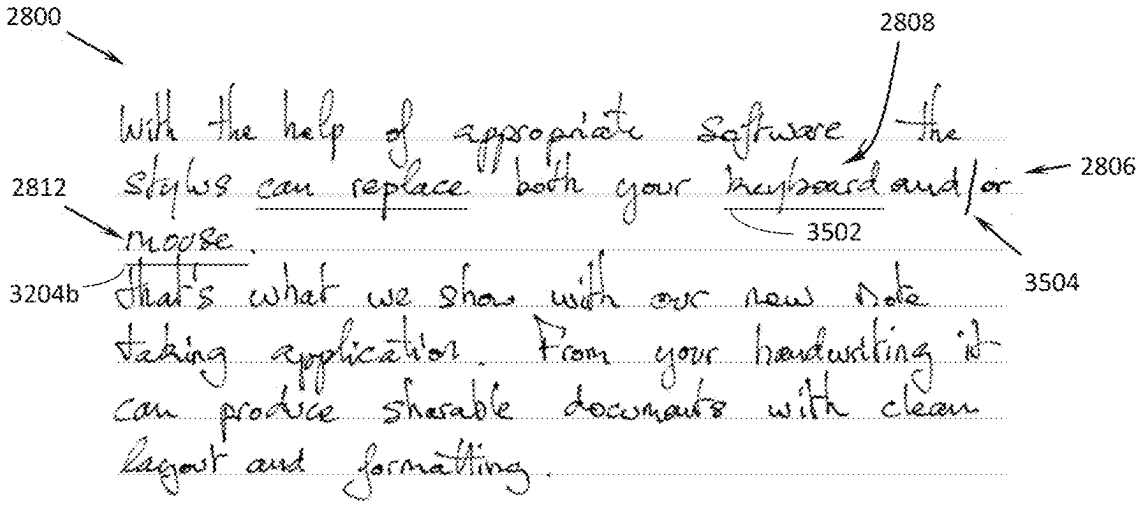


FIG. 35C

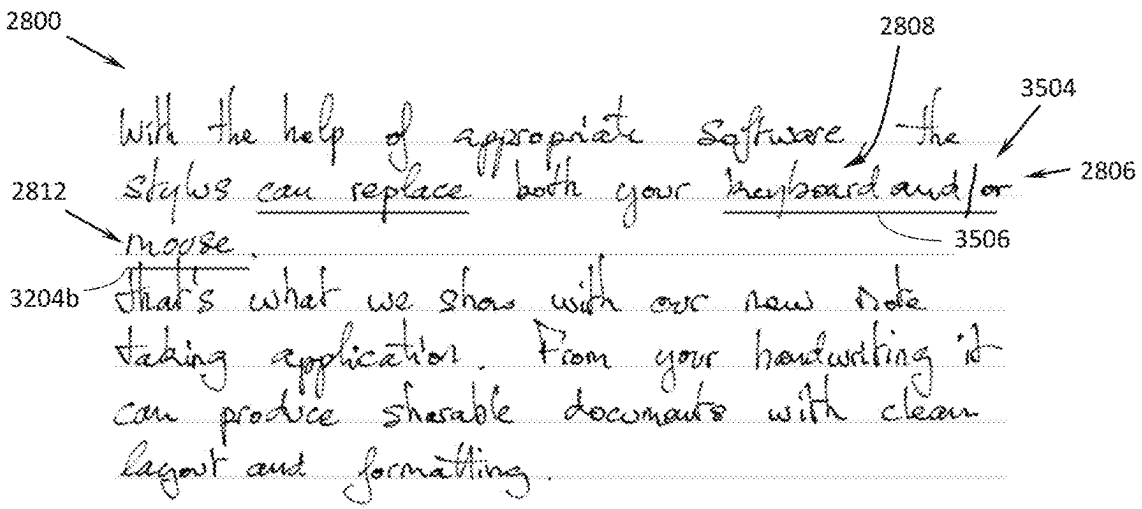


FIG. 35D

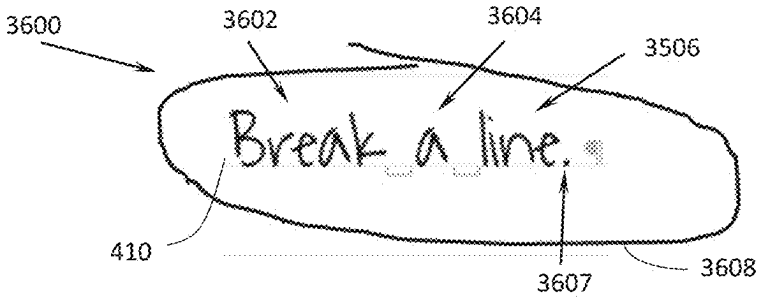


FIG. 36

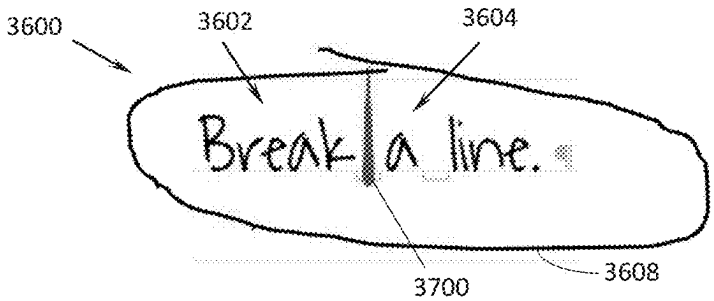


FIG. 37A

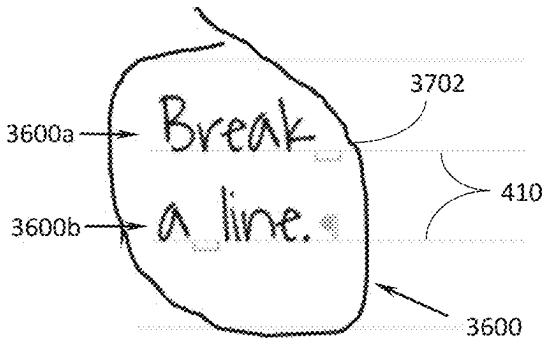


FIG. 37B

BREAK

A LINE

FIG. 38

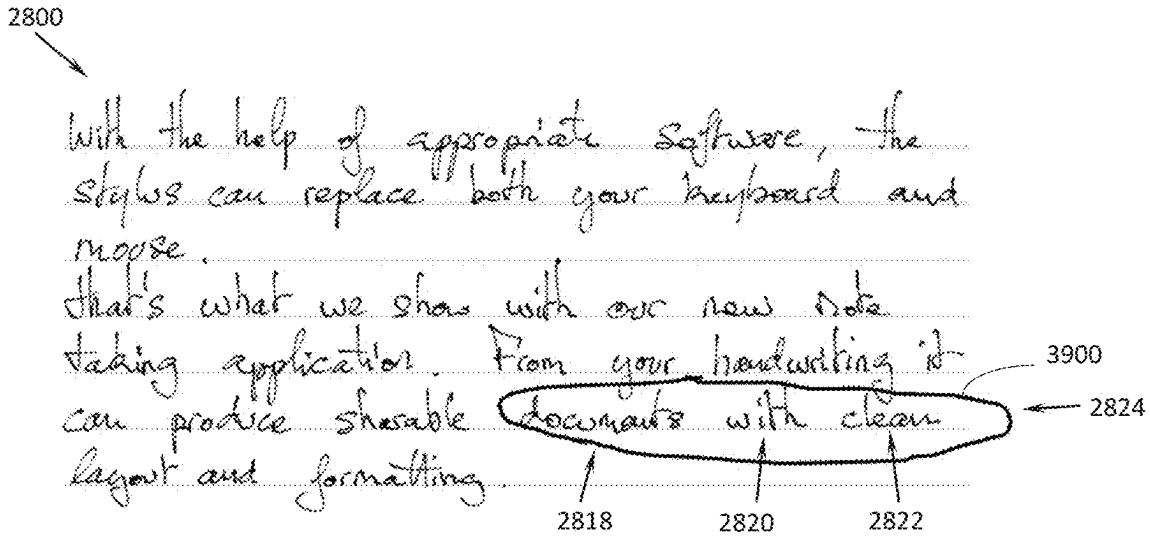


FIG. 39

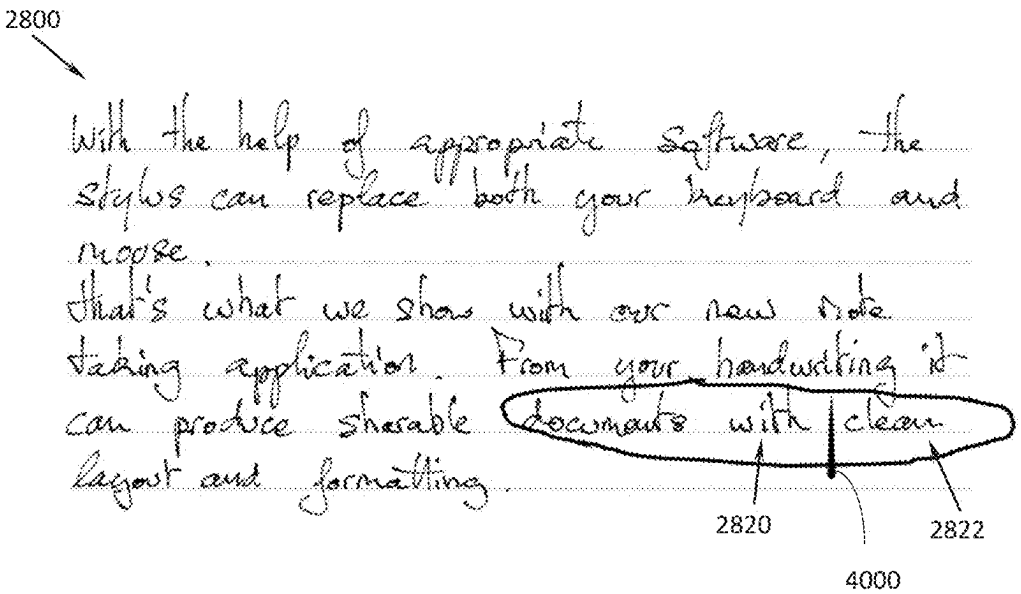


FIG. 40A

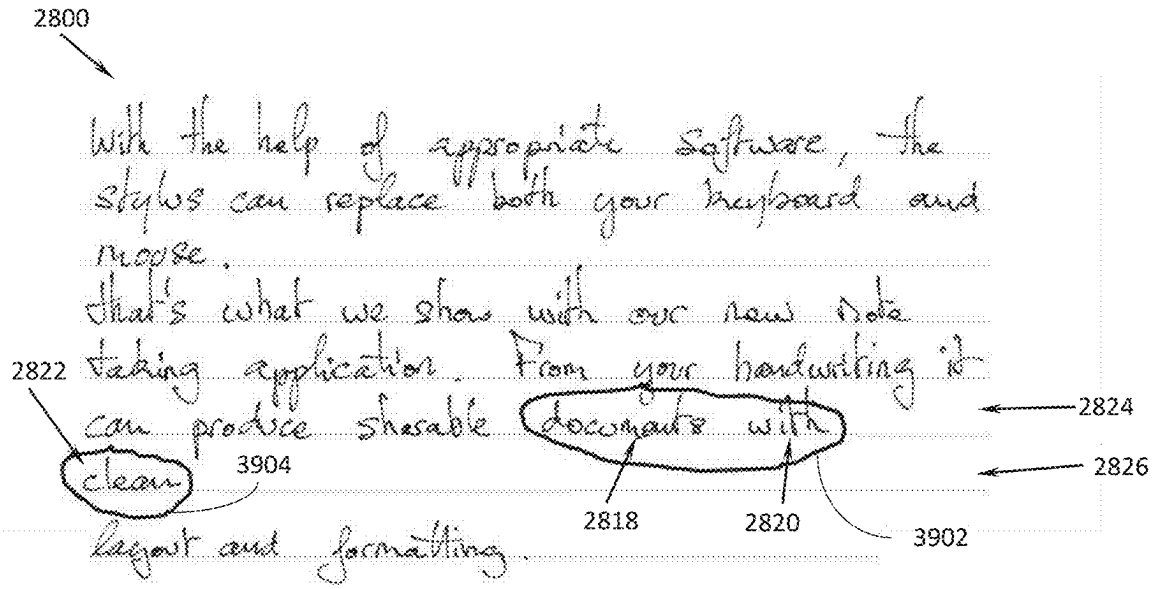


FIG. 40B

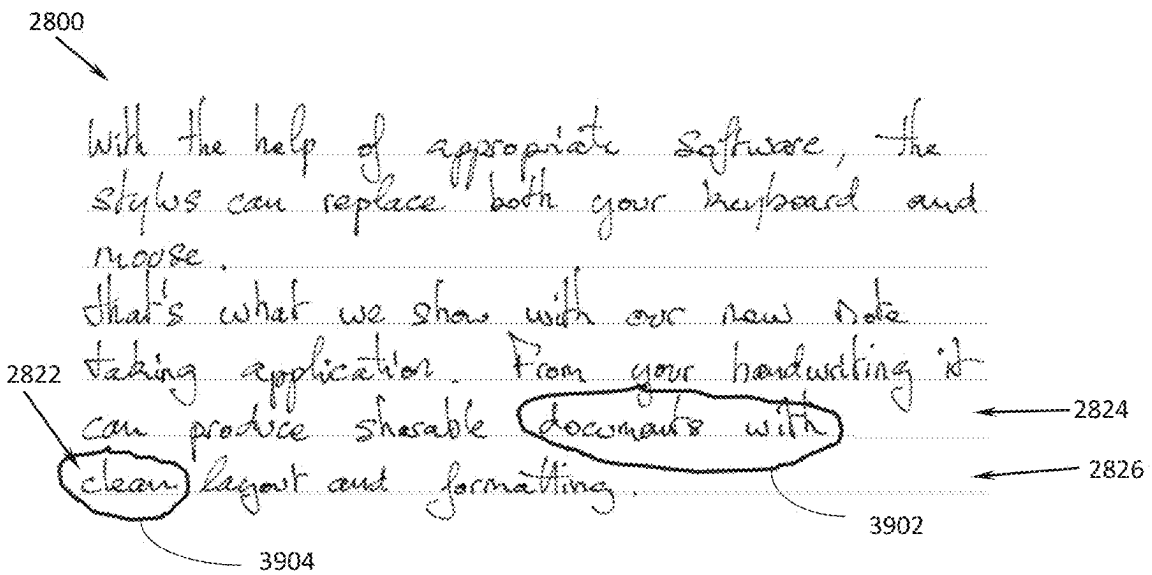


FIG. 40C

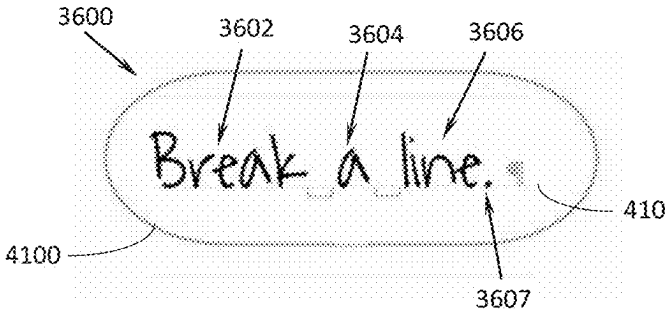


FIG. 41

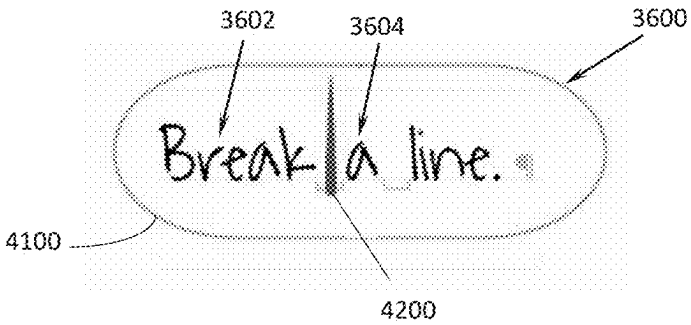


FIG. 42A

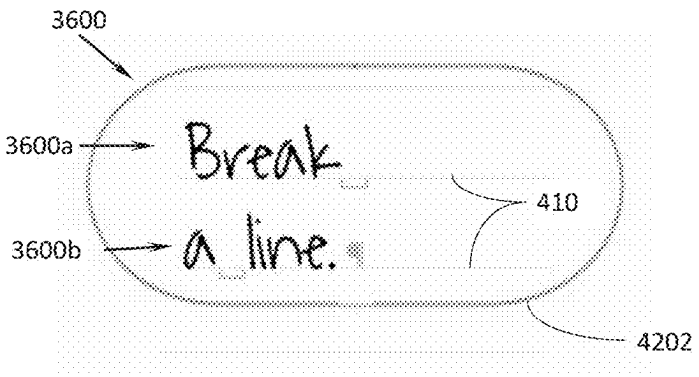


FIG. 42B

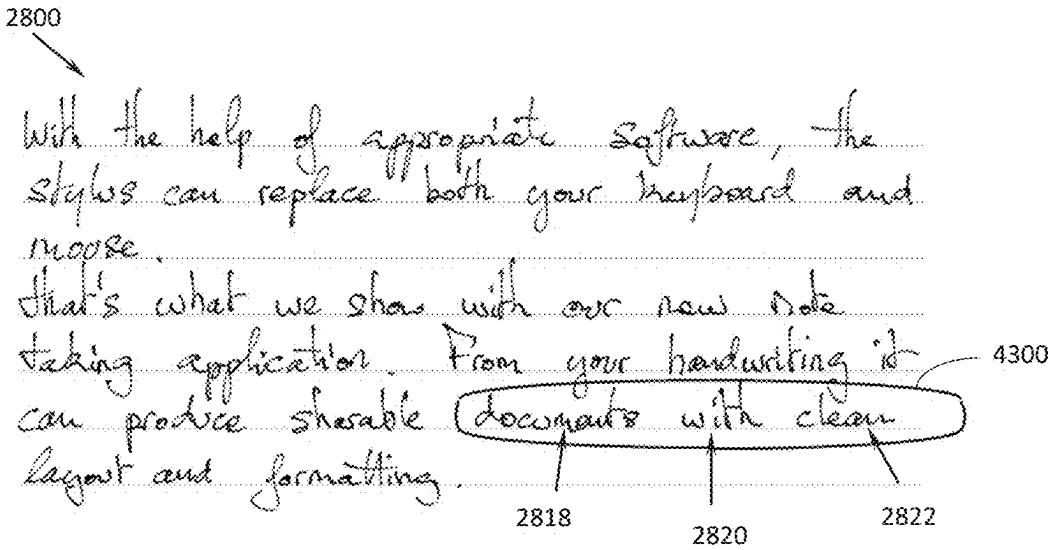

2800
With the help of appropriate software, the stylus can replace both your keyboard and mouse.
That's what we show with our new note taking application. From your handwriting it can produce shareable documents with clean layout and formatting.
4300
2818
2820
2822

FIG. 43

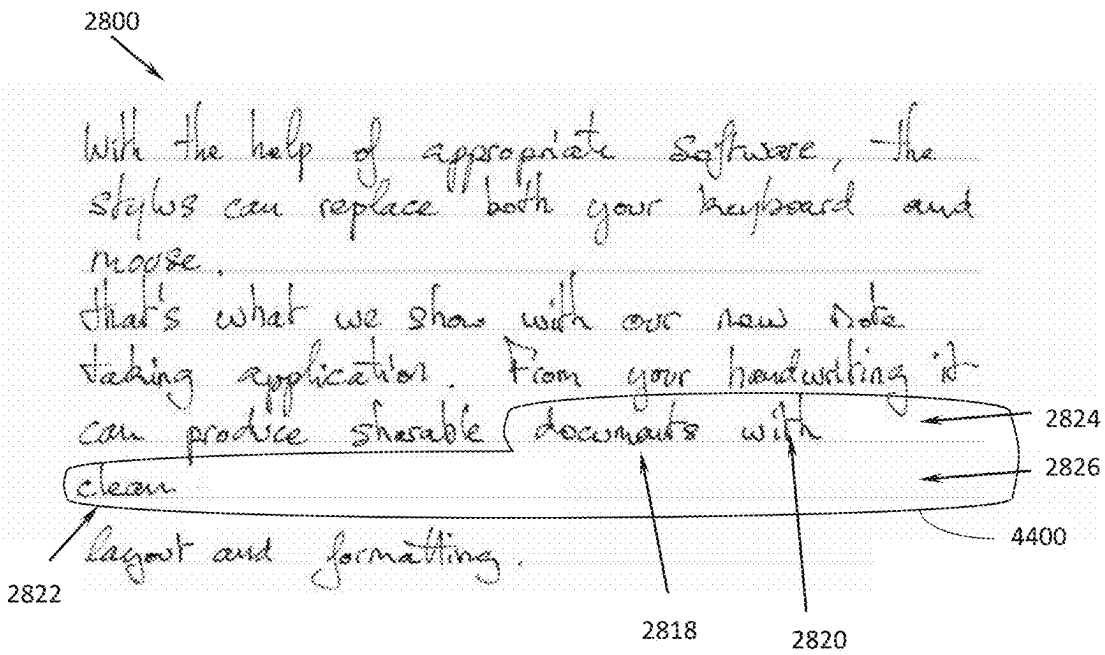

2800
With the help of appropriate software, the stylus can replace both your keyboard and mouse.
That's what we show with our new note taking application. From your handwriting it can produce shareable documents with clean layout and formatting.
2824
2826
4400
2822
2818
2820

FIG. 44

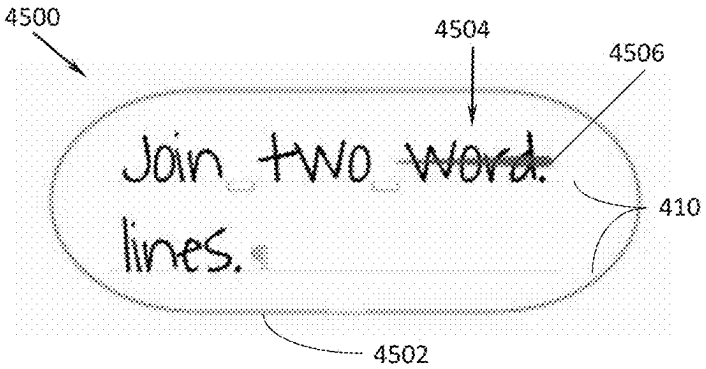


FIG. 45A

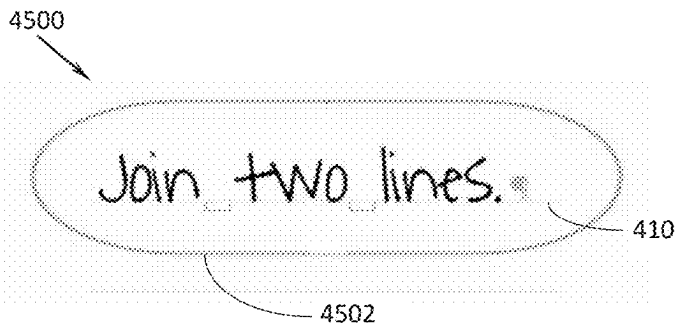


FIG. 45B

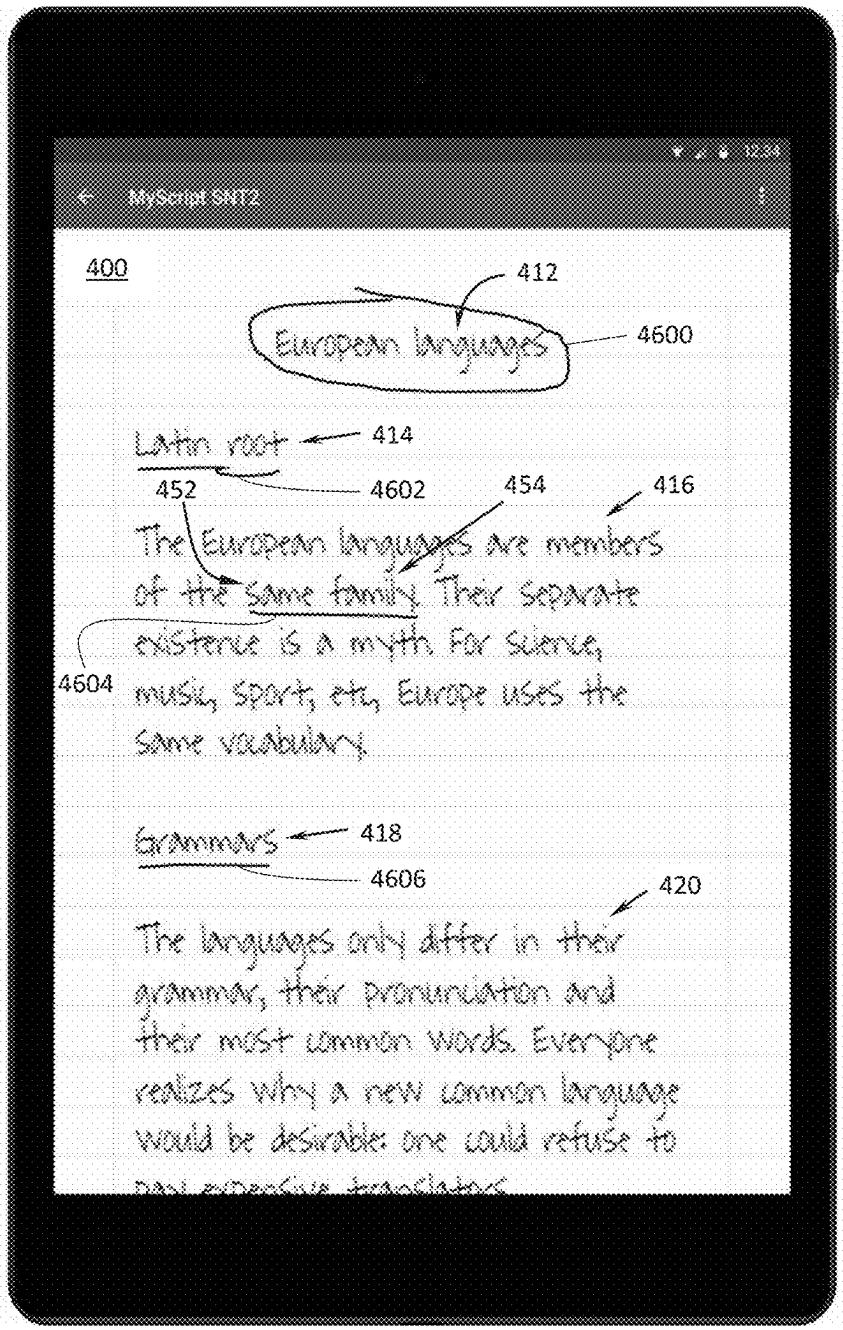

MyScript SNT2
400
412
European languages
4600
Latin root
414
452
4602
454
416
The European languages are members of the same family. Their separate existence is a myth. For science, music, sport, etc, Europe uses the same vocabulary.
4604
Grammars
418
4606
420
The languages only differ in their grammar, their pronunciation and their most common words. Everyone realizes why a new common language would be desirable: one could refuse to

FIG. 46

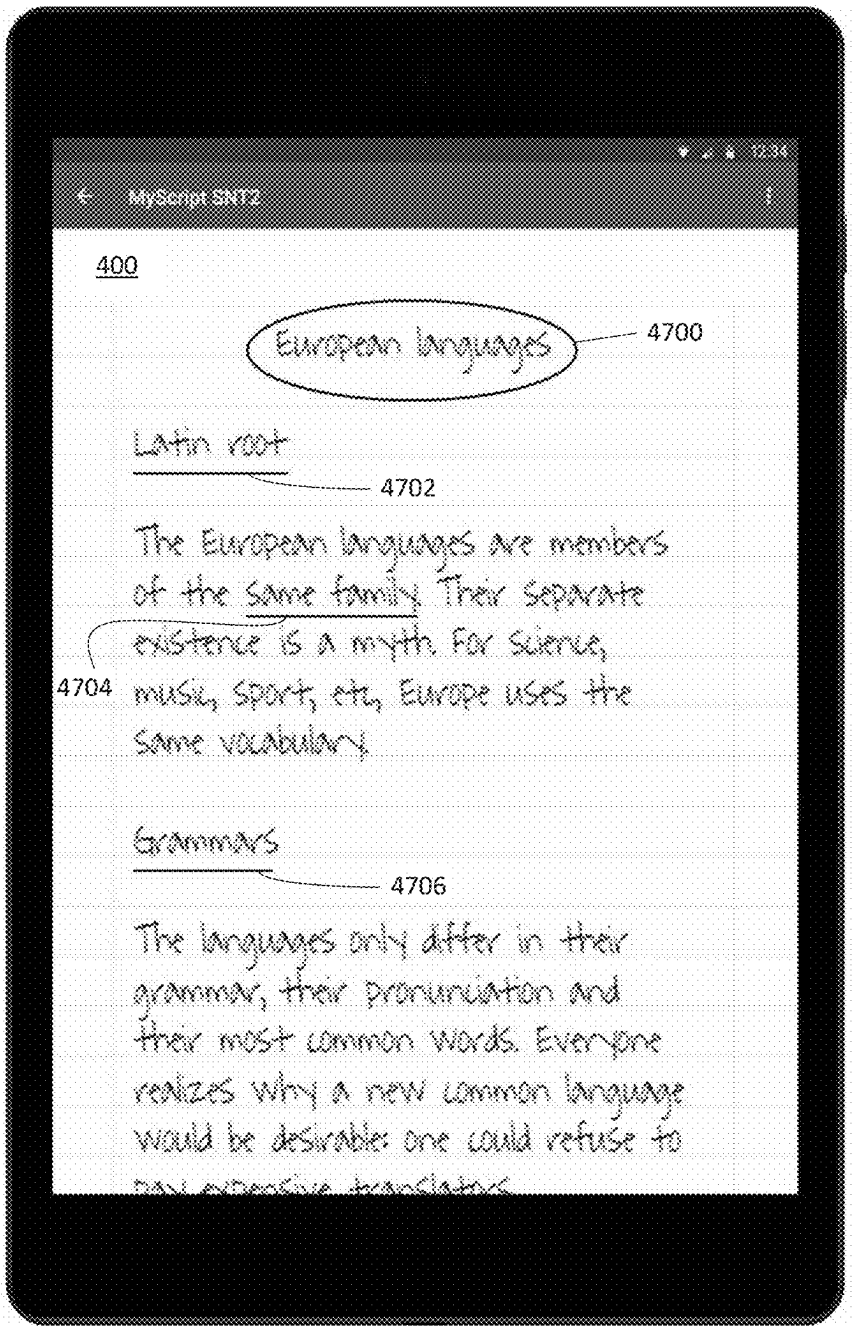

MyScript SNT2
400
European languages
4700
Latin root
4702
The European languages are members of the same family. Their separate existence is a myth. For science, music, sport, etc, Europe uses the same vocabulary.
4704
Grammars
4706
The languages only differ in their grammar, their pronunciation and their most common words. Everyone realizes why a new common language would be desirable: one could refuse to

FIG. 47

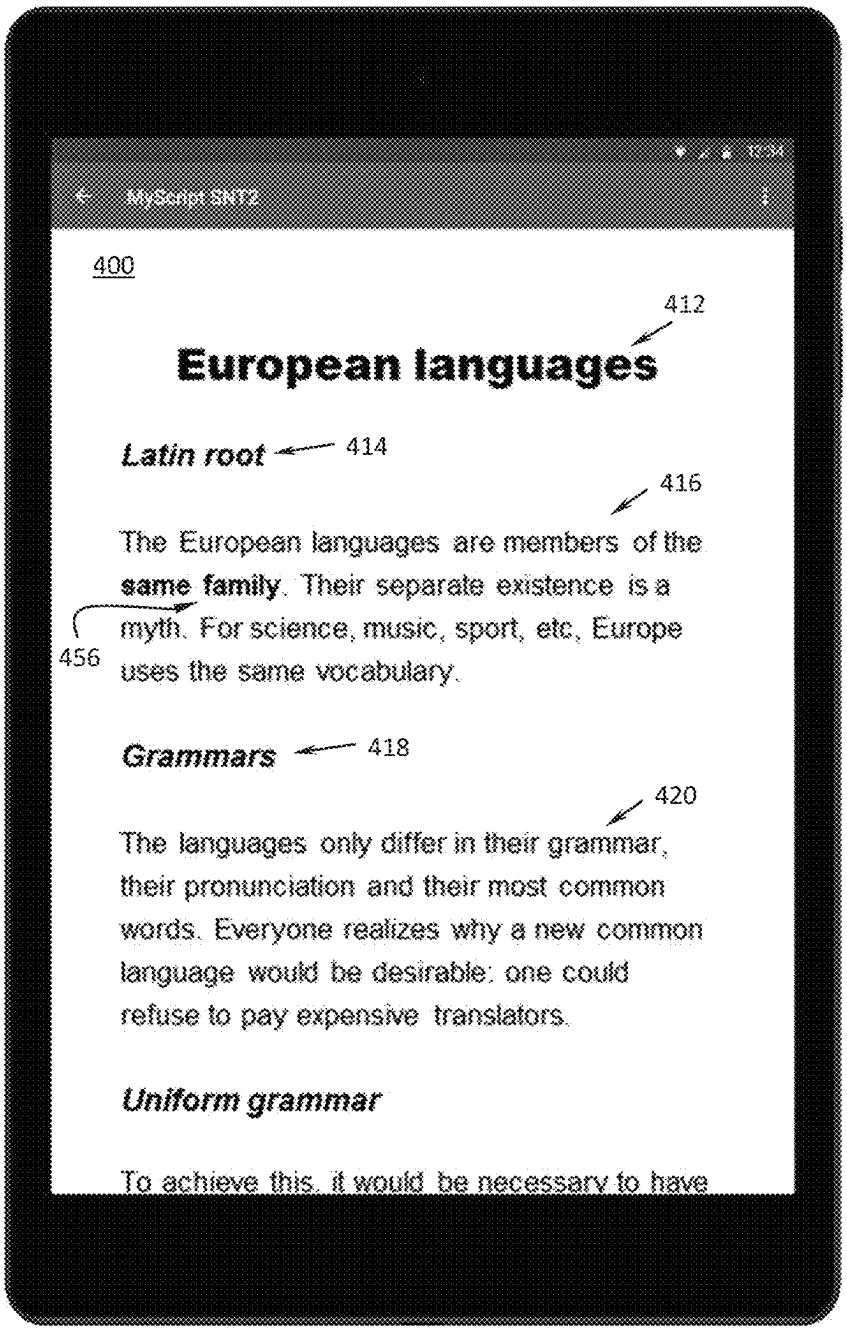

MyScript SNT2
400
412
European languages
Latin root
414
416
The European languages are members of the same family. Their separate existence is a myth. For science, music, sport, etc, Europe uses the same vocabulary.
456
Grammars
418
420
The languages only differ in their grammar, their pronunciation and their most common words. Everyone realizes why a new common language would be desirable: one could refuse to pay expensive translators.
Uniform grammar
To achieve this, it would be necessary to have

FIG. 48

SYSTEM AND METHOD FOR NOTE TAKING WITH GESTURES

CROSS-REFERENCE TO RELATED APPLICATIONS

This application claims priority to European Application No. 15290296.1 filed on Dec. 1, 2015, which is incorporated herein by reference in its entirety and made a part thereof.

TECHNICAL FIELD

The present description relates generally to the field of digital note taking systems and methods using computing device interfaces. The present description relates more specifically to digital note taking systems and methods capable of recognizing user input handwriting of various gestures in relation to displayed characters, symbols and objects.

BACKGROUND

Computing devices continue to become more ubiquitous to daily life. They take the form of computer desktops, laptop computers, tablet computers, hybrid computers (2-in-1s), e-book readers, mobile phones, smartphones, wearable computers (including smartwatches, smart glasses/headsets), global positioning system (GPS) units, enterprise digital assistants (EDAs), personal digital assistants (PDAs), game consoles, and the like. Further, computing devices are being incorporated into vehicles and equipment, such as cars, trucks, farm equipment, manufacturing equipment, building environment control (e.g., lighting, HVAC), and home and commercial appliances.

Computing devices generally consist of at least one processing element, such as a central processing unit (CPU), some form of memory, and input and output devices. The variety of computing devices and their subsequent uses necessitate a variety of interfaces and input devices. One such input device is a touch sensitive surface such as a touch screen or touch pad wherein user input is received through contact between the user's finger or an instrument such as a pen or stylus and the touch sensitive surface. Another input device is an input surface that senses gestures made by a user above the input surface. A further input device is a position detection system which detects the relative position of either touch or non-touch interactions with a non-touch physical or virtual surface. Any of these methods of input can be used generally for drawing or inputting text. When user input is text the user's handwriting is interpreted using a handwriting recognition system or method.

One application of handwriting recognition in portable computing devices, such as smartphones, phablets and tablets, is in note taking. This particularly occurs in education and business settings where the user of the computing device captures notes, for example, during a lecture or meeting. This is usually done by the user launching a handwritten note taking application on the computing device which accepts and interprets, either locally in the device or remotely via a communications link of the device, handwritten notes input on the touch sensitive surface. Conventionally such handwritten note taking applications are limited in their capabilities to provide a full document creation experience to users from the notes taken, since the focus of these applications has primarily been recognition accuracy rather than document creation. That is, available applications provide recognition of handwriting and conversion of the recognized handwriting into typeset text, with various

feedback mechanisms to the user, but that is generally the extent of the interaction with the input.

If the user desires any further interaction with the output text, such as editing the content, manipulating the layout of the notes, or converting or adding the notes into a document, the text generally needs to be imported into a separate document processing application. This in itself is no great problem, however as the original layout of the handwritten notes and the actual input handwriting itself, so-called digital ink, is typically discarded in the import process, the user must refer back to the original notes in order to ascertain the intent in the notes taken. For example, the user may have emphasized certain words or passages in the notes either by annotation or decoration, or through the layout of the notes content itself.

Some available handwriting digital note taking applications provide the ability to edit on the digital ink. However, this is generally done through the input of particular gestures for causing some sort of control, e.g., the launching of menus or running processes. The Applicant has found however that when using handwriting note taking applications users generally are unable or do not desire to learn specific gestures that are not natural or intuitive, or to make editing selections through menus and the like.

Further, gestures are also used for causing some sort of change of content, e.g., editing, style change, formatting. Accordingly, it is necessary for the application to determine the users' intent of using the gesture so that gestures for control and gestures for content input can be distinguished and the appropriate actions taken. The challenge is some interactions have more than one intent or could even constitute content itself.

SUMMARY

The examples of the present invention that are described herein below provide systems and methods for editing digital documents on computing devices.

In one example, the computing device may include a processor and at least one non-transitory computer readable medium for recognizing handwriting input under control of the processor. The non-transitory computer readable medium may be configured to cause display of first digital ink in accordance with document content on a display interface of a computing device, detect handwriting input as one or more editing gestures with respect to the document content, and cause display of second digital ink in accordance with handwriting input on the display interface. The second digital ink may be different than the first digital ink and/or may be a normalized version of the first digital ink. The non-transitory computer readable medium may be configured to cause re-display of, on the display interface, the document content as the third digital ink based on the detected editing gestures. The third digital ink may be different than the first and second digital ink, and may further be a typeset version of the first digital ink.

In another example, the present invention includes a method for editing documents on computing devices. Each computing device may comprise a processor and at least one non-transitory computer readable medium for recognizing handwriting input under control of the processor. The method may include the steps of displaying first digital ink in accordance with document content on a display interface of a computing device, detecting handwriting input as one or more editing gestures with respect to the document content; and displaying second digital ink in accordance with handwriting input on the display interface. The second digital ink

may be different than the first digital ink and/or may be a normalized version of the first digital ink. The method may include the step of re-displaying of, on the display interface, the document content as the third digital ink based on the detected editing gestures. The third digital ink may be different than the first and second digital ink, and may further be a typeset version of the first digital ink.

In another example, the present invention includes a non-transitory computer readable medium having a computer readable program code embodied therein. The computer readable program code may be adapted to be executed to implement a method for recognizing input of editing handwriting to a computing device. The computing device may comprise a processor and at least one system non-transitory computer readable medium for recognizing the input under control of the processor. The method may include the steps of displaying first digital ink in accordance with document content on a display interface of a computing device, detecting handwriting input as one or more editing gestures with respect to the document content, and displaying second digital ink in accordance with handwriting input on the display interface. The second digital ink may be different than the first digital ink and/or may be a normalized version of the first digital ink. The method may include the step of re-displaying of, on the display interface, the document content as the third digital ink based on the detected editing gestures. The third digital ink may be different than the first and second digital ink, and may further be a typeset version of the first digital ink.

BRIEF DESCRIPTION OF THE DRAWINGS

The present system and method will be more fully understood from the following detailed description of the examples thereof, taken together with the drawings. In the drawings like reference numerals depict like elements. In the drawings:

FIG. **1** shows a block diagram of a computing device in accordance with an example of the present system and method;

FIG. **2** shows a block diagram of a system for handwriting recognition in accordance with an example of the present system and method;

FIG. **3** shows a block diagram illustrating detail of the handwriting recognition system of FIG. **2** in accordance with an example of the present system and method;

FIG. **4** shows a schematic view of an example visual rendering of a scroll page on a portion of an input surface of the computing device;

FIGS. **5**A and **5**B show an example of an editing gesture relating to breaking a word in accordance with the present system and method;

FIGS. **6**A and **6**B show an example of an editing gesture relating to breaking a line in accordance with the present system and method;

FIGS. **7**A and **7**B show another example of an editing gesture relating to breaking a line in accordance with the present system and method;

FIGS. **8**A-**8**C show a schematic view of an example visual rendering of a scroll page on a portion of an input surface of the computing device that includes an example of an editing gesture relating to breaking a line in accordance with the present system and method;

FIGS. **9**A and **9**B show an example of an editing gesture relating to breaking a paragraph in accordance with the present system and method;

FIGS. **10**A-**10**D show another example of an editing gesture relating to breaking a paragraph in accordance with the present system and method;

FIGS. **11**A and **11**B show a schematic view of an example visual rendering of a scroll page on a portion of an input surface of the computing device that includes an example of an editing gesture relating to breaking a paragraph in accordance with the present system and method;

FIGS. **12**A-**12**C show an example of an editing gesture relating to joining two words in accordance with the present system and method;

FIGS. **13**A-**13**E show an example of an editing gesture relating to joining two lines in accordance with the present system and method;

FIG. **14** shows an example of an editing gesture relating to joining two lines in accordance with the present system and method;

FIGS. **15**A-**15**C show an example of an editing gesture relating to joining two paragraphs in accordance with the present system and method;

FIGS. **16**A and **16**B show an example of an editing gesture relating to unspacing two paragraphs in accordance with the present system and method;

FIGS. **17**A-**17**D show a schematic view of an example visual rendering of a scroll page on a portion of an input surface of the computing device that includes an example of an editing gesture relating unspacing paragraphs in accordance with the present system and method;

FIGS. **18**A-**18**D show examples of editing gestures relating to erasing one character in accordance with the present system and method;

FIGS. **19**A-**19**C show examples of editing gestures relating to erasing several characters in accordance with the present system and method;

FIGS. **20**A-**20**C show examples of editing gestures relating to erasing one word in accordance with the present system and method;

FIGS. **21**A-**21**C show examples of editing gestures relating to erasing two words in accordance with the present system and method;

FIGS. **22**A-**22**C show examples of editing gestures relating to erasing several words in accordance with the present system and method;

FIGS. **23**A-**23**B show a schematic view of an example visual rendering of a scroll page on a portion of an input surface of the computing device that includes an example of an editing gesture relating erasing words in accordance with the present system and method;

FIGS. **24**A-**24**C show examples of editing gestures relating to erasing one line of text in accordance with the present system and method;

FIGS. **25**A-**25**C show examples of editing gestures relating to erasing several lines of text in accordance with the present system and method;

FIGS. **26**A and **26**B show examples of editing gestures relating to erasing one paragraph in accordance with the present system and method;

FIGS. **27**A and **27**B show examples of editing gestures relating to erasing several paragraphs in accordance with the present system and method;

FIG. **28** shows a multi-line paragraph input as handwritten content to the input area;

FIG. **29** shows the paragraph of FIG. **28** with additional handwritten content input and rendered as digital ink in accordance with the present system and method;

FIG. **30** shows the result of typesetting the paragraph of FIG. **28** with the additional input of FIG. **29** in accordance with the present system and method;

FIGS. **31**A and **31**B show another example of the paragraph of FIG. **28** with additional handwritten content input and rendered as digital ink in accordance with the present system and method;

FIG. **32** shows another example of the paragraph of FIG. **28** with additional handwritten content input and rendered as digital ink in accordance with the present system and method;

FIGS. **33**A and **33**B show another example of the paragraph of FIG. **28** with additional handwritten content input and rendered as digital ink in accordance with the present system and method;

FIG. **34** shows the result of typesetting the paragraph of FIG. **28** with the additional input of FIG. **33**B in accordance with the present system and method;

FIGS. **35**A-**35**D show another example of the paragraph of FIG. **28** with additional handwritten content input and rendered as digital ink in accordance with the present system and method;

FIG. **36** shows another example of an editing gesture relating to breaking a line in accordance with the present system and method;

FIGS. **37**A and **37**B shows another example of an editing gesture relating to breaking a line in accordance with the present system and method;

FIG. **38** shows the result of typesetting the paragraph of FIGS. **37**A and **37**B in accordance with the present system and method;

FIG. **39** shows another example of the paragraph of FIG. **28** with additional handwritten content input and rendered as digital ink in accordance with the present system and method;

FIGS. **40**A-**40**C show another example of the paragraph of FIG. **28** with additional handwritten content input and rendered as digital ink in accordance with the present system and method;

FIG. **41** shows another example of an editing gesture relating to breaking a line in accordance with the present system and method;

FIGS. **42**A and **42**B shows another example of an editing gesture relating to breaking a line in accordance with the present system and method;

FIG. **43** shows another example of the paragraph of FIG. **28** with additional handwritten content input and rendered as digital ink in accordance with the present system and method;

FIG. **44** shows another example of the paragraph of FIG. **28** with additional handwritten content input and rendered as digital ink in accordance with the present system and method;

FIGS. **45**A and **45**B shows another example of an editing gesture relating to joining two lines in accordance with the present system and method;

FIG. **46** shows a schematic view of an example visual rendering of a scroll page on a portion of an input surface of the computing device that includes additional handwritten content input and rendered as digital ink in accordance with the present system and method;

FIG. **47** shows a schematic view of an example visual rendering of a scroll page on a portion of an input surface of the computing device that the result of the typesetting of FIG. **46** in accordance with the present system and method; and

FIG. **48** shows a schematic view of an example visual rendering of a scroll page on a portion of an input surface of the computing device that the result of the typesetting of FIGS. **45** and **46** in accordance with the present system and method.

DETAILED DESCRIPTION

In the following detailed description, numerous specific details are set forth by way of examples in order to provide a thorough understanding of the relevant teachings. However, it should be apparent to those skilled in the art that the present teachings may be practiced without such details. In other instances, well known methods, procedures, components, and/or circuitry have been described at a relatively high-level, without detail, in order to avoid unnecessarily obscuring aspects of the present teachings.

Reference to and discussion of directional features such as up, down, above, below, lowest, highest, horizontal, vertical, etc., are made with respect to the Cartesian coordinate system as applied to the input surface on which the input to be recognized is made. Further, terms such as left and right are made in relation to the reader's frame of reference when viewing the drawings. Furthermore, the use of the term 'text' in the present description is understood as encompassing all alphanumeric characters, and strings thereof, in any written language and common place non-alphanumeric characters, e.g., symbols, used in written text. Further still, the term 'non-text' in the present description is understood as encompassing freeform handwritten or hand-drawn content and rendered text and image data, as well as non-alphanumeric characters, and strings thereof, and alphanumeric characters, and strings thereof, which are used in non-text contexts. Furthermore, the examples shown in these drawings are in a left-to-right written language context, and therefore any reference to positions can be adapted for written languages having different directional formats.

The various technologies described herein generally relate to capture, processing and management of hand-drawn and handwritten content on portable and non-portable computing devices in a manner which retains the inputted style of the content while allowing conversion to a faithful typeset or beautified version of that content. The systems and methods described herein may utilize recognition of users' natural writing and drawing styles input to a computing device via an input surface, such as a touch sensitive screen, connected to, or of, the computing device or via an input device, such as a digital pen or mouse, connected to the computing device or via a physical or virtual surface monitored by a position detection system. Whilst the various examples are described with respect to recognition of handwriting input using so-called online recognition techniques, it is understood that application is possible to other forms of input for recognition, such as offline recognition in which images rather than digital ink are recognized. The terms hand-drawing and handwriting are used interchangeably herein to define the creation of digital content by users through use of their hands either directly onto a digital or digitally connected medium or via an input tool, such as a hand-held stylus. The term "hand" is used herein to provide concise description of the input techniques, however the use of other parts of a users' body for similar input is included in this definition, such as foot, mouth and eye.

FIG. **1** shows a block diagram of an example computing device **100**. The computing device may be a computer desktop, laptop computer, tablet computer, e-book reader, mobile phone, smartphone, wearable computer, digital

watch, interactive whiteboard, global positioning system (GPS) unit, enterprise digital assistant (EDA), personal digital assistant (PDA), game console, or the like. The computing device **100** includes components of at least one processing element, some form of memory and input and/or output (I/O) devices. The components communicate with each other through inputs and outputs, such as connectors, lines, buses, cables, buffers, electromagnetic links, networks, modems, transducers, IR ports, antennas, or others known to those of ordinary skill in the art.

The computing device **100** has at least one display **102** for outputting data from the computing device such as images, text, and video. The display **102** may use LCD, plasma, LED, iOLED, CRT, or any other appropriate technology that is or is not touch sensitive as known to those of ordinary skill in the art. At least some of the display **102** is co-located with at least one input surface **104**. The input surface **104** may employ technology such as resistive, surface acoustic wave, capacitive, infrared grid, infrared acrylic projection, optical imaging, dispersive signal technology, acoustic pulse recognition, or any other appropriate technology as known to those of ordinary skill in the art to receive user input. The input surface **104** may be bounded by a permanent or video-generated border that clearly identifies its boundaries.

In addition to the input surface **104**, the computing device **100** may include one or more additional I/O devices (or peripherals) that are communicatively coupled via a local interface. The additional I/O devices may include input devices such as a keyboard, mouse, scanner, microphone, touchpads, bar code readers, laser readers, radio-frequency device readers, or any other appropriate technology known to those of ordinary skill in the art. Further, the I/O devices may include output devices such as a printer, bar code printers, or any other appropriate technology known to those of ordinary skill in the art. Furthermore, the I/O devices may include communications devices that communicate both inputs and outputs such as a modulator/demodulator (modem; for accessing another device, system, or network), a radio frequency (RF) or other transceiver, a telephonic interface, a bridge, a router, or any other appropriate technology known to those of ordinary skill in the art. The local interface may have additional elements to enable communications, such as controllers, buffers (caches), drivers, repeaters, and receivers, which are omitted for simplicity but known to those of skill in the art. Further, the local interface may include address, control, and/or data connections to enable appropriate communications among the other computer components.

The computing device **100** also includes a processor **106**, which is a hardware device for executing software, particularly software stored in the memory **108**. The processor can be any custom made or commercially available general purpose processor, a central processing unit (CPU), a semiconductor based microprocessor (in the form of a microchip or chipset), a macroprocessor, microcontroller, digital signal processor (DSP), application specific integrated circuit (ASIC), field programmable gate array (FPGA) or other programmable logic device, discrete gate or transistor logic, discrete hardware components, state machine, or any combination thereof designed for executing software instructions known to those of ordinary skill in the art. Examples of suitable commercially available microprocessors are as follows: a PA-RISC series microprocessor from Hewlett-Packard Company, an 80×86 or Pentium series microprocessor from Intel Corporation, a PowerPC microprocessor from IBM, a Sparc microprocessor from Sun Microsystems, Inc., a 68xxx series microprocessor from Motorola Corporation, DSP microprocessors, or ARM microprocessors.

The memory **108** can include any one or a combination of volatile memory elements (e.g., random access memory (RAM, such as DRAM, SRAM, or SDRAM)) and nonvolatile memory elements (e.g., ROM, EPROM, flash PROM, EEPROM, hard drive, magnetic or optical tape, memory registers, CD-ROM, WORM, DVD, redundant array of inexpensive disks (RAID), another direct access storage device (DASD)). Moreover, the memory **108** may incorporate electronic, magnetic, optical, and/or other types of storage media. The memory **108** can have a distributed architecture where various components are situated remote from one another but can also be accessed by the processor **106**. Further, the memory **108** may be remote from the device, such as at a server or cloud-based system, which is remotely accessible by the computing device **100**. The memory **108** is coupled to the processor **106**, so the processor **106** can read information from and write information to the memory **108**. In the alternative, the memory **108** may be integral to the processor **106**. In another example, the processor **106** and the memory **108** may both reside in a single ASIC or other integrated circuit.

The software in the memory **108** includes an operating system **110** and an application **112**. The software optionally further includes a handwriting recognition (HWR) system **114** which may each include one or more separate computer programs. Each of these has an ordered listing of executable instructions for implementing logical functions. The operating system **110** controls the execution of the application **112** (and the HWR system **114**). The operating system **110** may be any proprietary operating system or a commercially available operating system, such as WEBOS, WINDOWS®, MAC and IPHONE OS®, LINUX, and ANDROID. It is understood that other operating systems may also be utilized.

The application **112** includes one or more processing elements related to detection, management and treatment of user input (discussed in detail later). The software may also include one or more other applications related to handwriting recognition, different functions, or both. Some examples of other applications include a text editor, telephone dialer, contacts directory, instant messaging facility, computer-aided design (CAD) program, email program, word processing program, web browser, and camera. The application **112**, and the other applications, include program(s) provided with the computing device **100** upon manufacture and may further include programs uploaded or downloaded into the computing device **100** after manufacture.

The HWR system **114**, with support and compliance capabilities, may be a source program, executable program (object code), script, application, or any other entity having a set of instructions to be performed. When a source program, the program needs to be translated via a compiler, assembler, interpreter, or the like, which may or may not be included within the memory, so as to operate properly in connection with the operating system. Furthermore, the handwriting recognition system with support and compliance capabilities can be written as (a) an object oriented programming language, which has classes of data and methods; (b) a procedure programming language, which has routines, subroutines, and/or functions, for example but not limited to C, C++, Pascal, Basic, Fortran, Cobol, Perl, Java, Objective C, Swift, and Ada; or (c) functional programing languages for example but no limited to Hope, Rex, Common Lisp, Scheme, Clojure, Racket, Erlang, OCaml, Haskell, Prolog, and F#. Alternatively, the HWR system **114**

may be a method or system for communication with a handwriting recognition system remote from the device, such as server or cloud-based system, but is remotely accessible by the computing device **100** through communications links using the afore-mentioned communications I/O devices of the computing device **100**. Further, the application **112** and the HWR system **114** may operate together or be combined as a single application.

Strokes entered on or via the input surface **104** are processed by the processor **106** as digital ink. A user may enter a stroke with a finger or some instrument such as a pen or stylus suitable for use with the input surface. The user may also enter a stroke by making a gesture above the input surface **104** if technology that senses motions in the vicinity of the input surface **104** is being used, or with a peripheral device of the computing device **100**, such as a mouse or joystick. A stroke is characterized by at least the stroke initiation location, the stroke termination location, and the path connecting the stroke initiation and termination locations. Because different users may naturally write the same object, e.g., a letter, a shape, a symbol, with slight variations, the HWR system accommodates a variety of ways in which each object may be entered whilst being recognized as the correct or intended object.

FIG. **2** is a schematic pictorial of an example of the HWR system **114**, in either its local (i.e., loaded on the device **100**) or remote (i.e., remotely accessible by the device **100**) forms. The HWR system **114** includes stages such as preprocessing **116**, recognition **118** and output **120**. The preprocessing stage **116** processes the digital ink to achieve greater accuracy and reducing processing time during the recognition stage **118**. This preprocessing may include normalizing of the path connecting the stroke initiation and termination locations by applying size normalization and/or methods such as B-spline approximation to smooth the input. The preprocessed strokes are then passed to the recognition stage **118** which processes the strokes to recognize the objects formed thereby. The recognized objects are then output **120** to the display **102** generally as a typesetted version of the handwritten elements/characters.

The recognition stage **118** may include different processing elements or experts. FIG. **3** is a schematic pictorial of the example of FIG. **2** showing schematic detail of the recognition stage **118**. Three experts, a segmentation expert **122**, a recognition expert **124**, and a language expert **126**, are illustrated which collaborate through dynamic programming to generate the output **120**.

The segmentation expert **122** defines the different ways to segment the input strokes into individual element hypotheses, e.g., alphanumeric characters and mathematical operators, text characters, individual shapes, or sub expression, in order to form expressions, e.g., words, mathematical equations, or groups of shapes. For example, the segmentation expert **122** may form the element hypotheses by grouping consecutive strokes of the original input to obtain a segmentation graph where each node corresponds to at least one element hypothesis and where adjacency constraints between elements are handled by the node connections. Alternatively, the segmentation expert **122** may employ separate experts for different input types, such as text, drawings, equations, and music notation.

The recognition expert **124** provides classification of the features extracted by a classifier **128** and outputs a list of element candidates with probabilities or recognition scores for each node of the segmentation graph. Many types of classifiers exist that could be used to address this recognition task, e.g., Support Vector Machines, Hidden Markov Models, or Neural Networks such as Multilayer Perceptrons, Deep, Convolutional or Recurrent Neural Networks. The choice depends on the complexity, accuracy, and speed desired for the task.

The language expert **126** generates linguistic meaning for the different paths in the segmentation graph using language models (e.g., grammar or semantics). The expert **126** checks the candidates suggested by the other experts according to linguistic information **130**. The linguistic information **130** can include a lexicon, regular expressions, etc. The language expert **126** aims at finding the best recognition path. In one example, the language expert **126** does this by exploring a language model such as finite state automaton (FSA) representing the content of linguistic information **130**. In addition to the lexicon constraint, the language expert **126** may use statistical information modeling for how frequent a given sequence of elements appears in the specified language or is used by a specific user to evaluate the linguistic likelihood of the interpretation of a given path of the segmentation graph.

The present system and method make use of the HWR system **114** in order to recognize handwritten input to the device **100**. As mentioned earlier, the application **112** includes application(s) for handling the layout of the recognized user input. Such applications may be provided in an architecture with separate layers for handling different processing. One or more of these layers may be remote to the device **100** accessible via the communications channels mentioned earlier. The layers may include application wrapper(s), platform wrapper(s) and (platform specific) application user interface(s).

The application **112** provided by the present system and method allows users, such as students, academic and working professionals, to take handwritten notes, such as during lectures, meetings and brainstorming sessions, which they can transform into sharable formatted documents on their portable or non-portable computing devices using natural and intuitive operations, such as gestures. The application allows taking of 'clean' notes which can be searched and memorized properly using their computing devices. The present system and method further allow natural writing to be input since the writing is made on (ruled) lines, and paragraphs and other layout elements are respected responsive to orientation (i.e., portrait or landscape) and device display capacity. Synchronization of the captured notes between devices and auto-saving of notes are also supported, as are export options to various formats, such as a text, Latex, HTML, image, pdf, etc. These and other aspects are now described.

The application **112** utilizes a framework for the definition and handling of documents and document elements so as to provide structured content allowing layout definition and preservation. This document model handles different types of document elements, which are best described with HTML5 reference, for example, see W3C HTML5 DOM—Kinds of content: http://www.w3.org/TR/htm15/dom.html#kinds-of-content. The document model utilized by the application **112** does not reproduce HTML specifications, rather it takes inspiration from them. A subset of the elements of the document model is flow, sectioning, heading, phrasing, embedded and interactive.

The Applicant has found that users of existing digital note taking applications have issues understanding how text behaves when editing, with text reflow and uncontrolled text/element position. In order to lower the learning curve for use of such applications, handwriting should be closer to what users are familiar with. That is, writing with keyboard

in text editing software/web services and editing through keyboard and mouse interaction, with the same model and behaviors. Any digital device user is already accustomed to gesturing on screen to write or edit content. Gesturing is a natural and intuitive pattern on touch and hover devices. A quick solution in the handwritten note taking environment would be to just adapt presently used gestures, such a tap, double tap, press, long press, swipe, pinch, to allow editing in the digital note taking and document creation environment. While these gestures are generally intuitively provided for their existing functions on touch and hover devices, the Applicant has further found that this intuitiveness does not necessarily carry over well to content editing functions based on user feedback. Accordingly, the present system and method provides new classes of gesturing and behaviors for providing content editing functions. These and other features of the present system and method are now described in detail.

FIG. **4** shows a schematic view of an example visual rendering of a note taking page **400** on a portion of the input surface **104** of an example computing device **100**. The page **400** is shown in editing view in which content entry and editing is performed. However, other views may be used for document structure editing and notebook navigation. Besides providing user interface (UI) controls (described later) the editing view provides a full-width page that overflows vertically from the display **102** of the device **100**. The page **400** is of flexible height, in essence of infinite height, and can be vertically scrolled using well understood touch screen computing device gestures or non-touch screen computing device methods, such as a with a mouse. In order to maximize writing space for users (particularly on smaller devices), an empty scrollable page is defined to occupy at least the whole editing view height.

In the editing view (and other views as appropriate) several page elements are displayed including a top padding **402**, a left margin **404** and a right margin **406**. The padding and margins define an input area **408** which has a line pattern background **410**. The line pattern **410** has horizontal lines separated by a multiple of the vertical rhythm height unit being the density independent pixel (dp). Regardless of the particular device **100**, the vertical distance between the horizontal lines **410** is defined by a line pattern unit (LPU) and the vertical rhythm height unit provides a graduated measure of the LPU on a particular device. For example, the LPU may be set at about one centimeter for any device being a certain multiple of the vertical rhythm height unit or users may be allowed to customize the line pattern unit to a different multiple of the vertical rhythm height unit according to their writing style. Alternatively, the vertical rhythm may be based on typeset text size (e.g., the minimum text size) and the LPU is provided as a multiple of this typeset text size. All lines **410** are displayed with the same light and subtle color, e.g., grey, that is visible but faded with respect to the rendering of the content itself. In this way the line pattern is noticeable but unobtrusive so as to guide the handwriting input without distracting from the content entry. The line pattern background **410** is displayed in handwritten text context and serves multiple purposes:

- forces users to write onto it so that handwriting recognition is optimized
- helps users adopt regular vertical rhythm writing leading to same writing size and better block dimensions and positions
- helps users control line skipping to define paragraphs.

The top padding **402** is displayed on any new page or section and does not contain the line pattern background. This helps users write with proper vertical spacing, i.e., not too close from the interface **104** border and leaves room for other elements, such as section titling and tagging. The top padding **402** is defined by multiples of the LPU, e.g., in the example of FIG. **4** the top padding is two times the LPU. However, it may be desired to allow writing in the top padding for headings, e.g., document and section titles, annotations and the like. Similar bottom padding at the bottom of the input area **408** may also be provided.

The left and right margins **404** and **406** are displayed with vertical lines. In the example of FIG. **4** these lines are similar to the line pattern **410** in color and definition, however these lines may be more boldly displayed if desired. The vertical line margins are positioned from the corresponding left and right 'page' edges by multiples of the vertical rhythm unit. In the example of FIG. **4**, the margins **404**, **406** are not displayed in the top padding **402** and start from the first line pattern **410** horizontal line. However, the margins may extend into the top padding if desired. Further, the line pattern **410** is not displayed in the margin elements in order to discourage writing in these areas. However, it may be desired to allow writing in the margins for annotations and the like. Alternatively, the margins may only be displayed on hover-over or contact interaction with the horizontal extents of the page **400** or totally omitted, thereby, for example, relying on natural handwriting behaviors of users to not write too close to the screen/UI edges.

The input of handwritten content onto the page **400** is performed through the use of gestures in relation to the input surface **104**, e.g., through touch, force and/or proximity depending on the screen technology of the device **100**. Gesture detection may be handled differently depending on the ability of the computing device **100** to differentiate a users' finger from a stylus or pen (by which the device defines passive and active styli) or the ability of a stylus to indicate or communicate to the device that it is being used for handwriting or the ability of users to provide such an indication. By default, and in devices which do not differentiate, any single-point touch or hover event within the input area **408** is to be considered as content input or content interaction.

The scroll page can be provided as a constraint-free canvas that allows users to create object blocks (blocks of text, drawings, etc.) anywhere without worrying about sizing or alignment. However, in order to provide professionally formatted documents containing structured content, certain alignment elements or pattern may be defined onto which all content is to be aligned/defined (either at input or along user interaction). This allows blocks to be defined as that rendering of the blocks is responsive to the device size/orientation. An example alignment pattern and responsive display is described in United States Patent Application Publication No. 2017/0060819 titled "System and Method of Digital Note Taking" filed in the name of the present Applicant and Assignee, the entire content of which is incorporated by reference herein.

FIG. **4** further shows handwritten input rendered as digital ink. In particular, in the example of FIG. **4**, there is shown five text objects **412**, **414**, **416**, **418** and **420**. The text object **412** is a text block containing two words on a single line **410** and defines a document or section title, for example. The text object **414** is a text block containing two words on a single line **410** spaced one line **410** from the block **412** and defines a section title or sub-title, for example. The text object **416** is a text block containing many words over multiple lines **410** with the first (topmost) line **410** spaced one line **410** from the block **414** and defines a (section) paragraph, for

example. The text object **418** is a text block containing two words on a single line **410** spaced one line **410** from the bottommost line **410** of the block **416** and defines a section title or sub-title, for example. The text object **420** is a text block containing many words over multiple lines **410** with the topmost line **410** spaced one line **410** from the block **418** and defines a (section) paragraph, for example.

The document model defines sectioning content as content that defines the scope of headings and footers. Heading content defines the header of a section, whether explicitly marked up using sectioning content elements or implied by the heading content itself. A paragraph is typically a run of flow content that forms a block of text with one or more sentences, without any empty lines therebetween. Paragraphs may be any width (from single word to full width) depending on input length and position. At any time, the widest line of a paragraph defines its boundaries with potential consequences on the document model when new content is input. As long as all lines (but the last) are roughly the same width, the document model causes new content to be appended inline to the paragraph when typesetting reflow is performed. As discussed below, paragraphs are mainly created by skipping a line **410** between content input. When text is entered immediately before or after a paragraph without leaving an empty line, all new content is merged into the existing paragraph. All paragraphs may be split or merged with either content editing or gestures (described later).

As can be seen, the content of the text block **420** flows beyond the bottom of the input area **408** as displayed in FIG. **4**. However, the content can be viewed (and input) through users scrolling the page **400** by single-point interaction (e.g., one-finger or stylus gesture touching or hovering over the interface **104**) or multi-point interaction (e.g., two fingers touching or hovering over the interface **104** at about the same time) either anywhere on the input area or in a dedicated area(s), such as in one or more of the margins. When using a device that is able to differentiate finger from stylus, or a stylus which indicates its use to the device, or the user provides this indication, thus defining an active stylus for content input for example, multi-point interaction is not necessary and scrolling is available with a one-finger vertical scroll gesture anywhere on the interface surface **104**, for example.

As mentioned above, the text blocks **412**-**420** are each separated by an empty line **410**, e.g., a spacing of two LPUs. This simple constraint is placed on the input of separate paragraphs (and object blocks) in the vertical flow, in a manner which is easily understood and applied by users so that relative layout between object blocks in the vertical flow of the scroll page **400** can be cleanly maintained and applied to directly input content and dragged & dropped or pushed & dropped blocks. Benefits of this approach will become apparent from the following description of certain examples of editing provided by the present system and method. Alternatively or additionally however, such a constraint may not be required, particularly with the use of gestures to define styling, as described later.

FIGS. **5** to **7**, **9**, **10**, **12**, **13**, **15**, **16**, **18** to **22** and **24** to **45** illustrate examples of editing gestures as performed on general example text blocks and FIGS. **8**, **11**, **14**, **17**, **23** and **46** to **48** illustrates those example editing gestures as performed on the example input of FIG. **4**. In the illustrations of FIGS. **5** to **7**, **9**, **10**, **12**, **13**, **15**, **16**, **18** to **22**, **24** to **27**, **36** to **38**, **41**, **42** and **45** document processing marks are shown depicting paragraph "¶", line break or carriage return "□" and character/word space "□" for illustrative purposes only, not as displayed handwritten input. In the context of the paragraphing rule described above:

- a paragraph is defined when a text block is separated from another text (or other object) block by at least one empty line **410**;
- a line break is defined when two text blocks are separated from one another through input of an associated gesture (described below) without an empty line **410** therebetween. Line breaks are not part of the explicit input, rather they are implicitly created/deleted by the document model only to take text layout into account and preserve user input. In this way, line breaks serve as layout helpers to express space in-between paragraphs;
- a space is defined when characters, words, symbols, etc. within a text block are separated from one another on the same line **410** or over subsequent lines **410**.

The symbols are depicted to aid understanding of the results of the detection and recognition of the editing gestures and to provide a contextual comparison to similar operations performed using a keyboard, e.g., the keying of a return or enter key to provide a paragraph "¶" or line break or carriage return "□" and the keying of a space bar to provide a space "□". A line break is a special element that introduces a carriage return into phrasing content or content flow.

Before description of the examples is made, a brief description of an example of how the application **112** and the HWR system **114** of the present system and method detects and recognizes handwritten gestures is now provided. As described earlier, strokes entered on or via the input surface **104** are processed by the HWR system **114** using segmentation, recognition and language models in order to recognize each stroke and groups of strokes as handwritten objects, such as text characters and words. The recognition of gestures uses the same approach to detect strokes of the gestures, e.g., characterized by at least the stroke initiation location, the stroke termination location, and the path connecting the stroke initiation and termination locations, and combines this stroke detection with the determination of other characteristics of the strokes to detect gesture input and the context of that input in order to recognize the particular gesture used. This is done because it is not known at this point whether the newly entered strokes are further handwritten content, such as additional text, or a gesture for enacting some sort of control on already entered (and recognized) content.

Accordingly, the input strokes are processed by the recognizer **118** of the HWR system **114** to determine the probabilities of the strokes being certain input (e.g., text, symbol, shape, a straight line, a zig-zag line, a curly line) and to determine characteristics such as writing direction (e.g., left, right, vertical, horizontal), geometry (e.g., type and numbers of changes in direction, shape) and relation to the LPU. These determined characteristics together with the determined probable stroke types allow a determination of the possibility of the processed strokes being a type of gesture (e.g., an underline, a strikethrough, a scratch-out). The determined possible gesture types are evaluated for context by comparing the strokes of the possible gestures and the proximate recognized objects (e.g., characters, words, paragraphs). This comparison is used to verify that the input strokes are gestures and not further text input, for example, and whether the possible gestures types can apply in the detected case. This is done, for example, by comparing relative geometric features of the strokes of the gesture and objects/text (e.g., base-lines, bounding boxes, overlapping ratios) and/or by consideration of the language of the

recognized text. A set of results is then proposed to the application **112** with probability scores and intervals of digital ink on which the probable gesture(s) must be applied. As such, the HWR system **114** only proposes gesture recognition results and does not make any updates to the context itself. In this way, any pre-determined recognition results, e.g., the recognized text for which a gesture has been input, is maintained. The application **112** processes these gesture recognition results in order to apply the modifications on the content or to ignore them.

The evaluation of some geometries can be disabled through UI settings on the application **112**. This allows constraint on the availability of certain gestures. For example, the HWR system **114** may be able to recognize a strikethrough deletion gesture in both the left-to-right and right-to-left directions, however it may be desired to only allow strike-through in one of those directions. Such a disablement could be used to ease recognition or to allow the use of further gestures, e.g., a (reverse) strikethrough in the disabled direction may be recognized as an 'undo' of the recognized strikethrough. It is also possible to disable some types of gestures altogether, e.g., all erase gestures. Further, the evaluation of certain characteristics may be used to increase the probability scores. For example, writing flow may be used to increase the probability scores of add, overwrite, insert gestures so that other gestures will not be the best result when input is written continuously, e.g., the editing gestures subsequently follow content input, rather than being applied some time thereafter. This can be used to ease recognition of out of time content input, such as diacritics, e.g., the stroke for the character "t" after writing the full word containing the "t".

The example editing gestures are now described. FIG. **5**A shows a single line paragraph **500** of several words, "Break" **502**, "a" **504** and "word" **506**, as defined by the spaces therebetween, and a period **507** after the word **506**. An editing gesture **508** is detected as input as a top-to-bottom vertical line in a single stroke by a user swiping their finger or stylus tip from the top to the bottom in the displayed position shown, for example. That is, the stroke start position is above the stroke end position, in time order, as depicted by the gradual thickening of the stroke from the top to the bottom. The editing stroke **508** is detected by the HWR system **114** as passing between characters/letters of the word **506**, namely the "o" and the "r". This relative location of the editing stroke **508** to the (recognized) text **500**, e.g., within a word, provides one of the above-described characteristics of the editing gesture **508** whereas the top-to-bottom writing direction provides another characteristic.

Consideration of these characteristics alone may result in a determination that the editing gesture **508** is intended by the user to break or split the word **506** at the detected location. This recognition result is shown in FIG. **5**B in which the word **506** has been broken into two parts **506***a* and **506***b* having a space between the "o" and "r". As can be seen, the features of the paragraph **500** are otherwise retained. However, such a simple editing gesture could be confused through the handwriting recognition as being an additionally input character, e.g., the letter "l", if the HWR system **114** allows the recognition of such overwritten characters. In the illustrated case, the probability of this recognition result of the character "l" would most likely be lower than the probability of the editing gesture, 'break', because the language model of the HWR system **114** would return a poor result for the word "wolrd", for example. However, the language model could provide a relatively high probability for the miss-spelling of "world", for example, or the gesture could have been located between the "r" and "d" of the word **506** instead. Such recognition results depend on the language and context of input and as such recognition that is largely independent of such factors is preferred in order to provide a robust and widely applicable system and method.

Accordingly, the consideration of additional characteristics is made by the present system and method. Such a characteristic is the vertical extent of the editing stroke **508** in comparison to the input environment of the page **400**. For example, the stroke **508** crosses the line **410** on which the paragraph **500** is written (and also the line **410** thereabove), which is not a typical characteristic of the letter "l", for example, and can therefore be used to relatively decrease the probability of that character being recognized. Alternatively or additionally, such a characteristic is the vertical extent of the editing stroke **508** in comparison to combined characteristics of the proximate objects defining the geometrical relationship therebetween, such as, the mean, average and/or maximum line patterns of the elements of those objects (e.g., base-line, center-line, top-line, ascender-line, descender-line, matra-line, rukar-line, etc.). For example, the editing stroke **508** crosses the base-line of the strokes of the word **506** (and also of the paragraph **500**), which is not a typical characteristic of the letter "l", for example, and can therefore be used to relatively decrease the probability of that character being recognized.

The application **112** may be configured to display the input editing stroke(s) as digital ink, either with a shape as, for example, shown in FIG. **5**A, or some other appropriate rendering until the recognized editing operation is performed or may just perform the editing operation without such display. The display of the detected editing gesture may be used to provide recognition feedback to users with respect to recognition of the gesture itself and the relative location thereof. Otherwise, the eventual editing operation, e.g., the modified display shown in FIG. **5**B, may be used for such feedback.

FIG. **6**A shows a single line paragraph **600** of several words, "Break" **602**, "a" **604** and "line" **606**, as defined by the spaces therebetween, and a period **607** after the word **606**. An editing gesture **608** is detected as input as a top-to-bottom vertical line in a single stroke in the displayed position shown. The editing stroke **608** is detected by the HWR system **114** as passing between the words **602** and **604**. Like the example of FIG. **5**, this relative location of the editing stroke **608** to the (recognized) text **600**, e.g., in a space between words, provides one of the above-described characteristics of the editing gesture **608** whereas the top-to-bottom writing direction provides another characteristic. Consideration of these characteristics alone may result in a determination that the editing gesture **608** is intended by the user to break or split the single line of the paragraph **600** at the detected location into two lines. This recognition result is shown in FIG. **6**B in which the paragraph **600** has been broken into two lines **600***a* and **600***b* of text with the word **602** on the original (first) line **410** of the paragraph **600** as the line **600***a* and the words **604** and **606** and period **607** on the next (second) line **410** directly beneath the first line **410** of the paragraph **600** as the line **600***b* having a space between the word **602** and the word **604**. As can be seen, the features of the paragraph **600** are otherwise retained. As described above in relation to FIG. **5**, the consideration of additional characteristics, such as the vertical extent of the editing stroke **608** in comparison to the input environment of the page **400** and/or combined characteristics of the proxi-

mate objects may be further used by the present system and method to further verify such a recognition result.

As discussed above in relation to FIG. **6**, the paragraph **600** is split over two lines with the provision of a space between the lines. Alternatively, however the split may be performed with the provision of a line break. The difference between these situations is that upon a change in the display conditions, such as responsive display (e.g., the window size of the UI for the application **112** or the aspect or screen size of the device **100** is changed) or further editing of the content or typesetting of the digital ink into typeset ink, the provision of a space between lines of a paragraph will allow reflow of the digital or typeset ink, e.g., words of paragraphs are flowed to adjacent lines to respect word spacing convention, whereas the provision of a line break preserves the flow as input, such that reflow is not allowed across the broken lines.

FIG. **7**A shows an example in which a line break instead of space is provided upon the breaking of a line. In FIG. **7**A, a single line paragraph **700** of several words, "Break" **702**, "a" **704**, "longer" **705**, and "line" **706**, as defined by the spaces therebetween, and a period **707** after the word **706**, has been input. An editing gesture **708** is detected as input as a top-to-bottom vertical line in a single stroke in the displayed position shown. The editing stroke **708** is detected by the HWR system **114** as passing between the words **702** and **704**. Like the example of FIG. **6**, this relative location of the editing stroke **608** to the (recognized) text **700**, e.g., in a space between words, provides one of the above-described characteristics of the editing gesture **708** whereas the top-to-bottom writing direction provides another characteristic. Consideration of these characteristics alone may result in a determination that the editing gesture **708** is intended by the user to break or split the single line of the paragraph **700** at the detected location into two lines. This recognition result is shown in FIG. **7**B in which the paragraph **700** has been broken into two lines **700***a* and **700***b* having a line break between the word **702** and the word **704**, such that the word **702** remains on the original (first) line **410** and the remaining words **704**, **705** and **706** and the period **707** are moved the next (second) line **410** below the first line **410**. As can be seen, the features of the paragraph **700** are otherwise retained. As described above in relation to FIG. **6**, the consideration of additional characteristics, such as the vertical extent of the editing stroke **708** in comparison to the input environment of the page **400** and/or combined characteristics of the proximate objects may be further used by the present system and method to further verify such a recognition result.

The determination of whether to provide a space or line break in scenarios such as in FIGS. **6** and **7** could be provided through consideration of further characteristics of the input, such as the paragraph dimensions and/or gesture position, such as the overall width of the paragraph **600** being reduced in FIG. **6** leading to a consideration that the user's intent was to input an additional object block next to the paragraph **600** (e.g., an image block), such that allowance of reflow is appropriate, whereas in FIG. **7** the second line **410** is provided with more content than the first line **410** after the breaking operation (see FIG. **7**B) leading to the consideration that the user intended a layout change of the paragraph **700**, such that a line break is appropriate. It is understood that additional and/or alternative characteristics may be used, and may be settable via the UI by users.

FIG. **8** shows an example line break operation performed on the example input of FIG. **4**. In FIG. **8**, a break gesture **800** is detected as input as a top-to-bottom vertical line in a single stroke in the displayed position shown. The editing stroke **800** is detected by the HWR system **114** as passing between elements **422** and **424** on a line **426** of the paragraph **416**, being the period **422** and capitalized word **424**. Like the examples of FIGS. **5** to **7**, this relative location of the editing stroke **800** to the (recognized) paragraph **416**, e.g., in a space between sentences, provides one of the above-described characteristics of the editing gesture **800** whereas the top-to-bottom writing direction provides another characteristic. Consideration of these characteristics alone may result in a determination that the editing gesture **800** is intended by the user to break or split the paragraph **416** at the detected location. Depending on context of this recognition and/or settings of the application **112**, this recognition result may cause the provision of a space or a line break at the detected break position, as discussed above. The provision of either of these layout operations may cause different actions on other parts of the content.

For example, if the application **112** provides a line break in determination of the recognition result, which means that text reflow is not allowed, then any downstream content of the broken paragraph may similarly be non-reflowed during the break operation. This is shown in FIG. **8**B in which the paragraph **416** has been broken so that the period **422** remains on the original (first) line **410** (together with the preceding words) and the word **424** (and subsequent word **428** and comma **430**) is moved onto the subsequent (second) line **410** below as a new line **432** of the paragraph **416**, with a line break (not shown) provided between the lines **426** and **432**. Consequently, the text of the paragraph **416** on the second line **410** (a line **434**) in the original layout of the input **400** shown in FIG. **8**A is moved to the next (third) line **410** below the second line **410**, with consequential and similar movement of all remaining lines of the paragraph **416** and the heading **418** and paragraph **420**, as shown in FIG. **8**B (e.g., it can be seen that a last line **436** of text of the paragraph **420** as displayed in FIG. **8**A is scrolled out of the display **104**).

On the other hand, if the application **112** provides a space in determination of the recognition result, which means that text reflow is allowed, then any downstream content of the broken paragraph may similarly be reflowable during the break operation. This is shown in FIG. **8**C in which the paragraph **416** has been split so that the period **422** remains on the first line **410** (together with the preceding words) and the word **424** (and subsequent word **428** and comma **430**) is moved onto the second line **410** below, with a space (not shown) provided between the line **426** and the content on the second line **410**. However, unlike the example of FIG. **8**B, the moved content does not create a new line, instead this content (i.e., the words **424** and **428** and comma **430**) is reflowed onto the next line **434** of the paragraph **416** with consequential reflow of some of the content originally in the line **434** onto the third line **410** (a line **438** of the paragraph **416**) and reflow of some of the content originally in the line **438** to accommodate the reflowed upper content. As can be seen, no further reflow of the downstream content (e.g., the heading **418** and paragraph **420**) is required in this case, as the reflowed line **438** of the paragraph **416** accommodates the reflowed and original content.

The above described examples illustrate the breaking of a line of text within a single- or multi-line paragraph using the described top-to-bottom single stroke break gesture. It is considered that this gesture is relatively intuitive to break elements of object blocks for the vertical flow scenario of the left-to-right language input illustrated in the examples. However, it is understood that a single stroke gesture in a

different direction, particularly with horizontal flow languages, multiple stroke gestures, such as a two stroke gesture tracing the line break or carriage return symbol, or non-directional gestures, such as a tap, can also be used. Similar break gestures can also be used in the context of breaking a paragraph rather than a line.

FIG. **9**A shows a multi-line paragraph **900** of several words, "Break" **902** in a first line **900***a* of the paragraph **900** (on the first line **410**) and "a" **904** and "paragraph" **906**, and a period **907** after the word **906**, in a second line **900***b* of the paragraph **900** (on the second line **410** directly beneath the first line **410**), having a line break therebetween. An editing gesture **908** is detected at the line break of the line **900***a* (e.g., to the right of the word **902**) or an editing gesture **910** is detected at the beginning of the line **900***b* (e.g., to the left of the word **904**), with either input as a top-to-bottom vertical line in a single stroke. In either case the editing stroke **908** or **910** is detected by the HWR system **114** as passing between the lines **900***a* and **900***b* of the paragraph **900**.

Like the examples of FIGS. **5** to **8**, this relative location of the editing stroke **908** or **910** to the (recognized) text **900**, e.g., at a line break, provides one of the above-described characteristics of the editing gesture **908** or **910** whereas the top-to-bottom writing direction provides another characteristic. Consideration of these characteristics alone may result in a determination that the editing gesture **908** or **910** is intended by the user to break or split the single paragraph **900** at the detected location into two paragraphs. This recognition result is shown in FIG. **9**B in which the paragraph **900** has been broken into two single line paragraphs **912** and **914** with the (first) paragraph **912** containing the word **902** of the original (first) line **900***a* of the paragraph **900** (on the original first line **410**) and the (second) paragraph **914** containing the words **904** and **906** and the period **907** of the original (second) line **900***b* of the paragraph **900** on the next (third) line **410** directly below the second line **410**, such that the second line **410** is now empty, e.g., free of content between the lines of the paragraph **900**, and the line break is replaced with a paragraph marker. As described above in relation to FIGS. **5** to **8**, the consideration of additional characteristics, such as the vertical extent of the editing strokes **908** or **910** in comparison to the input environment of the page **400** and/or combined characteristics of the proximate objects may be further used by the present system and method to further verify such a recognition result.

As discussed above in relation to FIG. **9**, the multi-line paragraph **900** is split into two paragraphs over an existing line break. Such a split may also be performed over an existing space, as shown in FIG. **10**. FIG. **10**A shows a multi-line paragraph **1000** of several words, "Break" **1002** and 'again' **1003** in a first line **1000***a* of the paragraph **1000** (on the first line **410**) and "a" **1004** and "paragraph" **1006**, and a period **1007** after the word **1006**, in a second line **1000***b* of the paragraph **1000** (on the second line **410** directly beneath the first line **410**), having a space therebetween. An editing gesture **1008** is detected at the space after the word **1003** of the line **1000***a* (e.g., to the right of the word **1003**) or an editing gesture **1010** is detected at the beginning of the line **1000***b* (e.g., to the left of the word **1004**), with either input as a top-to-bottom vertical line in a single stroke. In either case the editing stroke **1008** or **1010** is detected by the HWR system **114** as passing between the lines **1000***a* and **1000***b* of the paragraph **1000**.

Like the example of FIG. **9**, this relative location of the editing stroke **1008** or **1010** to the (recognized) text **1000**, e.g., at an end space, provides one of the above-described characteristics of the editing gesture **1008** or **1010** whereas the top-to-bottom writing direction provides another characteristic. Consideration of these characteristics alone may result in a determination that the editing gesture **1008** or **1010** is intended by the user to break or split the single paragraph **1000** at the detected location into two paragraphs. This recognition result is shown in FIG. **10**B in which the paragraph **1000** has been broken into two single line paragraphs **1012** and **1014** with the (first) paragraph **1012** containing the words **1002** and **1003** of the original (first) line **1000***a* of the paragraph **1000** (on the original first line **410**) and the (second) paragraph **1014** containing the words **1004** and **1006** and the period **1007** of the original (second) line **1000***b* of the paragraph **1000** on the next (third) line **410** directly below the second line **410**, such that the second line **410** is now empty, and the end space of the line **1000***a* is replaced with a paragraph marker. Accordingly, a similar recognition result to that of the example of FIG. **9**B is obtained. As described above in relation to FIG. **9**, the consideration of additional characteristics, such as the vertical extent of the editing strokes **1008** or **1010** in comparison to the input environment of the page **400** and/or combined characteristics of the proximate objects may be further used by the present system and method to further verify such a recognition result.

Further space between separate paragraphs, whether they have been formed by splitting a single paragraph as in the examples of FIGS. **9** and **10** or initially input in that manner, can be provided using similarly located break gestures. FIG. **10**C shows the separate paragraphs **1012** and **1014** of FIG. **10**B with the editing gesture **1008** detected at the paragraph marker after the word **1003** of the paragraph **1012** (e.g., to the right of the word **1003**) or the editing gesture **1010** detected at the beginning of the paragraph **1014** (e.g., to the left of the word **1004**) or an editing gesture **1016** detected in the empty space between the paragraphs **1012** and **1014** (in the second line **410**), with either of these gestures input as a top-to-bottom vertical line in a single stroke. In either case the editing stroke **1008**, **1010** or **1016** is detected by the HWR system **114** as passing between the paragraphs **1012** and **1014**.

As for FIG. **10**A, this relative location of the editing stroke **1008**, **1010** or **1016** to the (recognized) text **1012** and **1014**, e.g., between the paragraphs, provides one of the above-described characteristics of the editing gesture **1008**, **1010** or **1016** whereas the top-to-bottom writing direction provides another characteristic. Consideration of these characteristics alone may result in a determination that the editing gesture **1008**, **1010** or **1016** is intended by the user to add a further space between the paragraph **1012** and **1014** at the detected location. This recognition result is shown in FIG. **10**D in which the (second) paragraph **1014** is moved to the next (fourth) line **410** directly below the third line **410**, such that both the second and third lines **410** are now empty, and within the third line **410** a line break marker is recorded. The provision of this line break, which may be selectable through the application **112**, can be used to provide control of movement of downstream content if content is added to the second or third lines **410**, for example, e.g., if content is added to the second line, this is considered as content added to the paragraph **1012** such that the third line **410** is retained without movement of the paragraph **1014**. As described above, the consideration of additional characteristics, such as the vertical extent of the editing strokes **1008**, **1010** or **1016** in comparison to the input environment of the page **400** and/or combined characteristics of the proximate objects

may be further used by the present system and method to further verify such a recognition result.

FIG. **11** shows an example space addition operation performed on the example input of FIG. **4**. In FIG. **11**, a space gesture **1100** is detected as input as a top-to-bottom vertical line in a single stroke in the displayed position shown. The editing stroke **1100** is detected by the HWR system **114** as passing between the paragraphs **416** and **418** on the empty (first) line **410** therebetween and above the (second) line **410** on which the paragraph **418** is provided. Like the example of FIGS. **10**C and **10**D, this relative location of the editing stroke **1100** to the (recognized) paragraphs **416** and **418**, e.g., in a space between the paragraphs, provides one of the above-described characteristics of the editing gesture **1100** whereas the top-to-bottom writing direction provides another characteristic. Consideration of these characteristics alone may result in a determination that the editing gesture **1100** is intended by the user to add a further space between the paragraphs **416** and **418** at the detected location.

This recognition result is shown in FIG. **11**B in which the paragraph **418** has been moved onto the subsequent (third) line **410** below the second line **410**. Consequently, the paragraph **420** on the (fourth) line **410** below the third line **410** in the original layout of the input **400** shown in FIG. **11**A is moved to the next (fifth) line **410** below the fourth line **410** (e.g., it can be seen that the last line **436** of text of the paragraph **420** as displayed in FIG. **11**A is scrolled out of the display **104**). Depending on context of this recognition and/or settings of the application **112**, this recognition result may cause the provision of a line break at the detected break position with the consequential actions on other parts of the content, as discussed above.

The above described examples illustrate the breaking of a single- or multi-line paragraph into separate paragraphs or spacing of separate paragraphs using the described top-to-bottom single stroke break or space gesture. It is considered that this gesture is relatively intuitive to break object blocks into separate blocks or to space object blocks for the vertical flow scenario of the left-to-right language input illustrated in the examples. However, it is understood that a single stroke gesture in a different direction, particularly with horizontal flow languages, multiple stroke gestures, such as a two stroke gesture tracing the line break or carriage return symbol, or non-directional gestures, such as a tap, can also be used.

The afore-described editing gestures were for performing breaking, splitting or spacing operations in relation to handwritten content. Other editing operations are also possible. FIG. **12**A shows a single line paragraph **1200** of several words, "Join" **1202**, "two" **1204** and "words" **1206**, as defined by the spaces therebetween, and a period **1207** after the word **1206**. An editing gesture **1208** is detected as input as a bottom-to-top vertical line in a single stroke by a user swiping their finger or stylus tip from the bottom to the top in the displayed position shown, for example. That is, the stroke start position is below the stroke end position, in time order, as depicted by the gradual thickening of the stroke from the bottom to the top. The editing stroke **1208** is detected by the HWR system **114** as passing in the space between the words **1204** and **1206**. This relative location of the editing stroke **1208** to the (recognized) text **1200**, e.g., in a space between words, provides one of the above-described characteristics of the editing gesture **1208** whereas the bottom-to-top writing direction provides another characteristic.

Consideration of these characteristics alone may result in a determination that the editing gesture **1208** is intended by the user to join the words **1204** and **1206** together or delete the space at the detected location, and as such the recognition result is as that shown in FIG. **12**B in which the words **1204** and **1206** have been joined into a new word **1209**, namely at the "o" of the word **1204** and the "w" of the word **1206**. As can be seen, the features of the paragraph **1200** are otherwise retained. However, such a simple editing gesture could be confused through the handwriting recognition as being an additionally input character, e.g., the letter "l". In the illustrated case, the probability of this recognition result of the character "l" would perhaps likely be lower than the probability of the editing gesture, 'join', because the language model the HWR system **114** would return a poor result for an "l" character written in the bottom-to-top stroke order, for example. However, the language model could provide a relatively similar probability for the word "twolwords" if the input is an "l" or the word "twowords" if the input is the editing gesture, 'join', for example. Such recognition results depend on the language and context of input and as such recognition that is largely independent of such factors is preferred in order to provide a robust and widely applicable system and method.

Accordingly, the consideration of additional characteristics is made by the present system and method. Like with the top-to-bottom vertical editing gesture, such a characteristic is the vertical extent of the editing stroke **1208** in comparison to the input environment of the page **400**. For example, the stroke **1208** crosses the line **410** on which the paragraph **1200** is written (and also the line **410** thereabove), which is not a typical characteristic of the letter "l", for example, and can therefore be used to relatively decrease the probability of that character being recognized. Alternatively or additionally, such a characteristic is the vertical extent of the editing stroke **1208** in comparison to combined characteristics of the proximate objects defining the geometrical relationship therebetween, such as, the mean, average and/or maximum line patterns of the elements of those objects (e.g., base-line, center-line, top-line, ascender-line, descender-line, matra-line, rukar-line, etc.). For example, the editing stroke **1208** crosses the base-line of the strokes of the words **1204** and **1206** (and also of the paragraph **1200**), which is not a typical characteristic of the letter "l", for example, and can therefore be used to relatively decrease the probability of that character being recognized.

The recognition of such a join or space delete gesture may be further enhanced by allowing or requiring a different gesture type. FIG. **12**C shows the single line paragraph **1200** with an editing gesture **1210** detected as input as a vertical zig-zag line in a single, multi-directional stroke by a user swiping their finger or stylus tip up and down several times in the displayed position shown, for example. The editing stroke **1210** is detected by the HWR system **114** as passing in the space between the words **1204** and **1206**. This relative location of the editing stroke **1210** to the (recognized) text **1200**, e.g., in a space between words, provides one of the above-described characteristics of the editing gesture **1210** whereas the zig-zagging provides another characteristic.

Consideration of these characteristics alone may result in a determination that the editing gesture **1210** is intended by the user to join the words **1204** and **1206** together or delete the space at the detected location, and as such the recognition result is as that shown in FIG. **12**B. In this way, a zig-zag or scratch-out gesture is used rather than the relatively simple, unidirectional editing gesture described in relation to FIG. **12**A. However, the zig-zag gesture **1210**

could also be confused through the handwriting recognition as being an additionally input character, e.g., the letter "w". In the illustrated case, the probability of this recognition result of the character "w" would perhaps likely be lower than the probability of the editing gesture, 'join', because the language model the HWR system **114** would return a poor result for a "w" character written with the depicted slant which is much different to the slant of the other characters of the recognized text **1200**, for example. However, the language model could provide a relatively similar probability for the word "twowwords" if the input is a "w" or the word "twowords" if the input is the editing gesture, 'join', for example. Such recognition results depend on the language and context of input and as such recognition that is largely independent of such factors is preferred in order to provide a robust and widely applicable system and method.

Accordingly, the consideration of additional characteristics is made by the present system and method. Like the bottom-to-top join or space delete gesture, such a characteristic is the vertical extent of the editing stroke **1210** in comparison to the input environment of the page **400**. For example, the stroke **1210** crosses the line **410** on which the paragraph **1200** is written, which is not a typical characteristic of the letter "w", for example, and can therefore be used to relatively decrease the probability of that character being recognized. Alternatively or additionally, such a characteristic is the vertical extent of the editing stroke **1210** in comparison to combined characteristics of the proximate objects defining the geometrical relationship therebetween, such as, the mean, average and/or maximum line patterns of the elements of those objects (e.g., base-line, center-line, top-line, ascender-line, descender-line, matra-line, rukar-line, etc.). For example, the editing stroke **1210** crosses the base-line of the strokes of the words **1204** and **1206** (and also of the paragraph **1200**), which is not a typical characteristic of the letter "w", for example, and can therefore be used to relatively decrease the probability of that character being recognized.

The application **112** may be configured to display the input editing stroke(s) as digital ink, either with a shape as, for example, shown in FIG. **12**A or FIG. **12**C, or some other appropriate rendering until the recognized editing operation is performed or may just perform the editing operation without such display. The display of the detected editing gesture may be used to provide recognition feedback to users with respect to recognition of the gesture itself and the relative location thereof. Otherwise, the eventual editing operation, e.g., the modified display shown in FIG. **12**B, may be used for such feedback.

Similar join or space (or line break) delete gestures can be used to join separate text lines of a paragraph. For example, FIG. **13**A shows a multi-line paragraph **1300** of several words, "Join" **1302** and "two" **1304** in a first line **1300***a* of the paragraph **1300** (on the first line **410**) and "lines" **1306**, and a period **1307** after the word **1306**, in a second line **1300***b* of the paragraph **1300** (on the second line **410** directly beneath the first line **410**), having a space therebetween. An editing gesture **1308** is detected at the end space of the line **1300***a* (e.g., to the right of the word **1304**) or an editing gesture **1310** is detected at the beginning of the line **1300***b* (e.g., to the left of the word **1306**), with either input as a bottom-to-top vertical line in a single stroke. Similarly, in FIG. **13**B an editing gesture **1312** detected as input as a vertical zig-zag line in a single, multi-directional stroke is detected at the end space of the line **1300***a* of the paragraph **1300**. In either case the editing stroke **1308**, **1310** or **1312** is detected by the HWR system **114** as passing between the lines **1300***a* and **1300***b* of the paragraph **900**. This relative location of the editing strokes to the (recognized) text **1300**, e.g., in a space between lines, provides one of the above-described characteristics of the editing gestures whereas the bottom-to-top writing direction or zig-zagging provides another characteristic.

Consideration of these characteristics alone may result in a determination that the editing gesture is intended by the user to join the lines **1300***a* and **1300***b* together or delete the space at the detected location, and as such the recognition result is as that shown in FIG. **13**C in which the lines **1300***a* and **1300***b* have been joined into a new line **1314** containing all of the content of the paragraph **1300**. A similar recognition result is obtained from the joining examples shown in FIGS. **13**D and **13**E, in which the bottom-to-top editing gesture **1308** or **1310** or the zig-zag gesture **1312** are used across a line break, instead of an end space between the two lines **1300***a* and **1300***b* of the paragraph **1300**.

FIG. **14** shows an example line join or space delete operation performed on the example recognition result of FIG. **8**B, in which a line break (not shown) is provided between the lines **426** and **432** of the broken paragraph **416**. In FIG. **14**, a join gesture **1400** is detected as input as a bottom-to-top vertical line in a single stroke in the displayed position shown. The editing stroke **1400** is detected by the HWR system **114** as passing between the period **422** and the word **424** respectively on the lines **426** and **432** of the paragraph **416**. Like the earlier examples, this relative location of the editing stroke **1400** to the (recognized) paragraph **416**, e.g., on a line break between lines of a paragraph, provides one of the above-described characteristics of the editing gesture **1400** whereas the bottom-to-top writing direction provides another characteristic. Consideration of these characteristics alone may result in a determination that the editing gesture **1400** is intended by the user to join the lines **426** and **432** of the paragraph **416** at the detected location with removal of the line break therebetween. Consequently, the word **424** (and the subsequent word **428** and comma **430**) is moved onto the (first) line **410** having the period **422** from the (second) line **410** directly below the first line **410**, the text of the paragraph **416** on third and fourth lines **410** are moved to the second and third lines, respectively, and the heading **418** and paragraph **420** are similarly moved up one line **410** each, such that the display returns to that of FIG. **4**.

The above described examples illustrate the joining of a line of text or deletion of space within a single- or multi-line paragraph using the described bottom-to-top single stroke join or delete space gesture or the zig-zag multi-directional single- or multi-stroke scratch-out gesture. It is considered that these gestures are relatively intuitive to join or to delete space between elements of object blocks for the vertical flow scenario of the left-to-right language input illustrated in the examples. However, it is understood that a single stroke gesture in a different direction, particularly with horizontal flow languages, multiple stroke gestures, or non-directional gestures, such as a tap, can also be used. Similar join or space (or line break) delete gestures can be used to join or remove space (or line breaks) between separate paragraphs.

FIG. **15**A shows a (first) single line paragraph **1500** of one word, "Join" **1502** on the (first) line **410**, the empty (second) line **410** directly under the first line **410**, and a (second) single line paragraph **1503** of several words, "two" **1504** and "paragraphs" **1506**, and a period **1507** after the word **1506**, on the (third) line **410** directly under the second line **410**. Any of an editing gesture **1508** at the end of the first paragraph **1500** (e.g., to the right of the word **1502**), an

editing gesture **1510** at the empty second line **410** between the paragraphs **1500** and **1503**, and an editing gesture **1512** at the beginning of the second paragraph **1503** (e.g., to the left of the word **1504**) are detected, with each input as a bottom-to-top vertical line in a single stroke. Similarly, in FIG. **15**B an editing gesture **1514** detected as input as a vertical zig-zag line in a single, multi-directional stroke is detected at empty second line **410** between the paragraphs **1500** and **1503**. In either case the editing stroke **1508**, **1510**, **1512** or **1514** is detected by the HWR system **114** as being between the first and second paragraphs **1500** and **1503**. This relative location of the editing strokes to the (recognized) text **1500** and **1503**, e.g., in an empty line between paragraphs, provides one of the above-described characteristics of the editing gestures whereas the bottom-to-top writing direction or zig-zagging provides another characteristic.

Consideration of these characteristics alone may result in a determination that the editing gesture is intended by the user to join the paragraphs **1500** and **1503** together at the detected location or delete the empty line therebetween, and as such the recognition result is as that shown in FIG. **15**C in which the paragraphs **1500** and **1503** have been joined into a new (third) paragraph **1516** containing the content of the first paragraph **1500** and the word **1504** of the second paragraph **1503** in a first line **1516***a* (on the first line **410**) and the word **1506** and period **1507** of the second paragraph **1503** in a second line **1516***b* (on the second line **410**), with a space therebetween (e.g., the end space of the line **1516***a*). As such, the content of the joined paragraphs is reflowed so as to produce a single paragraph, since this is considered to be the intent of the user in joining the paragraphs. Alternatively however, a line break could be provided such that reflow does not occur. In that case, a similar recognition result is obtained as in the joining examples shown in FIGS. **13**D and **13**E, in which the bottom-to-top editing gesture **1508**, **1510** or **1512** or the zig-zag gesture **1514** are used across a line break, instead of an end space between the paragraphs **1500** and **1503**.

Similar join or space (or line break) delete gestures can be used to remove additional space (line breaks) between separate paragraphs. For example, FIG. **16**A shows a (first) single line paragraph **1600** of several words, "Two" **1601** and "paragraphs" **1602**, and a period **1603** after the word **1602**, on the (first) line **410**, two empty (second and third) lines **410** directly under the first line **410**, with a line break on the third line **410** (similar to the example of FIG. **10**D), and a (second) single line paragraph **1604** of several words, "Unspace" **1605** and "paragraphs" **1606**, and a period **1607** after the word **1606**, on the (fourth) line **410** directly under the third line **410**. Any of an editing gesture **1608** at the end of the first paragraph **1600** (e.g., to the right of the period **1603**), an editing gesture **1610** at the empty second line **410** or an editing gesture **1612** at the empty third line **410** between the paragraphs **1600** and **1604**, and an editing gesture **1614** at the beginning of the second paragraph **1604** (e.g., to the left of the word **1605**) are detected, with each input as a bottom-to-top vertical line in a single stroke. The editing stroke **1608**, **1610**, **1612** or **1614** is detected by the HWR system **114** as being between the first and second paragraphs **1600** and **1604** separated by two empty lines **410**. This relative location of the editing strokes to the (recognized) text **1600** and **1604**, e.g., in an empty line between paragraphs, provides one of the above-described characteristics of the editing gestures whereas the bottom-to-top writing direction provides another characteristic.

Consideration of these characteristics alone may result in a determination that the editing gesture is intended by the user to remove one of the empty lines between the paragraphs **1600** and **1604** at the detected location, and as such the recognition result is as that shown in FIG. **16**B in which the second paragraph **1604** is moved onto the third line **410**, with the consequential removal of the line break and characteristics of the two paragraphs otherwise retained. Although, not specifically shown, it is understood that a similar recognition result can be obtained with detection of a scratch-out gesture between the paragraphs **1600** and **1604**.

FIG. **17**A shows an example operation to delete an empty line between paragraphs performed on the example recognition result of FIG. **11**B. In FIG. **17**A, a space delete gesture **1700** is detected as input as a bottom-to-top vertical line in a single stroke in the displayed position shown. The editing stroke **1700** is detected by the HWR system **114** as being at one of the two empty lines **410** (the third line **410**) between the paragraphs **416** and **418**. Like the earlier examples, this relative location of the editing stroke **1700** to the (recognized) paragraphs **416** and **418**, e.g., on an empty line between separate paragraphs, provides one of the above-described characteristics of the editing gesture **1700** whereas the bottom-to-top writing direction provides another characteristic. Consideration of these characteristics alone may result in a determination that the editing gesture **1700** is intended by the user to delete one of the empty lines **410** between the paragraphs **416** and **418** at the detected location with removal of the line break (not shown) therebetween. Consequently, the heading **418** is moved onto the previously empty (third) line **410** and the paragraph **420** is consequentially similarly moved up one line **410**, such that the last line **436** of the paragraph **420** reappears and the display returns to that of FIG. **4**.

FIG. **17**B shows an example operation to join, or delete the remaining empty line between, the paragraphs **416** and **418**. In FIG. **17**B, a join or space delete gesture **1702** is detected as input as a bottom-to-top vertical line in a single stroke in the displayed position shown. The editing stroke **1702** is detected by the HWR system **114** as being at the empty line **410** (the second line **410**) between the paragraphs **416** and **418**. Like the earlier examples, this relative location of the editing stroke **1702** to the (recognized) paragraphs **416** and **418**, e.g., on an empty line between separate paragraphs, provides one of the above-described characteristics of the editing gesture **1702** whereas the bottom-to-top writing direction provides another characteristic. Consideration of these characteristics alone may result in a determination that the editing gesture **1702** is intended by the user to join, or delete the empty line **410** between, the paragraphs **416** and **418** at the detected location.

This recognition result is shown in FIG. **17**C in which the single word of the paragraph **418** is reflowed onto the last line **438** of the paragraph **416**, e.g., after a period **440**, to form a new paragraph **442**. Consequentially, the paragraph **420** is moved up two lines **410** (to the third line **410**), leaving the empty (second) line **410** between the paragraphs **420** and **442**, similar to the recognition result example of FIG. **15**C. Depending on context of this recognition and/or settings of the application **112**, this recognition result may cause the provision of a line break at the detected join/delete position rather than reflow as in FIG. **17**C. This alternative recognition result is shown in FIG. **17**D in which the paragraph **418** is moved onto the previously empty (second) line **410** directly beneath the last line **438** of the paragraph **416** (on the first line **410**) to form a new paragraph **444** having the single word of the paragraph **418** on a new last line **446** with a line break (not shown) between the period **440** on the line

438 and the line 446. Consequentially, the paragraph 420 is moved up one line 410 (to the fourth line 410).

The above described examples illustrate the joining of single- or multi-line paragraphs into a single paragraph or deletion of space between separate paragraphs using the described bottom-to-top single stroke join or delete space gesture or the zig-zag multi-directional single- or multi-stroke scratch-out gesture. It is considered that these gestures are relatively intuitive to join object blocks or to delete space between object blocks for the vertical flow scenario of the left-to-right language input illustrated in the examples. However, it is understood that a single stroke gesture in a different direction, particularly with horizontal flow languages, multiple stroke gestures, or non-directional gestures, such as a tap, can also be used.

The afore-described editing gestures were for performing breaking, splitting or spacing operations and joining or space deletion operations in relation to handwritten content. Other editing operations are also possible. FIG. 18A shows a single line paragraph 1800 of several words, "Erase" 1802, "one" 1804 and "char" 1806, as defined by the spaces therebetween, and a period 1807 after the word 1806. An editing gesture 1808 is detected as input as a slanted zig-zag line in a single, multi-directional stroke by a user swiping their finger or stylus tip in a side-to-side slant several times in the displayed position shown, for example. The editing stroke 1808 is detected by the HWR system 114 as being (substantially) on or over a (entire) character, "e" 1804*a*, of the word 1804. This relative location of the editing stroke 1808 to the (recognized) text 1800, e.g., on a single character, provides one of the above-described characteristics of the editing gesture 1808 whereas the zig-zagging provides another characteristic.

Consideration of these characteristics alone may result in a determination that the editing gesture 1808 is intended by the user to delete the character 1804*a* of the word 1804, and as such the recognition result is as that shown in FIG. 18B in which the character 1804*a* is deleted leaving characters, "o" 1804*b* and "n" 1804*c* of the word 1804, forming a new word, "on" 1809. As can be seen, the features of the paragraph 1800 are otherwise retained. This (slanted) zig-zag gesture 1808 is considered as being relatively robust with respect to recognition by the HWR system 114 as a scratch-out deletion gesture, for at least the reasons discussed earlier (e.g., with consideration of the slant, crossing of base-line, higher probability of the word "on" than "onw", etc.). However, a certain amount of precision may be required for use of such a gesture so that more than one character is not erroneously deleted. Accordingly, alternative or additional gestures may be provided for such character deletion operations.

FIG. 18C shows the single line paragraph 1800 with an editing gesture 1810 detected as input as a bottom-to-top slanted line in a single stroke by a user swiping their finger or stylus tip from the bottom to the top from left to right in a slanted direction, for example. That is, the stroke start position is below, and to the right of, the stroke end position, in time order, as depicted by the gradual thickening of the stroke from the bottom to the top and from left to right. FIG. 18D shows the single line paragraph 1800 with an editing gesture 1812 detected as input as a bottom-to-top curly line in a single stroke by a user swiping their finger or stylus tip from the bottom to the top in a spiral to the right, for example. That is, the stroke start position is below, and to the left of, the stroke end position, in time order, as depicted by the gradual thickening of the stroke from the bottom to the top and from left to right. The editing stroke 1810 and the editing stroke 1812 is detected by the HWR system 114 as passing (substantially) over the (entire) character 1804*a* of the word 1804. This relative location of the editing strokes 1810 and 1812 to the (recognized) text 1800, e.g., on a single character, provides one of the above-described characteristics of the editing gestures 1810 and 1812 whereas the left to right and bottom-to-top writing direction of the editing gesture 1810 and the curly writing direction of the editing gesture 1812 each provide another characteristic.

Consideration of these characteristics alone may result in a determination that the editing gestures 1810 and 1812 are each intended by the user to delete the character 1804*a* of the word 1804, and as such the recognition result is as that shown in FIG. 18B. The (slanted) bottom-to-top gesture 1810 is considered as being relatively robust with respect to recognition by the HWR system 114 as a deletion gesture, for at least the reasons discussed earlier (e.g., with consideration of the slant, crossing of base-line, good probability of the word "on", etc.). Further, the (curly) bottom-to-top gesture 1810 is considered as being relatively robust with respect to recognition by the HWR system 114 as a "pig-tail" deletion gesture, for at least the reasons discussed earlier (e.g., with consideration of the slant, curled shape, good probability of the word "on", etc.). However, like the scratch-out gesture, a certain amount of precision may be required for use of such gestures so that more than one character is not erroneously deleted.

The application 112 may be configured to display the input editing stroke(s) as digital ink, either with a shape as, for example, shown in FIG. 18A, 18B or 18C, or some other appropriate rendering until the recognized editing operation is performed or may just perform the editing operation without such display. The display of the detected editing gesture may be used to provide recognition feedback to users with respect to recognition of the gesture itself and the relative location thereof. Otherwise, the eventual editing operation, e.g., the modified display shown in FIG. 18B, may otherwise be used for such feedback.

Similar deletion gestures can be used to delete multiple characters of a word. For example, FIG. 19A shows a multi-line paragraph 1900 of several words, "Erase" 1902 and "several" 1904 in a first line 1900*a* of the paragraph 1900 (on the first line 410) and "characters" 1906, and a period 1907 after the word 1906, in a second line 1900*b* of the paragraph 1900 (on the second line 410 directly beneath the first line 410), having a space therebetween. An editing gesture 1908 is detected (substantially) on or over a (entire) series of characters 1906*a* of the word 1906 input as a vertical zig-zag line in a single, multi-directional stroke. Similarly, in FIG. 19B an editing gesture 1910 detected as input as a left-to-right vertical line in a single stroke is detected (substantially) over the (entire) series of characters 1906*a* of the word 1906. In either case the editing stroke 1908 or 1910 is detected by the HWR system 114 as passing over the series of characters 1906*a* of the word 1906. This relative location of the editing strokes to the (recognized) text 1900, e.g., on a series of characters, provides one of the above-described characteristics of the editing gestures whereas the zig-zagging or left-to-right (horizontal) writing direction provides another characteristic.

Consideration of these characteristics alone may result in a determination that the editing gesture is intended by the user to delete the series of characters 1906*a* of the word 1906, and as such the recognition result is as that shown in FIG. 19C in which the multiple characters 1906*a* are deleted leaving the series of characters, "char" 1906*b*, and the character, "s" 1906*c* of the word 1906, respectively forming

two new words **1912** and **1914**. As can be seen, the features of the paragraph **1900** are otherwise retained and the removed characters are replaced with space(s) rather than reflowing the character **1906***c* to join the characters **1906***b* in the word **1906**. This is done in this example because it is considered that the user intended to delete the multiple characters in order to add new characters. It is understood however that different considerations can be made or set, such as joining the remaining characters of the partially deleted word.

The zig-zag gesture **1908** is considered as being relatively robust with respect to recognition by the HWR system **114** as a scratch-out deletion gesture, for at least the reasons discussed earlier (e.g., with consideration of the slant, crossing of base-line, passage over several characters requiring less precision). Further, the horizontal gesture **1910** is considered as being relatively robust with respect to recognition by the HWR system **114** as a strike-through deletion gesture, as it passes over several characters requiring less precision, is a well known editing gesture for deletion, and has relatively low probability of being recognized as a character by the HWR system **114** in relation to the recognized text **1900**, since it is probably relatively too long to be a hyphen or dash character, for example.

The application **112** may be configured to display the input editing stroke(s) as digital ink, either with a shape as, for example, shown in FIG. **19**A or **19**B, or some other appropriate rendering until the recognized editing operation is performed or may just perform the editing operation without such display. The display of the detected editing gesture may be used to provide recognition feedback to users with respect to recognition of the gesture itself and the relative location thereof. Otherwise, the eventual editing operation, e.g., the modified display shown in FIG. **19**C, may be used for such feedback.

Similar deletion gestures can be used to delete one or more words from a line (or paragraph). For example, FIG. **20**A shows a single-line paragraph **2000** of several words, "Erase" **2002**, "one" **2004** and "word" **2006**, defined by spaces therebetween and followed by a period **2007**. An editing gesture **2008** is detected (substantially) on or over the (entire) word **2006** and the period **2007** input as a vertical zig-zag line in a single, multi-directional stroke. Similarly, in FIG. **20**B an editing gesture **2010** detected as input as a left-to-right horizontal line in a single stroke is detected (substantially) over the (entire) word **2006** and the period **2007**. In either case the editing stroke **2008** or **2010** is detected by the HWR system **114** as passing over the word **2006** and the period **2007**. This relative location of the editing strokes to the (recognized) text **2000**, e.g., on a word, provides one of the above-described characteristics of the editing gestures whereas the zig-zagging or left-to-right (horizontal) writing direction provides another characteristic.

Consideration of these characteristics alone may result in a determination that the editing gesture is intended by the user to delete the word **2006** and the period **2007**, and as such the recognition result is as that shown in FIG. **20**C in which the word **2006** and the period **2007** are deleted leaving the words **2002** and **2004**.

Further, FIG. **21**A shows a single-line paragraph **2100** of several words, "Erase" **2102**, "two" **2104** and "words" **2106**, defined by spaces therebetween and followed by a period **2107**. An editing gesture **2108** is detected (substantially) on or over the (entire) words **2104** and **2106** and the period **2107** input as a vertical zig-zag line in a single, multi-directional stroke. Similarly, in FIG. **21**B an editing gesture **2110** detected as input as a left-to-right horizontal line in a single stroke is detected (substantially) over the (entire) words **2104** and **2106** and the period **2107**. In either case the editing stroke **2108** or **2110** is detected by the HWR system **114** as passing over the words **2104** and **2106** and the period **2107**. This relative location of the editing strokes to the (recognized) text **2100**, e.g., on a word, provides one of the above-described characteristics of the editing gestures whereas the zig-zagging or left-to-right (horizontal) writing direction provides another characteristic.

Consideration of these characteristics alone may result in a determination that the editing gesture is intended by the user to delete the words **2104** and **2106** and the period **2107**, and as such the recognition result is as that shown in FIG. **21**C in which the words **2104** and **2106** and the period **2107** are deleted leaving the word **2102**.

Similar deletion gestures can be used to delete words from multi-line paragraphs. For example, FIG. **22**A shows a multi-line paragraph **2200** of several words, "Erase" **2202** and "several" **2203** in a first line **2200***a* of the paragraph **2200** (on the first line **410**) and "words" **2204**, "at" **2205** and "once" **2206** defined by spaces therebetween and followed by a period **2207**, in a second line **2200***b* of the paragraph **2200** (on the second line **410** directly below the first line **410**), having a space therebetween. An editing gesture **2208** is detected (substantially) on or over the (entire) words **2203**, **2205** and **2206** and the period **2207** input as a vertical zig-zag line in a single, multi-directional stroke. The editing stroke **2208** is detected by the HWR system **114** as passing over the words **2203**, **2205** and **2206** and the period **2207**. This relative location of the editing stroke to the (recognized) text **2200**, e.g., on several words on separate lines, provides one of the above-described characteristics of the editing gestures whereas the zig-zagging provides another characteristic.

Consideration of these characteristics alone may result in a determination that the editing gesture is intended by the user to delete the words **2203**, **2205** and **2206** and the period **2207**, and as such the recognition result is as that shown in FIG. **22**B in which the words **2203**, **2205** and **2206** and the period **2207** are deleted leaving the words **2202** and **2204**, on their respective lines **2200***a* and **2200***b* with a space therebetween. This is done in this example because it is considered that the user intended to delete the multiple words in order to add new words to the multi-line paragraph. It is understood however that different considerations can be made or set. For example, FIG. **22**C shows an alternative recognition result in which the word **2204** is reflowed onto the first line **410** after the word **2202** with a space therebetween forming a new single-line paragraph **2210**. As before, the scratch-out and strike-through gestures are considered as being relatively robust for recognition.

FIG. **23** shows an example operation to delete a word performed on the input example of FIG. **4**. In FIG. **23**A, a strike-through deletion gesture **2300** is detected as input as a left-to-right horizontal line in a single stroke in the displayed position shown. The editing stroke **2300** is detected by the HWR system **114** as being (substantially) on or over a (entire) word **448** in the line **434** of the paragraph **416**. Like the earlier examples, this relative location of the editing stroke **2300** to the (recognized) paragraph **416**, e.g., on a word of the paragraph, provides one of the above-described characteristics of the editing gesture **2300** whereas the left-to-right (horizontal) writing direction provides another characteristic. Consideration of these characteristics

alone may result in a determination that the editing gesture **2300** is intended by the user to delete the word **448** of the paragraph **416**.

This recognition result is shown in FIG. **23**B in which the word **446** is omitted with consequential reflow of the remaining words on the line **434** (to the left margin **404**), reflow up of a word **450** from the line **438** to the line **434** after the other (reflowed) words, and reflow of the remaining word and period on the line **438** (to the left margin **404**). This reflow is performed as a space (not shown) was present between the lines **434** and **438**. As can be seen, as content remains on the line **438** of the edited paragraph **416**, the downstream content (i.e., paragraphs **418** and **420**) is not moved.

Similar deletion gestures can be used to delete one or more lines of text from multi-line paragraphs. For example, FIG. **24**A shows a multi-line paragraph **2400** of several words, "Erase" **2402** and "one" **2403** in a first line **2400***a* of the paragraph **2400** (on the first line **410**) and "line" **2404**, "of" **2405** and "text" **2406** defined by spaces therebetween and followed by a period **2407**, in a second line **2400***b* of the paragraph **2400** (on the second line **410** directly below the first line **410**), having a space therebetween. An editing gesture **2408** is detected (substantially) on or over the (entire) line **2400***b* of the paragraph **2400**, e.g., the words **2404**, **2405** and **2406** and the period **2407**, input as a vertical zig-zag line in a single, multi-directional stroke. Similarly, in FIG. **24**B an editing gesture **2410** detected as input as a left-to-right horizontal line in a single stroke is detected (substantially) over the (entire) line **2400***b*. In either case the editing stroke **2408** or **2410** is detected by the HWR system **114** as passing over the line **2400***b*. This relative location of the editing stroke to the (recognized) text **2400**, e.g., on all words on a line, provides one of the above-described characteristics of the editing gestures whereas the zig-zagging or left-to-right (horizontal) writing direction provides another characteristic.

Consideration of these characteristics alone may result in a determination that the editing gesture is intended by the user to delete the line **2400***b* (containing the words **2404**, **2405** and **2406** and the period **2407**), and as such the recognition result is as that shown in FIG. **24**C in which the second line is omitted leaving the first line **2400***a* with the words **2402** and **2403**.

Further, FIG. **25**A shows a multi-line paragraph **2500** of several lines, a (first) line **2500***a*, a (second) line **2500***b* and a (third) line **2500***c* each respectively containing a series of words **2502**, **2504** and **2506** defined by spaces therebetween and followed by a period **2507** in the third line **2500***c*, each having a space therebetween. An editing gesture **2508** is detected (substantially) on or over the (entire) first and second lines **2500***b* and **2500***c* of the paragraph **2500**, e.g., the series of words **2504** and **2506** and the period **2507**, input as a vertical zig-zag line in a single, multi-directional stroke. Similarly, in FIG. **25**B an editing gesture **2510** detected as input as a left-to-right and bottom-to-top (slanted) line in a single stroke is detected (substantially) over the (entire) first and second lines **2500***b* and **2500***c* of the paragraph **2500**. In either case the editing stroke **2508** or **2510** is detected by the HWR system **114** as passing over both of the lines **2500***b* and **2500***c*. This relative location of the editing stroke to the (recognized) text **2500**, e.g., on several words on separate lines, provides one of the above-described characteristics of the editing gestures whereas the zig-zagging or left-to-right/bottom-to-top (slanted) writing direction provides another characteristic.

Consideration of these characteristics alone may result in a determination that the editing gesture is intended by the user to delete the second and third lines **2500***b* and **2500***c* (containing the series of words **2504** and **2506** and the period **2507**), and as such the recognition result is as that shown in FIG. **25**C in which the second and third lines are omitted leaving the first line **2500***a* with the series of words **2502**. As before, the scratch-out and slanted strike-through gestures are considered as being relatively robust for recognition.

Similar deletion gestures can be used to delete one or more single- and/or multi-line paragraphs. For example, FIGS. **26**A and **26**B respectively show a multi-line paragraph **2600** on which scratch-out and slanted strike-through deletion gestures **2602** and **2604**, respectively, are input (substantially) on or over the (entire) paragraph **2600** as detected by the HWR system **114** leading to the determination that each of the editing gestures is intended by the user to delete the entire paragraph **2600**. Further, FIGS. **27**A and **27**B respectively show a multi-line paragraph **2700** and a single line paragraph **2701** (e.g., separated by one empty one of the lines **410**) on which scratch-out and slanted strike-through deletion gestures **2702** and **2704**, respectively, are input (substantially) on or over the (entire) paragraphs **2700** and **2701** (and the empty line **410** therebetween) as detected by the HWR system **114** leading to the determination that each of the editing gestures is intended by the user to delete both of the paragraphs **2700** and **2701**.

The above described examples illustrate the deletion of single characters (letters, symbols, etc.) from words and lines of text, deletion of single or plural words from one or more lines of text, deletion of single or plural lines of text from one or more single- or multi-line paragraphs, and deletion of one or more single- or multi-line paragraphs using the described (relatively straight) bottom-to-top and left-to-right single slanted stroke deletion gesture, bottom-to-top and left-to-right single curled stroke deletion gesture, left-to-right multi-direction single slanted stroke deletion gesture, and (relatively straight) left-to-right single horizontal deletion gesture. It is considered that these gestures are relatively intuitive to delete elements of object blocks or object blocks themselves for the vertical flow scenario of the left-to-right language input illustrated in the examples. However, it is understood that a single stroke gesture in different directions and form, particularly with horizontal flow languages, multiple stroke gestures, or non-directional gestures, such as a tap, can also be used.

The afore-described editing gestures were for performing breaking, splitting, spacing, joining, and deletion operations in relation to handwritten content. Other editing operations are also possible. FIG. **28** shows a multi-line paragraph **2800** input as handwritten content to the input area **408** generally on the lines of the line pattern **410** and rendered as digital ink on the interface surface **104** of the device, for example. FIG. **29** shows the paragraph **2800** of FIG. **28** with additional handwritten content input and rendered as digital ink **2900**. The additional content **2900** includes an underline **2902** written beneath plural words, "can" **2802** and "replace" **2804** on a (first) line **2806** of the paragraph **2800** and an underline **2904** written beneath several words, "keyboard" **2808** and "and" **2810** on the first line **2806** and a word "mouse" **2812** on a subsequent (second) line **2814** of the paragraph **2800**. The additional content **2900** may be treated as content by the HWR system **114** through recognition of the content being underlines with respect to the recognized text of the paragraph **2800**. The present system and method however provides a mechanism for such "content" to be considered as gestures for editing, as follows.

In the pen and paper context, users might use underlines, or other decorations such as double underlines, circles, ellipses, boxes, coloring, for a part(s) or a whole paragraph to emphasis certain text therein or the whole paragraph itself, or even sections of a document. Accordingly, as the application **112** of the present system and method renders handwritten input, particular the input of note taking, as digital ink and allows continued interaction (such as editing) with that digital ink as handwriting for document creation and sharing, the handwritten decorations are maintained as users have input them, or are beautified as described in detail later. This maintains the intent of the users in providing the decorations whilst the notes/documents are retained in the handwritten (digital ink) form.

The present system and method through the HWR system **114** also provides a mechanism to digitize or convert the handwritten input through the typesetting or fontification of the digital ink into typeset ink. This function is particularly provided to allow the creation of at least near-final documents for communication and sharing, e.g., conversion of a handwritten note which has been edited and formatted in digital ink into a typeset document which could have been produced using keyboard (and associated gesturing devices) alone. In such typeset documents however the use of decorations like those in handwriting may not provide the same emphasizing effect or be suitable for a professional looking document outcome. Accordingly, the present system and method determines additional information to be applied to such decoration input. This is done, for example, so as to allow conversion of these decorations into style changes within the typeset documents and to assist in distinguishing between pure content and editing gestures.

For example, FIG. **30** shows the result of typesetting the paragraph **2800** with the decoration **2900** of FIG. **29** into a typeset paragraph **3000**. As can be seen, the application **112** causes conversion of the digital ink underlines **2902** and **2904** to a bold font for typeset ink versions of the associated words **2802**, **2804**, **2808**, **2810** and **2812** of the paragraph **2800**. In this way, proper emphasis of the underlined words is retained in the typeset document thereby retaining the users' intent. It is understood the styling of bold font is only an example, and other fontification can be provided, as discussed later. Further, the original decoration style may also be maintained if suitable, e.g., with the underlines **2902** and **2904** rendered as typeset underlines. Further, different decorations may be used to apply different styling (discussed later). Such typesetting of input decorations or style changes may be settable via the UI of the application **112**.

In order to properly recognize the decoration input and the association of that input with the content (e.g., digital ink words, paragraphs), the HWR system **114** detects the geometrical and spatial relationships between the existing (recognized) content and newly input (probable) decoration input. Decorations such as shapes, underlines, etc., are easily recognized from the various models of the HWR system **114**, in particular the language model. The associations are determined through the relative locations of such recognized decorations and proximate content in relation to the type of decoration recognized. For example, an underline is a generally horizontal line, generally underneath one or more characters, words or lines of a paragraph, where the underline is related to those elements which it is under.

An issue that arises with the use of handwritten gestures for editing of content, is that similar gestures may be provided for content selection and application control. For example, circling content, e.g., one or more words, could be intended by a user to select that content for interaction, such as copy or cut and paste, or drag & drop or push & drop operations. Another issue that arises with the use of handwritten gestures for editing of content, is that similar input may be provided for content itself, particularly if non-text input is supported. For example, a circle may be input by a user as a circle in a diagram or drawing. Accordingly, it is necessary for the application **112** to be able to distinguish between such events in order to determine and properly adhere to the users' intent.

In available digital note taking applications, separate pens or styli are provided to enable explicit distinction of these events. For example, one pen may be used for text input and simple text editing, such as overwrite, strike-though, etc. on the digital ink, another pen is used for drawing or non-text input, and another pen is used for selection of content and complex content editing, such as decoration. The text, drawing and selection pens are provided with encoding or the like which is detectable by the device and application to differentiate the pens from one another. Alternatively, one or more of these pens may be provided with the ability to exhibit more than one of these attributes, through selection or switching, e.g., through pressing of a button on the pen. The Applicant has found however that users do not appreciate having to use different hardware or to manipulate that hardware for special functions.

The present system and method therefore enables all content and interaction input to be performed using a single input device, such as a user's finger/hand or stylus, without function selection thus handling content, decoration and selection with a single tool. The application **112** differentiates between these to determine user intent. This is done by initially treating all suitable input recognized as probable decoration input as decoration by default and all input recognized as non-text input as non-text by default. The recognition of non-text is provided by the recognition system implemented by the present system and method, for example, in the manner described in U.S. patent application Ser. No. 14/955,155 titled "System and Method of Handwriting Recognition in Diagrams" filed in the name of the present Applicant and Assignee, the entire content of which is incorporated by reference herein.

With respect to the default decoration classification, if subsequent or further (with a suitable time delay of about one to three strokes and/or about 50 ms to about 500 ms) interaction with the content determined to be associated with the recognized decoration is detected by the application **112**, the treatment is changed to that of selection input. For example, when input is detected and recognized as a circle about one or more words, the circle is classified as a decoration by the HWR system **114**, such that whatever styling or editing has been set or is settable for circle decoration is to be applied upon typesetting. Then if a gesture, such as a single or multi-point gesture (e.g., a tap or double tap) is detected as the next input on or about the circle and/or the words surrounded thereby, the input is treated as a selection and the decoration classification is removed by the application **112**. Alternatively, if the next input is further content written or other gesture in a non-associated part of the content, the decoration classification is retained. Manual designation of such gesture input may also be provided, e.g., through the UI, for users.

A further issue that arises with the use of handwritten gestures to provide decoration of digital ink content which is editable, is the proper rendering of the decorations when editing or change of device orientation, window size display or device display occurs which triggers re-display of the

digital ink content, such as with reflow and paragraph resizing, etc. This is described in detail below.

FIG. **31**A shows the paragraph **2800** with the decoration **2900** of FIG. **29**. An editing gesture **3100** is detected as input as a top-to-bottom vertical line in a single stroke in the displayed position shown. The editing stroke **3100** is detected by the HWR system **114** as passing between the words **2808** and **2810**. Like the example of FIG. **6**, this relative location of the editing stroke **3100** to the (recognized) text **2800** and decoration **2900**, e.g., in a space between words, results in a determination that the editing gesture **3100** is intended by the user to break or split the first line **2806** of the paragraph **2800** at the detected location into two lines. This recognition result is shown in FIG. **31**B in which the word **2810** is moved to the second line **2814**, with consequential movement of the word **2812** onto a subsequent (third) line **2816** of the paragraph **2800** and movement of downstream content onto the next lines. This is done because the paragraph **2800** has a line break (not shown) between the lines **2814** and **2816**. Alternatively, the split word **2810** may be reflowed onto the line **2814** with consequential reflow of the word **2812** on that line **2814**. In either case, since the words **2808**, **2810** and **2812** are associated with the underline **2904** editing of the underline **2904** is also required.

The underline **2904** is actually input in two sections or portions, a (first) portion **2904***a* on the line **2806** and a (second) portion **2904***b* on the line **2814**. The underline portion **2904***b* is simply moved along with the associated word **2812**, as shown in FIG. **31**B. However, the underline portion **2904***a* is associated with both of the words **2808** and **2810** and is therefore broken by the editing gesture **3100**. Accordingly, the underline portion **2904***a* is segmented (for example by the segmenter of the HWR system **114**) into a (first) segment **3102** and a (second) segment **3104**. The first segment **3102** contains all of the digital ink of the underline portion **2904***a* to the left of the location at which the editing gesture **3100** crosses the underline portion **2904***a*, and is therefore retained on the line **2806** of the paragraph **2800**, with retention of the relative position to the word **2808**. The second segment **3104** contains all of the digital ink of the underline portion **2904***a* to the right of the location at which the editing gesture **3100** crosses the underline portion **2904***a*, and is therefore moved onto the line **2814** of the paragraph **2800**, with retention of the relative position to the word **2810**.

As can be seen, the resultant display of the first segment **3102** of the underline portion **2904***a* may not be completely satisfactory to the user, since the underline now projects past the underlined word **2808**. In the present example, this may not be overly problematic, but it can be understood that situations may arise when the segmented decoration is not satisfactorily displayed, examples of which are described later. It is possible to configure the segmentation of the decoration so as to omit parts of the digital ink so that the edited display is more satisfactory. For example, the first segment **3102** could be 'cut' so that the right most extent does not project past the last character of the underlined word **2808**. However, the proper extent of such omission may be difficult to ascertain and apply in a general manner to different types of content and decoration. This is particularly the case when the handwriting rendered as the digital ink is not very neat, e.g., it is sketchy, wobbly, wavy, etc., such that cutting the digital ink in an arbitrary position which is not related to the form of the digital ink itself, could cause issues with the legibility of the resultant edited digital ink.

An alternative operation is to re-scale the digital ink rather than cut. However, such rescaling could also cause issues with the resultant scaled digital ink being too small or too large, or imperfections in the handwriting being emphasized. A possible process for scaling digital ink in a sensible fashion is described in U.S. patent application Ser. No. 14/955,174 titled "System and Method for Handwritten Diagram Connectors" filed in the name of the present Applicant and Assignee, the entire content of which is incorporated by reference herein.

The present system and method further provides an intermediary step between the digital ink and full fontification into typeset ink. This intermediary step is beautification. In the beautification process the application **112** and/or HWR system **114** determines extremum, average and mean characteristics of the digital ink and applies smoothing and normalization similar to the preprocessing **116** of the HWR system **114**. The result is beautified digital ink which retains the 'raw' quality of handwriting with increased legibility and regularity. Such beautification may be a selectable setting of the application **112**. With respect to the above discussed issues with editing the 'raw' digital ink of decorations, the present system and method is able to apply beautification to the decorations, either together with the digital ink content or individually, to minimize the noted issues. This is because, the beautified decorations are regular by nature, and so can be cut, resized, scaled, etc., in controllable, known and expected ways. Examples of editing operations on beautified decorations are now described.

FIG. **32** shows the paragraph **2800** as depicted in FIG. **29** but with the decoration **2900** rendered in beautified digital ink decoration **3200**. Accordingly, the underline **2902** is converted into a 'near' typeset underline **3202** and the underline **2904** is converted into a 'near' typeset underline **3204**, in the associated portions **3204***a* and **3204***b*. Alternatively, the beautified version may be of full typeset quality. For the present example, i.e., relatively thin underlines, the difference between these versions may be marginal. However, for more stylistic decorations, such as two-dimensional shapes with sketch-like strokes, the beautified digital ink is considered to more closely reflect the intended handwritten form of the document for continued working prior to finalizing as a fully digitized document.

FIG. **33**A shows the paragraph **2800** with the decoration **3200** of FIG. **32**. An editing gesture **3300** is detected as input as a top-to-bottom vertical line in a single stroke in the displayed position shown. The editing stroke **3300** is detected by the HWR system **114** as passing between the words **2802** and **2804**. Like the example of FIG. **31**A, this relative location of the editing stroke **3300** to the (recognized) text **2800** and decoration **3200**, e.g., in a space between words, results in a determination that the editing gesture **3300** is intended by the user to break or split the first line **2806** of the paragraph **2800** at the detected location into two lines. This recognition result is shown in FIG. **33**B in which the word **2804** is moved to the second line **2814**, with consequential movement of the remaining words of the first line **2806** onto the second line **2814** (including the words **2808** and **2810** and a (first) portion **3204***a* of the underline **3204**) and reflow of the word **2812** on that line **2814**. This reflow is shown to illustrate the alternative to the line break operation result of FIG. **31**B. As can be seen the other parts of the paragraph **2800** remain the same.

This editing operation results in re-rendering of both of the underlines **3202** and **3204** as follows. The underline **3202** is segmented (for example by the segmenter of the HWR system **114**) into a (first) segment **3302** and a (second)

segment **3304**. The first segment **3302** contains the beautified digital ink of the underline **3202** to the left of the location at which the editing gesture **3300** crosses the underline **3202** contained to and beneath the word **2802** only. That is, the beautified digital ink of the underline **3202** from the right most character extent of the word **2802** to the location at which the editing gesture **3300** crosses the underline **3202** is omitted. This is a simpler process for the beautified digital ink than the 'raw' digital ink, as discussed above, as the beautified digital ink has an reasonably invariant rendering along the length of the underline, and so can be cut at an arbitrary position without loss of quality or meaning. The second segment **3304** contains the beautified digital ink of the underline **3202** to the right of the location at which the editing gesture **3300** crosses the underline **3202**.

For the underline **3204**, the first portion **3204***a* and the second portion **3204***b*, are joined on the line **2814** of the paragraph **2800** to form a new beautified digital ink underline **3306** of the combined length of the first and second portions **3204***a* and **3204***b* and relative position to the words **2808**, **2810** and **2812**. This is a simpler process for the beautified digital ink than it would be for the 'raw' digital ink, as the beautified digital ink has an reasonably invariant rendering along the length of the underline, and so can be joined at an arbitrary position without loss of quality or meaning, whereas the underline portions **2904***a* and **2904***b* of the 'raw' digital ink underline **2904** may require significant adjustment to allow joining, which may not be suitably applied in a general fashion.

As can be seen, the resultant display in FIG. **33**B retains clear indication of the original decorations on the content. FIG. **34** shows the result of typesetting the paragraph **2800** with the decoration **3300** of FIG. **33**B into a typeset paragraph **3400**. As can be seen, the application **112** causes conversion of the beautified digital ink underlines **3302**, **3304** and **3306** to a bold font for typeset ink versions of the associated words **2802**, **2804**, **2808**, **2810** and **2812** of the paragraph **2800**, in a manner similar to that of the example of FIG. **30**. In this way, proper emphasis of the underlined words is retained in the typeset document thereby retaining the users' intent.

The provision of the beautified decorations can also assist in the sensible re-display of the decorated content upon other editing operations. FIG. **35**A shows the paragraph **2800** with the decoration **3200** of FIG. **32**. An editing gesture **3500** is detected as input as a left-to-right horizontal line in a single stroke in the displayed position shown. The editing stroke **3500** is detected by the HWR system **114** as passing (substantially) on or over the (entire) word **2810**. Like the example of FIG. **20**B, this relative location of the editing stroke **3500** to the (recognized) text **2800** and decoration **3200**, e.g., over a word, results in a determination that the editing gesture **3500** is intended by the user to delete the word **2810** from the paragraph **2800**. This recognition result is shown in FIG. **35**B in which the word **2810** is omitted.

This editing operation results in re-rendering of the underline **3204** as follows. The underline portion **3204***a*, which is beneath the word **2810** in FIG. **35**A, is segmented (for example by the segmenter of the HWR system **114**) into a segment **3502** which contains the beautified digital ink of the underline **3204** contained to and beneath the word **2808** only. That is, the beautified digital ink of the underline **3204** from the right most character extent of the word **2808** is omitted. As can be seen the other parts of the paragraph **2800** remain the same. This is a simpler process for the beautified digital ink than the 'raw' digital ink, for at least the reasons discussed above. In the alternative, the whole of the underline portion **3204***a* may be retained if it is considered that the deletion operation was intended by the user for the insertion of new content in place thereof, which would retain the underlining.

Further, FIG. **35**C shows the paragraph **2800** with the modified decoration **3200** of FIG. **35**B. The input of further handwritten content is detected as input to the right of the (decorated) word **2808** on the line **2806** of the paragraph **2800**, and is displayed as digital ink content **3504** having been recognized by the HWR system **114** as the words "and/or". As the new content is detected as positioned between the decorated words **2808** and **2812**, this editing operation results in re-rendering of the underline **3504** as follows.

The underline **3504**, which is beneath the word **2808** in FIG. **35**C, is extended from its rightmost extent to the rightmost extent underneath the added content **3504** so as to form a new beautified digital ink underline **3506**, as shown in FIG. **35**D. As can be seen the other parts of the paragraph **2800** remain the same. This is a simpler process for the beautified digital ink than the 'raw' digital ink, for at least the reasons discussed above. In the alternative, the underlining may be retained without extension underneath the new content, whereby if the intent of the user was to have the new content similarly underlined, new decoration input could be made, at which point the application **112** would likely join the new underlining with the underline **3504** to form the new underline **3506**.

The above-described decoration examples are in relation to a relative simple form of decoration, namely underlining, with a single line in a single stroke. As discussed earlier, more complex forms of decoration are possible, including two-dimensional decoration such as circles, ellipses, boxes, which surround or are proximate the handwritten content.

FIG. **36** shows a single line paragraph **3600** of several words, "Break" **3602**, "a" **3604** and "line" **3606**, as defined by the spaces therebetween, and a period **3607** after the word **3606**, similar to FIG. **6**A. However, unlike FIG. **6**A, a hand-drawn or -written decoration **3608** is detected by the HWR system **114** as located around the words **3602**, **3604** and **3606**, and the period **3607**, is recognized as an ellipse and rendered in 'raw' digital ink, for example. FIG. **37**A shows an editing gesture **3700** detected as input as a top-to-bottom vertical line in a single stroke in the displayed position shown. The editing stroke **3700** is detected by the HWR system **114** as passing between the words **3602** and **3604**. Like the example of FIG. **6**, this relative location of the editing stroke **3700** to the (recognized) text **3600**, e.g., in a space between words, provides one of the above-described characteristics of the editing gesture **3700** whereas the top-to-bottom writing direction provides another characteristic.

Consideration of these characteristics alone may result in a determination that the editing gesture **3700** is intended by the user to break or split the single line of the paragraph **3600** at the detected location into two lines. This recognition result is shown in FIG. **37**B in which the paragraph **3600** has been broken into two lines **3600***a* and **3600***b* of text with the word **3602** on the original (first) line **410** of the paragraph **3600** as the line **3600***a* and the words **3604** and **3606** and period **3607** on the next (second) line **410** directly beneath the first line **410** of the paragraph **3600** as the line **3600***b* having a space between the word **3602** and the word **3604**. As can be seen, the features of the paragraph **3600** are otherwise retained. As described above in relation to FIG. **6**, the consideration of additional characteristics, such as the vertical extent of the editing stroke **3700** in comparison to

the input environment of the page **400** and/or combined characteristics of the proximate objects may be further used by the present system and method to further verify such a recognition result. In any case, since the single line is split into two lines editing of the associated ellipse **3608** is also required so that the decoration continues to surround all of the (previously surrounded) content. This is shown in FIG. **37**B in which the ellipse is re-sized and/or scaled into a new 'raw' digital ink decoration **3702** to surround the edited or re-displayed content of the paragraph **3600**.

As can be seen, the resultant display of the decoration **3702** may not be completely satisfactory to the user, as it is quite different to the original decoration **3608**. In the present example, this may not be overly problematic, but it can be understood that situations may arise when the re-sized decoration is not satisfactorily displayed, examples of which are described later. It is possible to configure the re-sizing or scaling of the decoration so that the resultant scaled digital ink is not too small or too large, or emphasizes imperfections in the handwriting, as discussed earlier with the incorporated by reference U.S. patent application Ser. No. 14/955,174 titled "System and Method for Handwritten Diagram Connectors".

FIG. **38** shows the result of typesetting the paragraph **3600** with the decoration **3702** of FIG. **37**B into a typeset paragraph **3800**. As can be seen, the application **112** causes conversion of the digital ink ellipse **3702** to a bold font for typeset ink versions of the associated content **3602** to **3607** of the paragraph **3600**. In this way, proper emphasis of the decorated (circled) content is retained in the typeset document thereby retaining the users' intent. It is understood the styling of bold font is only an example, and font changes or other fontification or styling can be provided, as discussed later. Further, different decorations may be used to apply different styling (discussed later). Such typesetting of input decorations or style changes may be settable via the UI of the application **112**.

FIG. **39** shows the paragraph **2800** of FIG. **28** with additional handwritten content input and rendered as digital ink **3900**. The additional content **3900** includes a ellipse written to surround plural words, "documents" **2818**, "with" **2820**, and "clean" **2822** on a (first) line **2824** of the paragraph **2800**. The additional content **3900** may be treated as content by the HWR system **114** through recognition of the content being a non-text element, e.g., an ellipse, with respect to the recognized text of the paragraph **2800**. However, as discussed above the present system and method provides a mechanism for such "content" to be considered as further gestures for editing, such as the styling of the decorated content to be different than the styling of the non-decorated content in the typeset (or beautified) version of the content.

FIG. **40**A shows the paragraph **2800** with the decoration **3900** of FIG. **39**. An editing gesture **4000** is detected as input as a top-to-bottom vertical line in a single stroke in the displayed position shown. The editing stroke **4000** is detected by the HWR system **114** as passing between the words **2820** and **2822**. Like the example of FIG. **31**, this relative location of the editing stroke **4000** to the (recognized) text **2800** and decoration **3900**, e.g., in a space between words, results in a determination that the editing gesture **4000** is intended by the user to break or split the first line **2824** of the paragraph **2800** at the detected location into two lines. This recognition result is shown in FIG. **40**B in which the word **2822** is moved to a (second) line **2826**, with consequential movement of the content on the second line **2826** to a subsequent line **410**. In this way, a line break is provided after the moved word **2822** on the line **2826**. Alternatively, the recognition result as shown in FIG. **40**C may be performed in which the word **2822** is moved onto the second line **2826**, with consequential reflow of the content on the second line **2826**. In either case, since the words **2818**, **2820** and **2822** are associated with the ellipse **3900** editing of the decoration **3900** is also required so that the decoration continues to surround all of the (previously surrounded) content. This is also shown in FIGS. **40**B and **40**C, where the ellipse is duplicated and re-sized and/or scaled into a new 'raw' digital ink decoration **3902** to surround the words **2818** and **2820** of the paragraph **2800** and a new 'raw' digital ink decoration **3904** to surround the word **2822** of the paragraph **2800**.

As can be seen, the resultant display of the (split) decoration **3902** and **3904** may not be completely satisfactory to the user, as it is quite different to the original decoration **3900**, such as it not being clear that all of the circled content is to be treated the same for subsequent styling, for example. Accordingly, as with the one-dimensional decorations described above, e.g., underlines, the two-dimensional decorations may also be beautified by the application **112** and/or HWR system **114** determining extremum, average and/or mean characteristics of the digital ink and applying smoothing and normalization similar to the preprocessing **116** of the HWR system **114**.

FIG. **41** shows the single line paragraph **3600** of FIG. **36**. However, unlike FIG. **36**, a beautified digital ink version of the hand-drawn or -written decoration **3608** is rendered by the application **112** and/or the HWR system **114** into a 'near' typeset ellipse **4100** located around the words **3602**, **3604** and **3606**, and the period **3607**. FIG. **42**A shows an editing gesture **4200** detected as input as a top-to-bottom vertical line in a single stroke in the displayed position shown. The editing stroke **4200** is detected by the HWR system **114** as passing between the words **3602** and **3604**. Like the example of FIG. **37**A, this relative location of the editing stroke **4200** to the (recognized) text **3600**, e.g., in a space between words, provides one of the above-described characteristics of the editing gesture **4200** whereas the top-to-bottom writing direction provides another characteristic.

Consideration of these characteristics alone may result in a determination that the editing gesture **4200** is intended by the user to break or split the single line of the paragraph **3600** at the detected location into two lines. This recognition result is shown in FIG. **42**B in which the paragraph **3600** has been broken into two lines **3600***a* and **3600***b* of text with the word **3602** on the original (first) line **410** of the paragraph **3600** as the line **3600***a* and the words **3604** and **3606** and period **3607** on the next (second) line **410** directly beneath the first line **410** of the paragraph **3600** as the line **3600***b* having a space between the word **3602** and the word **3604**. As can be seen, the features of the paragraph **3600** are otherwise retained. As described above in relation to FIG. **37**, the consideration of additional characteristics, such as the vertical extent of the editing stroke **4200** in comparison to the input environment of the page **400** and/or combined characteristics of the proximate objects may be further used by the present system and method to further verify such a recognition result. In any case, since the single line is split into two lines editing of the associated ellipse **4100** is also required so that the decoration continues to surround all of the (previously surrounded) content. This is shown in FIG. **42**B in which the ellipse is re-sized and/or scaled into a new beautified decoration **4202** to surround the edited or re-displayed content of the paragraph **3600**. As can be seen, the beautified decoration **4202** is not resized to surround the

content only, but retains the original size. This does not have to be done however, and resizing can occur. The non-resizing can assist however for the subsequent inclusion of further input into the decorated portion if this is considered the intent of the user for the splitting operation, for example.

The provision of such beautified two-dimensional decorations can assist in the sensible re-display of the content upon an editing operation. For example, FIG. **43** shows the content of FIG. **39** but with the digital ink ellipse **3900** rendered as a beautified digital ink ellipse **4300** surrounding the words **2818**, **2820** and **2822** on the first line **2824** of the paragraph **2800**. FIG. **44** shows the recognition result of FIG. **40**B but with the duplicated and re-sized and/or scaled 'raw' digital ink decorations **3902** and **3904** rendered as a beautified digital ink ellipse **4400** surrounding the words **2818**, **2820** and **2822** on the first and second lines **2824** and **2826** of the paragraph **2800**. In this way, the line break provided after the moved word **2822** on the line **2826** and the 'extended' beautified digital ink decoration **4400** retains clarity that all of the circled content is to be treated the same for subsequent styling, for example.

Similar editing operations on beautified decorations can be carried out to reduce the size of the decoration, rather than extend. For example, FIG. **45**A shows a multi-line paragraph **4500** surrounded by a beautified decoration **4502** and for which a word **4504** is deleted by a detected editing gesture **4506**. The recognition result of the deletion is shown in FIG. **45**B in which the word **4504** is omitted with consequential reflow of the downstream content such that the paragraph **4500** becomes a single line paragraph and the beautified decoration **4502** is reduced in size to only surround the content of the now single line paragraph **4500**.

FIG. **46** shows an example decoration of the example input of FIG. **4**. In FIG. **46**, further input handwriting is detected with the resultant display of the heading **412** surrounded by a 'raw' digital ink ellipse **4600**, the heading **414** underlined by a 'raw' digital ink underline **4602**, words **452** and **454** of the paragraph **416** underlined by a 'raw' digital ink underline **4604**, and the heading **418** underlined by a 'raw' digital ink underline **4606**. FIG. **47** shows the example decoration of FIG. **46** rendered as beautified digital ink. That is, in FIG. **47**, the 'raw' digital ink ellipse **4600** is rendered as a beautified digital ink ellipse **4700**, the 'raw' digital ink underline **4602** is rendered as a beautified digital ink underline **4702**, the 'raw' digital ink underline **4604** is rendered as a beautified digital ink underline **4704**, and the 'raw' digital ink underline **4606** is rendered as a beautified digital ink underline **4706**, with the beautified forms having the substantially same extents as the non-beautified forms.

As discussed earlier, such decorations may be considered by the application and/or HWR system **114**, as indicating certain styling changes when the digital ink content is converted to typeset ink. This styling application is performed irrespective of the display of raw or beautified digital ink prior to the conversion. FIG. **48** shows the content of FIGS. **46** and **47** after typesetting. This can be performed for example in response to detection of a gesture, such as a single- or multi-point gesture, e.g., a tap or double tap, in a specified location or any location of the page **400** and/or the interface surface **104** of the device **100**. As can be seen, in response to the different decorations in the different content areas of the page **400**, the typesetted style is different for different levels of the content. For example, the heading **412** is rendered in large (compared to other sections of the content) and bold font to provide a document or section heading in response to the circle decoration, the headings **414** and **418** are rendered in smaller (compared to section heading **412**) yet larger (compared to other sections of the content) and italic font to provide section or sub-section headings in response to the underline decoration for the isolated single line paragraphs represented by the headings **414** and **418**, and the underlined content **452** and **454** of the paragraph **416** is rendered in bold font in response to the underline decoration for body paragraph content. These applied styles are only examples, and other styling types are possible, such as further hierarchical heading styles, coloring of text and/or background, etc.

Since different decoration types may be provided for different typesetting outcomes, such as different stylings, and since the present system and method is directed to providing as much interaction as possible with the content using handwritten input of content and gestures, the ability to change or edit the decorations themselves may be provided through further handwritten input. For example, overwriting or appending a decoration with a different decoration changes the original decoration, e.g., an underline decoration to apply a header **3** style to a paragraph is appended with a second underline to create a double underline decoration to apply header **2** style. For example, overwriting or appending a decoration with a different decoration cancels all decoration, e.g., an underline from left to right provides a decoration and overwriting with an underline from right to left cancels the decoration, like a "backspace". For example, overwriting or appending a decoration with an editing gesture changes or deletes the decoration, e.g., a scratch-out (substantially or partially) over a decoration removes the decoration, but not the content associated therewith.

The present system and method provide an application which allows users, such as students, academic and working professionals, to take handwritten notes, such as during lectures, meetings and brainstorming sessions, on their computing devices in a meaningful and shareable format akin to word processed documents. The present system and method further allow natural writing to be input since the writing is made on (ruled) lines, and paragraphs and other layout elements are respected responsive to orientation (i.e., portrait or landscape) and device display capacity. Further interaction with the handwritten or typeset text, or non-text content, created in a note, such as editing the content, manipulating the layout of the notes, or converting or adding the notes into a document, can be performed directly within the application itself without the need to import the note into a separate document processing application. As such, the original layout of the handwritten notes is maintained, as is the actual input handwriting itself, the digital ink. Accordingly, annotations, decorations, special layout, etc. is maintained for easy comprehension of the notes taken.

While the foregoing has described what is considered to be the best mode and/or other examples, it is understood that various modifications may be made therein and that the subject matter disclosed herein may be implemented in various forms and examples, and that they may be applied in numerous other applications, combinations, and environments, only some of which have been described herein. Those of ordinary skill in that art will recognize that the disclosed aspects may be altered or amended without departing from the true spirit and scope of the subject matter. Therefore, the subject matter is not limited to the specific details, exhibits, and illustrated examples in this description. It is intended to protect any and all modifications and variations that fall within the true scope of the advantageous concepts disclosed herein.

We claim:

1. A system for editing digital documents on computing devices, each computing device comprising a processor and at least one non-transitory computer readable medium for recognizing handwriting input under control of the processor, the processor configured to:

process a handwriting input, wherein the handwriting input can represent handwriting content to be added to displayed text of previously processed handwriting content or an editing gesture for enacting control on the displayed text of the previously processed handwriting content;

cause display of, on a display interface of a computing device, first digital ink in accordance with document content;

determine whether one or more strokes of the handwriting input correspond to i) the handwriting content to be added to the displayed text of the previously processed handwriting content or ii) one or more editing gestures for enacting control on the displayed text of the previously processed handwriting content by 1) determining characteristics of the handwriting input, 2) determining context of the handwriting input by evaluating a location of the one or more strokes of the handwriting input relative to the displayed text of the previously processed handwriting content, 3) comparing geometric features of the one or more strokes of the handwriting input with respect to the displayed text of the previously processed handwriting content, and 4) considering a language of the displayed text of the previously processed handwriting content; and

cause display of, on the display interface, second digital ink in accordance with the handwriting input.

2. The system of claim 1, wherein the second digital ink is different than the first digital ink.

3. The system of claim 1, wherein the second digital ink is a normalized version of the first digital ink.

4. The system of claim 1, wherein the processor is configured to cause re-display of, on the display interface, the document content as a third digital ink based on the detected editing gestures.

5. The system of claim 4, wherein the third digital ink is different than the first and second digital ink.

6. The system of claim 5, wherein the third digital ink is typeset version of the first digital ink.

7. A method for editing digital documents on computing devices, each computing device comprising a processor and at least one non-transitory computer readable medium for recognizing handwriting input under control of the processor, the method comprising:

processing a handwriting input, wherein the handwriting input can represent handwriting content or an editing gesture for enacting control on previously processed handwriting content;

displaying, on a display interface of a computing device, first digital ink in accordance with document content;

determining whether one or more strokes of the handwriting input correspond to i) the handwriting content to be added to the displayed text of the previously processed handwriting content or ii) one or more editing gestures for enacting control on the displayed text of the previously processed handwriting content by 1) determining characteristics of the handwriting input, 2) determining context of the handwriting input by evaluating a location of the one or more strokes of the handwriting input relative to the displayed text of the previously processed handwriting content, 3) comparing geometric features of the one or more strokes of the handwriting input with respect to the displayed text of the previously processed handwriting content, and 4) considering a language of the displayed text of the previously processed handwriting content; and

displaying, on the display interface, second digital ink in accordance with the handwriting input.

8. The method of claim 7, wherein the second digital ink is different than the first digital ink.

9. The method of claim 7, wherein the second digital ink is a normalized version of the first digital ink.

10. The method of claim 7, comprising re-displaying, on the display interface, the document content as a third digital ink based on the detected editing gestures.

11. The method of claim 10, wherein the third digital ink is different than the first and second digital ink.

12. The method of claim 11, wherein the third digital ink is typeset version of the first digital ink.

13. A non-transitory computer readable medium having a computer readable program code embodied therein, said computer readable program code adapted to be executed to implement a method for recognizing input of editing handwriting to a computing device, the computing device comprising a processor and at least one system non-transitory computer readable medium for recognizing the input under control of the processor, the method comprising:

processing a handwriting input, wherein the handwriting input can represent handwriting content or an editing gesture for enacting control on previously processed handwriting content;

displaying, on a display interface of a computing device, first digital ink in accordance with document content;

determining whether one or more strokes of the handwriting input correspond to i) the handwriting content to be added to the displayed text of the previously processed handwriting content or ii) one or more editing gestures for enacting control on the displayed text of the previously processed handwriting content by 1) determining characteristics of the handwriting input, 2) determining context of the handwriting input by evaluating a location of the one or more strokes of the handwriting input relative to the displayed text of the previously processed handwriting content, 3) comparing geometric features of the one or more strokes of the handwriting input with respect to the displayed text of the previously processed handwriting content, and 4) considering a language of the displayed text of the previously processed handwriting content; and

displaying, on the display interface, second digital ink in accordance with the handwriting input.

14. The non-transitory computer readable medium of claim 13, wherein the second digital ink is different than the first digital ink.

15. The non-transitory computer readable medium of claim 13, wherein the second digital ink is a normalized version of the first digital ink.

16. The non-transitory computer readable medium of claim 13, comprising re-displaying, on the display interface, the document content as a third digital ink based on the detected editing gestures.

17. The non-transitory computer readable medium of claim 16, wherein the third digital ink is different than the first and second digital ink.

18. The non-transitory computer readable medium of claim 17, wherein the third digital ink is typeset version of the first digital ink.

19. The system of claim 1, wherein the editing gesture corresponding to a single multi-directional stroke including erasing two or more lines of text.

* * * * *

EXHIBIT K

US010410316B2

(12) **United States Patent**
Baudry et al.

(10) **Patent No.: US 10,410,316 B2**
(45) **Date of Patent: Sep. 10, 2019**

(54) **SYSTEM AND METHOD FOR BEAUTIFYING DIGITAL INK**

(71) Applicant: **MyScript**, Nantes (FR)

(72) Inventors: **Edgard Baudry**, Nantes (FR); **Bertrand Gourraud**, Nantes (FR)

(73) Assignee: **MyScript**, Nantes (FR)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 124 days.

(21) Appl. No.: **14/978,081**

(22) Filed: **Dec. 22, 2015**

(65) **Prior Publication Data**

US 2017/0161866 A1 Jun. 8, 2017

(30) **Foreign Application Priority Data**

Dec. 8, 2015 (EP) 15290303

(51) **Int. Cl.**
G06T 3/00 (2006.01)
G06F 3/0488 (2013.01)
(Continued)

(52) **U.S. Cl.**
CPC ***G06T 3/0075*** (2013.01); ***G06F 3/04883*** (2013.01); ***G06K 9/00402*** (2013.01); ***G06K 9/222*** (2013.01); ***G06T 7/60*** (2013.01)

(58) **Field of Classification Search**
CPC G06F 3/04883; G06F 3/03545; G06F 3/0488; G06K 2209/01; G06K 9/222; G06K 9/00442; G06K 9/00402; G06K 9/32
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,731,857 A 3/1988 Tappert
5,396,566 A 3/1995 Bruce et al.
(Continued)

FOREIGN PATENT DOCUMENTS

EP 0720084 7/1996

OTHER PUBLICATIONS

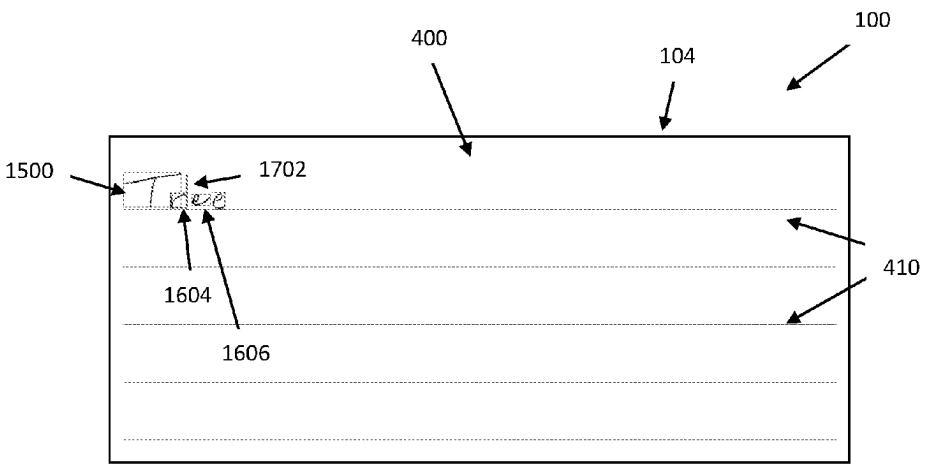
International Search Report for Related Application No. PCT/EP2016/001446, dated Nov. 29, 2016, EPO—International, WPI Data.
(Continued)

Primary Examiner — Jeffrey J Chow
(74) *Attorney, Agent, or Firm* — McDermott Will & Emery LLP

(57) **ABSTRACT**

A system, method and computer program product for use in beautifying digital ink of handwriting input to a computing device are provided. The computing device is connected to an input device in the form of an input surface. A user is able to provide input by applying pressure to or gesturing above the input surface using either his or her finger or an instrument such as a stylus or pen. The present system and method monitors the input strokes. The computing device further has a processor and at least one application for recognizing the handwriting input under control of the processor. The at least one system application is configured to cause display of the digital ink, determine from geometrical information of the handwriting input determined through the recognition of the handwriting input and geometrical information of an alignment structure of the display interface, a structuring transformation of the digital ink, and cause display of transformed digital ink in accordance with the structuring transformation.

9 Claims, 12 Drawing Sheets



(51) **Int. Cl.**

G06K 9/00	(2006.01)
G06T 7/60	(2017.01)
G06K 9/22	(2006.01)

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,850,477	A *	12/1998	Takada	G06F 3/04883 382/186
5,953,735	A *	9/1999	Forcier	G06F 3/0488 715/273
6,408,092	B1	6/2002	Sites	
7,137,077	B2 *	11/2006	Iwema	G06F 3/0481 715/863
7,284,200	B2	10/2007	Bernstein et al.	
7,751,623	B1	7/2010	Simmons et al.	
8,879,845	B2 *	11/2014	Rowley	G06F 3/017 345/179
2007/0003142	A1	1/2007	Simard et al.	
2009/0161958	A1 *	6/2009	Markiewicz	G06F 3/0237 382/186
2010/0309131	A1	12/2010	Clary	
2014/0300609	A1 *	10/2014	Kang	G06F 3/04883 345/467
2016/0147723	A1 *	5/2016	Lee	G06F 17/242 715/268
2016/0292501	A1 *	10/2016	Higashi	G06T 7/73
2017/0103046	A1 *	4/2017	Schroder	G06F 17/214

OTHER PUBLICATIONS

"MyScript—The Power of Handwriting", Dec. 11, 2014, pp. 1-3. URL: https://www.youtube.com/watch?v=yojFiBhPMUo.

International Search Report for related International Patent Application No. PCT/EP2016/002046, dated Feb. 2, 2017.

Improving Offline Handwritten Text Recognition with Hybrid HMM/ANN Models, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 33, No. 4, Apr. 2011.

International Search Report issued for PCT/EP2016/002046 dated Apr. 24, 2017.

Written Opinion of the International Searching Authority issued for PCT/EP2016/002046 dated Apr. 24, 2017.

* cited by examiner

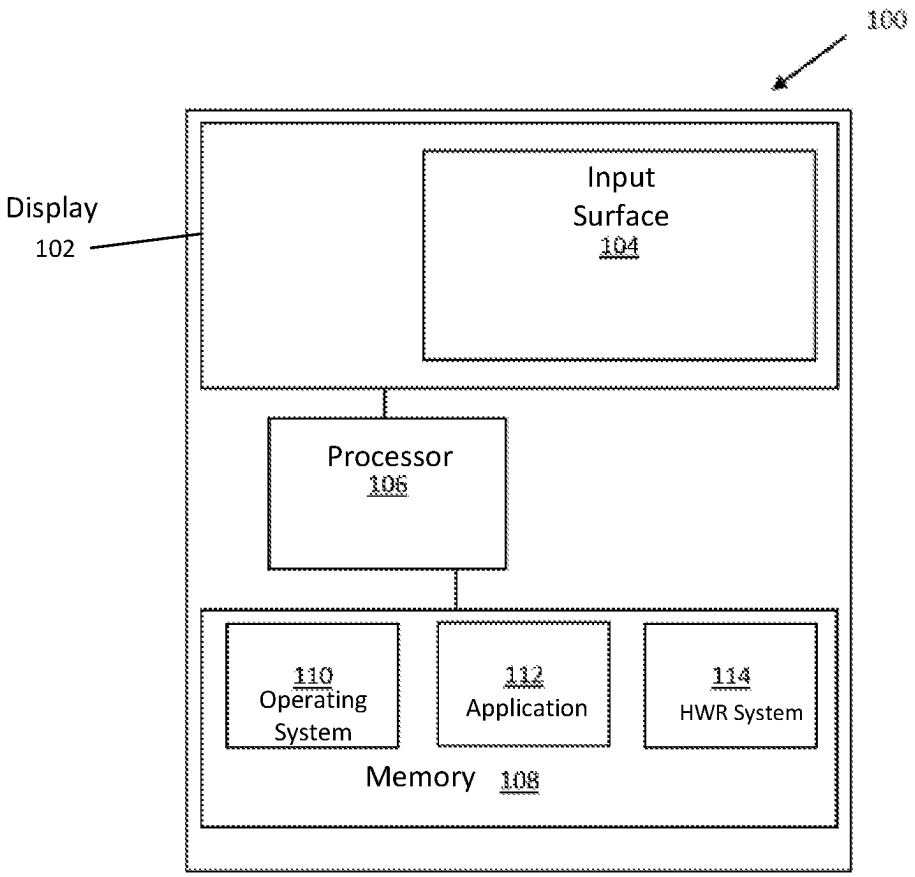

100
Display
102
Input
Surface
104
Processor
106
110
Operating
System
112
Application
114
HWR System
Memory 108

FIG. 1

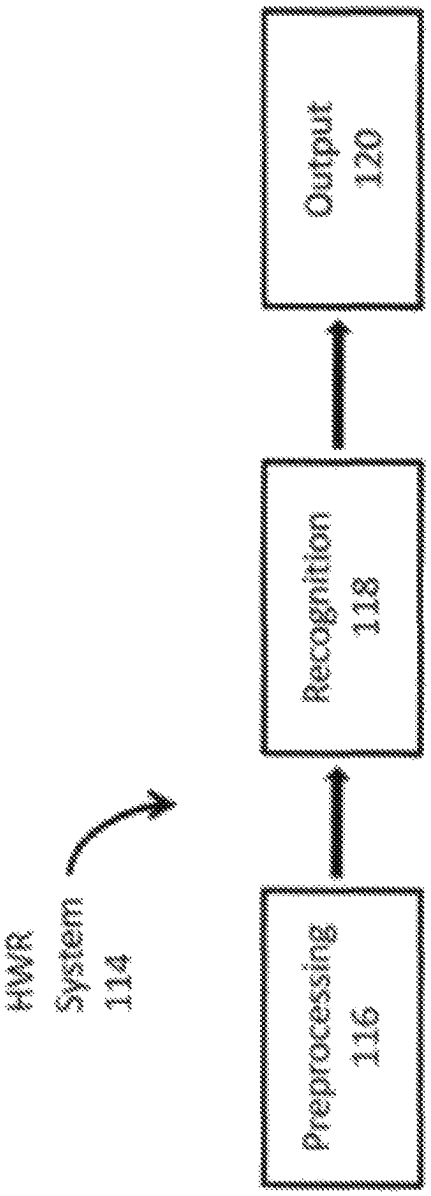

Output
120
Recognition
118
Preprocessing
116
HWR
System
114

FIG. 2

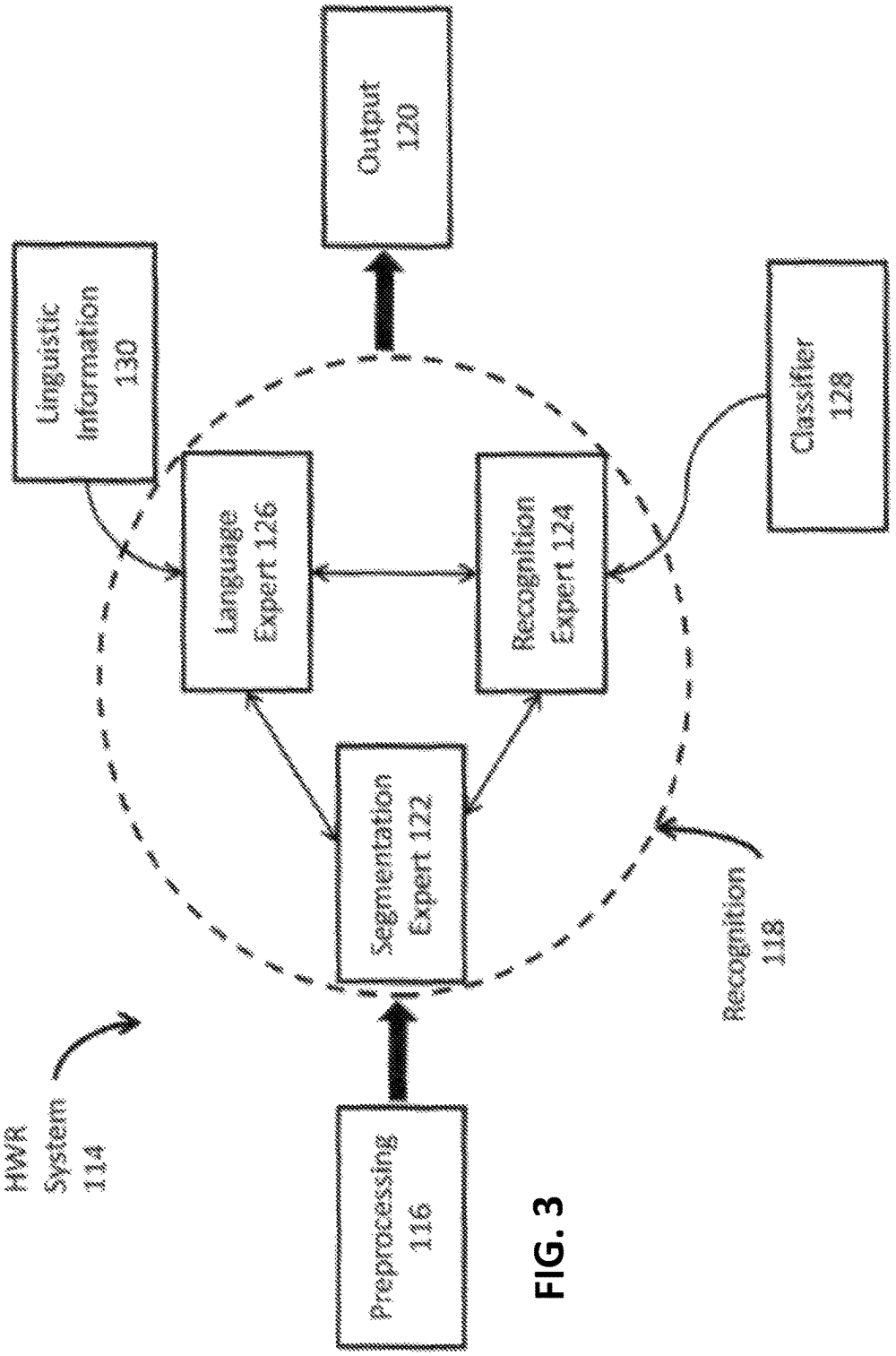

Output 120
Linguistic Information 130
Classifier 128
Language Expert 126
Recognition Expert 124
Segmentation Expert 122
Recognition 118
HWR System 114
Preprocessing 116

FIG. 3

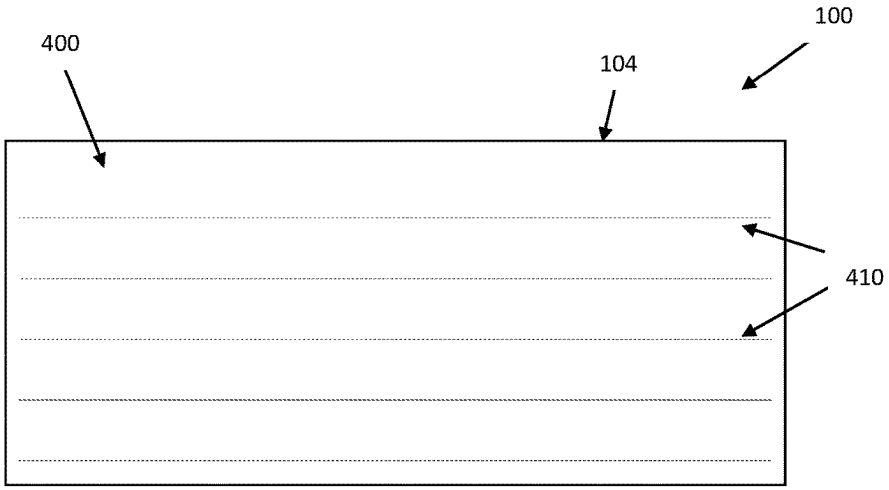

400
104
100
410

FIG. 4

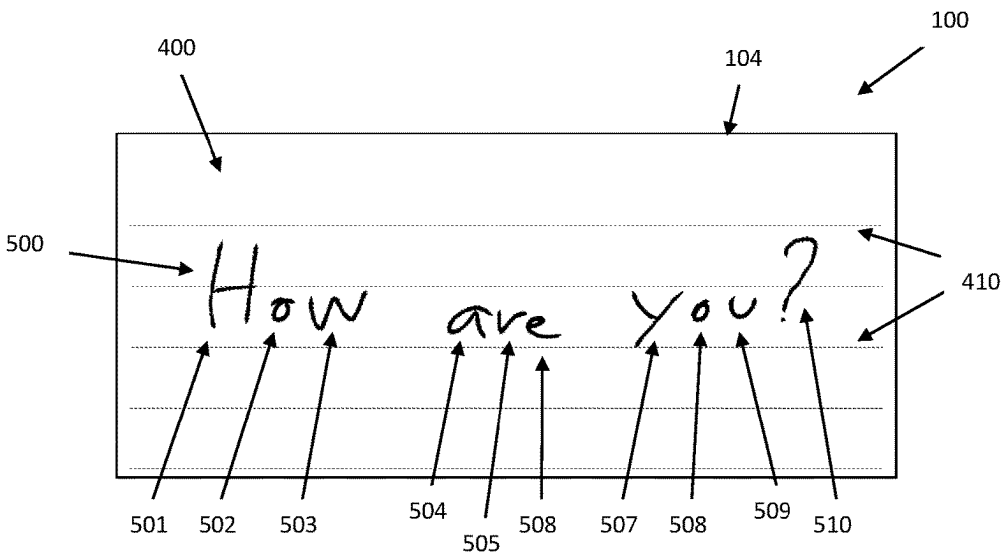

400
104
100
500
How are you?
410
501
502
503
504
505
508
507
508
509
510

FIG. 5

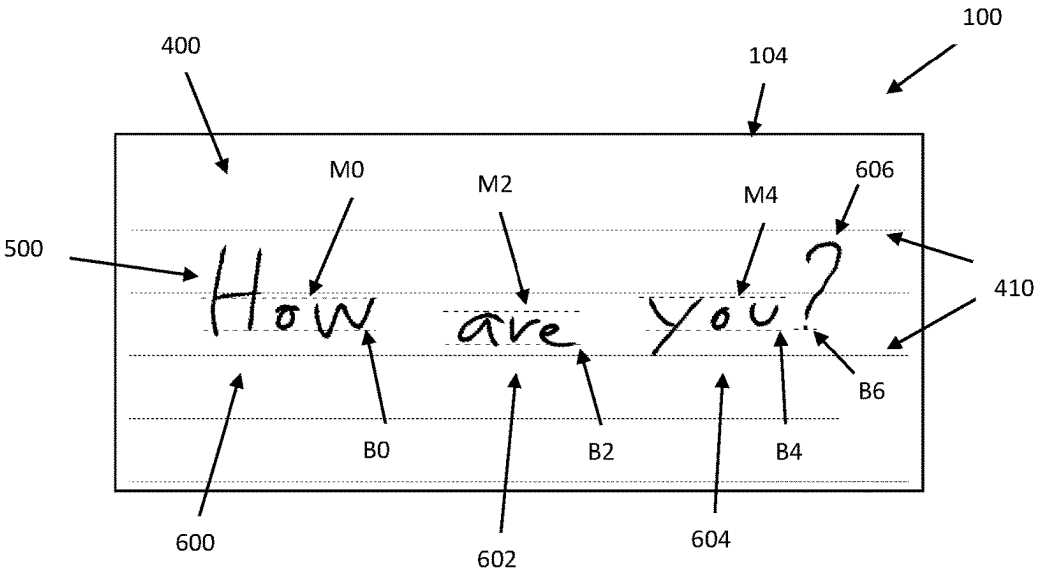

100
400
104
M0
M2
M4
606
500
How are you?
410
B6
B0
B2
B4
600
602
604

FIG. 6

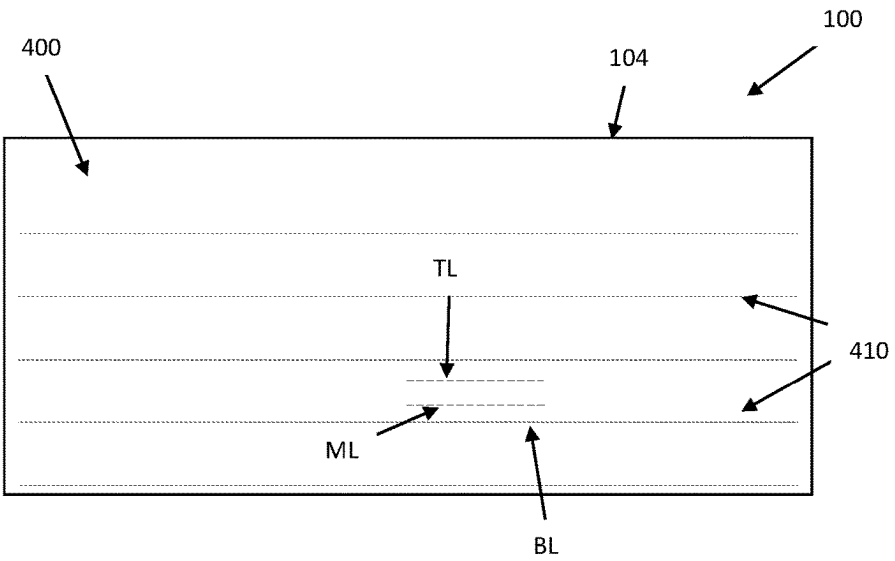

100
400
104
TL
410
ML
BL

FIG. 7

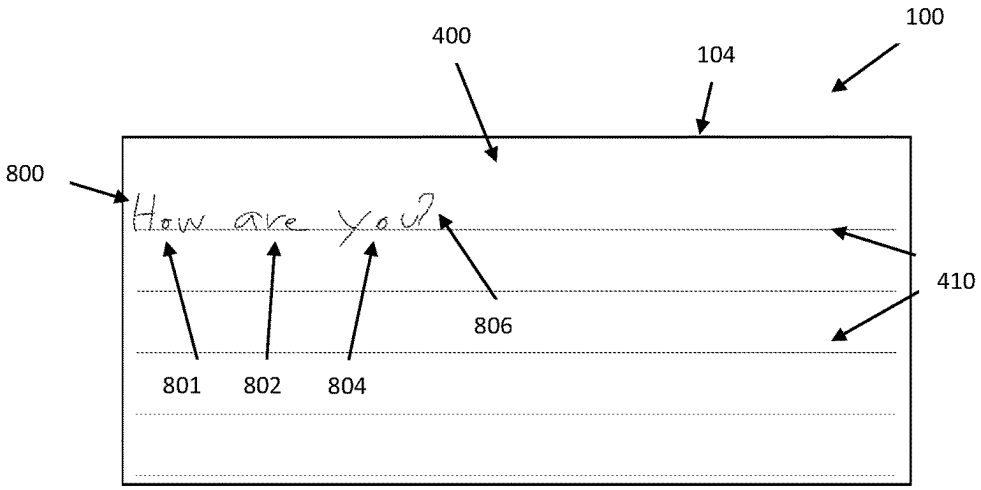

400
104
100
800
How are you?
410
806
801
802
804

FIG. 8

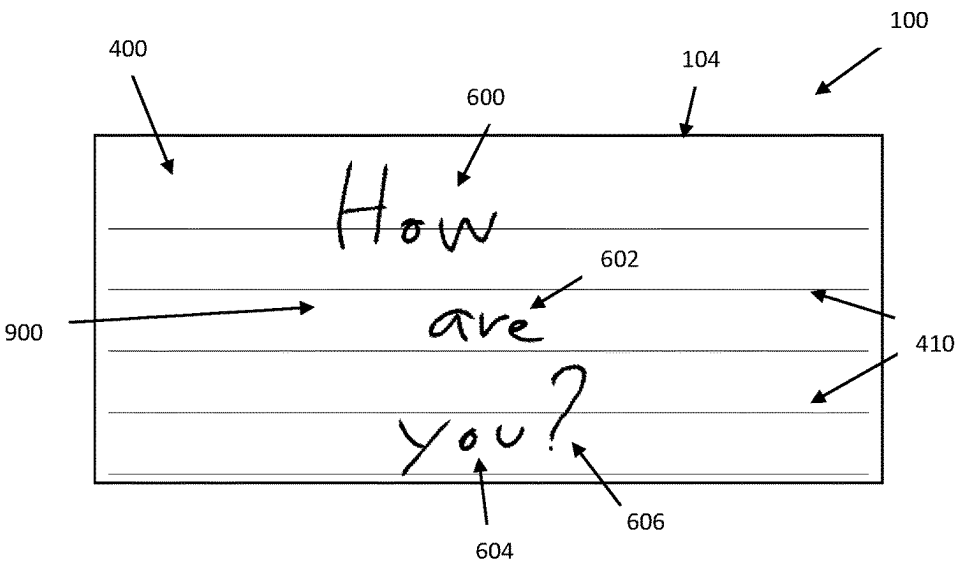

100
400
104
600
How
602
900
are
410
you?
606
604

FIG. 9

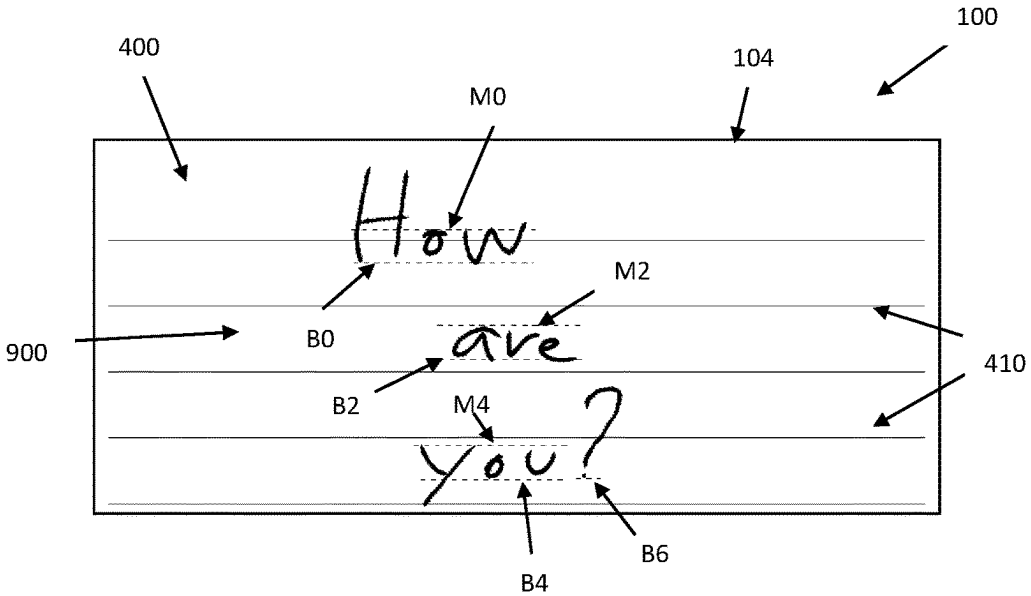

400
100
104
M0
How
M2
B0
are
900
410
B2
M4
you?
B6
B4

FIG. 10

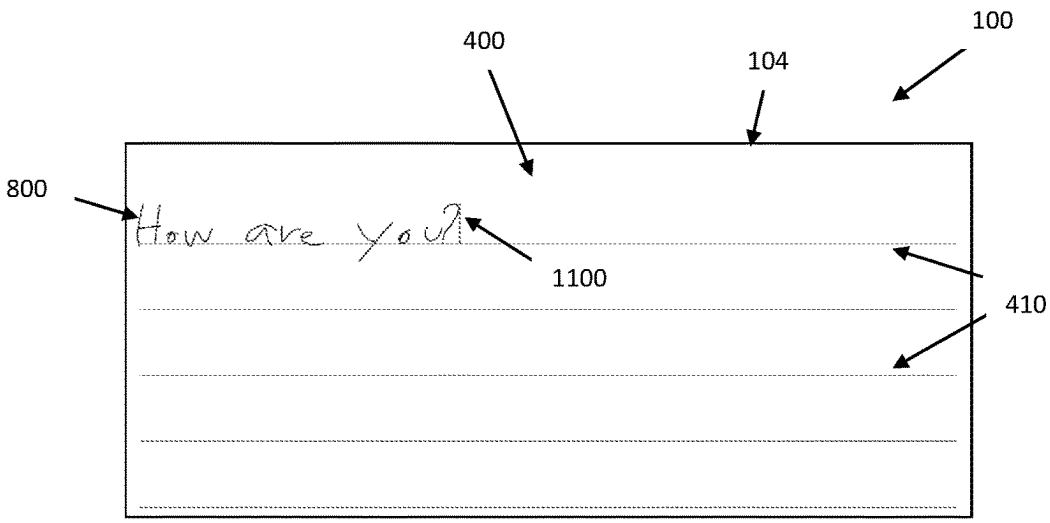

400
100
104
800
How are you?
1100
410

FIG. 11

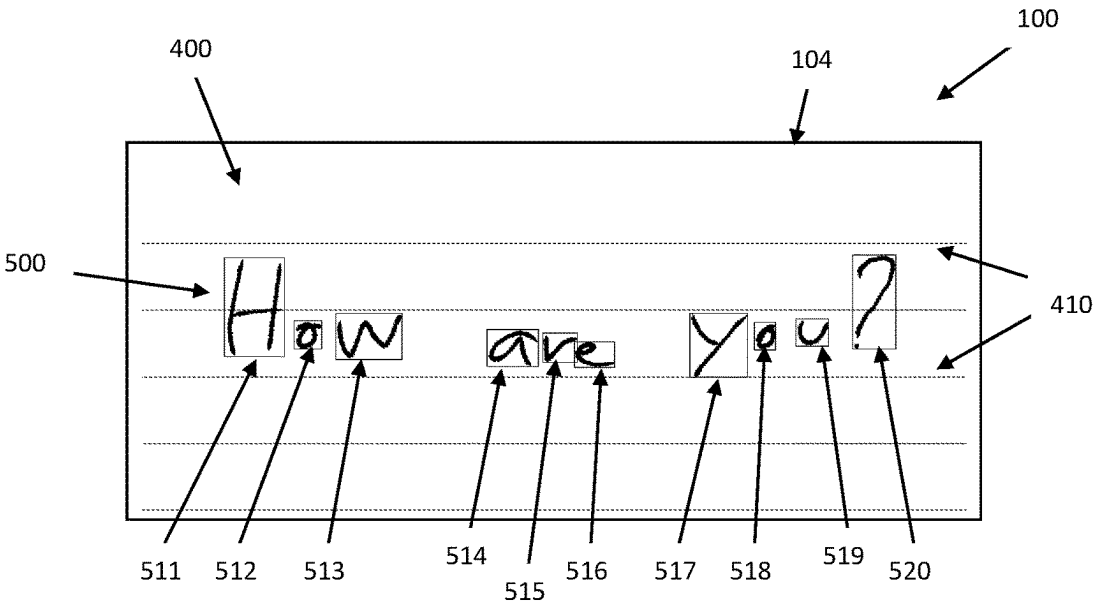

100
400
104
500
How are you ?
410
511
512
513
514
515
516
517
518
519
520

FIG. 12

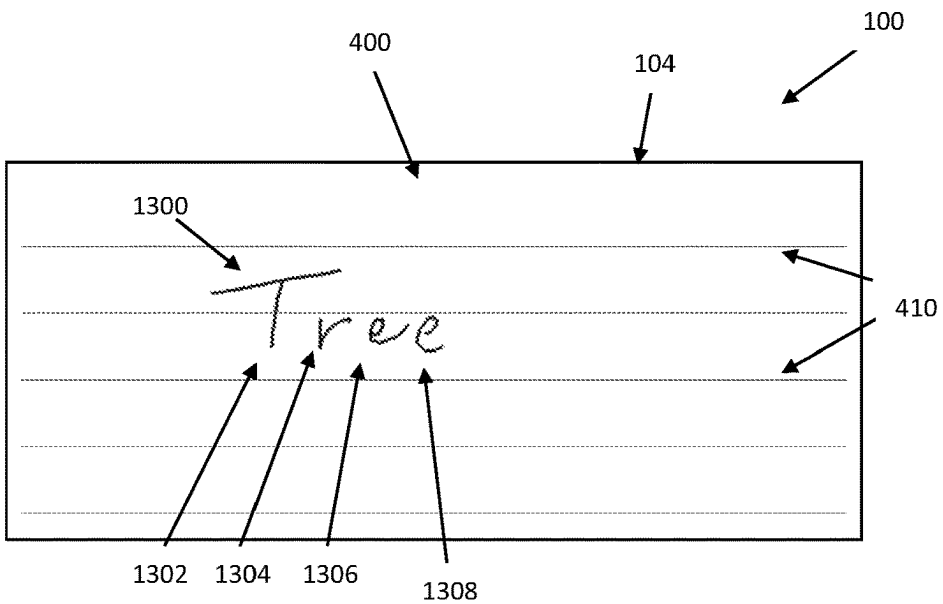

100
400
104
1300
Tree
410
1302
1304
1306
1308

FIG. 13

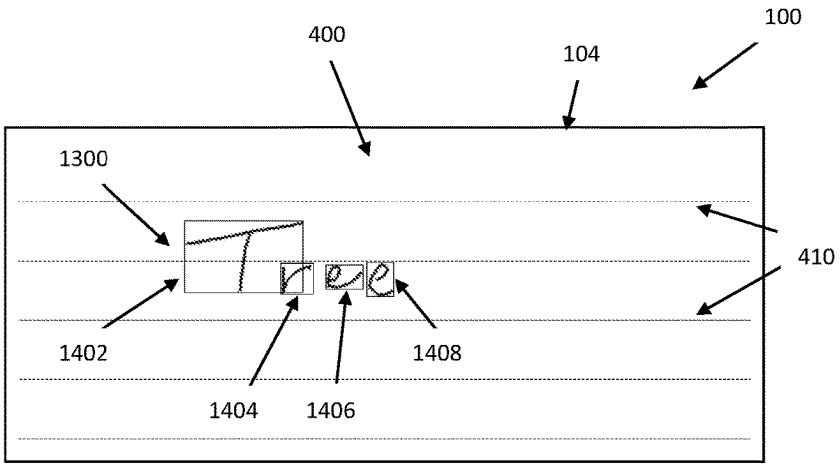

400
104
100
1300
410
1402
1404
1406
1408

FIG. 14

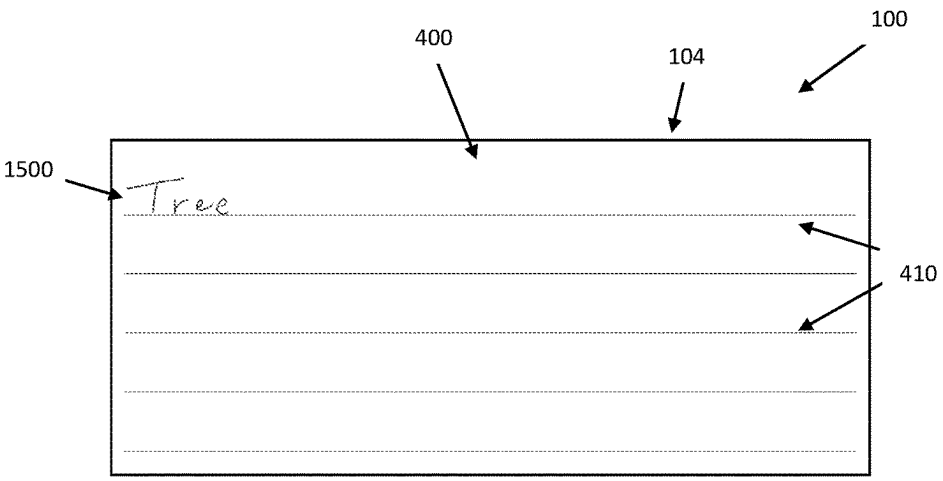

400
104
100
1500
Tree
410

FIG. 15

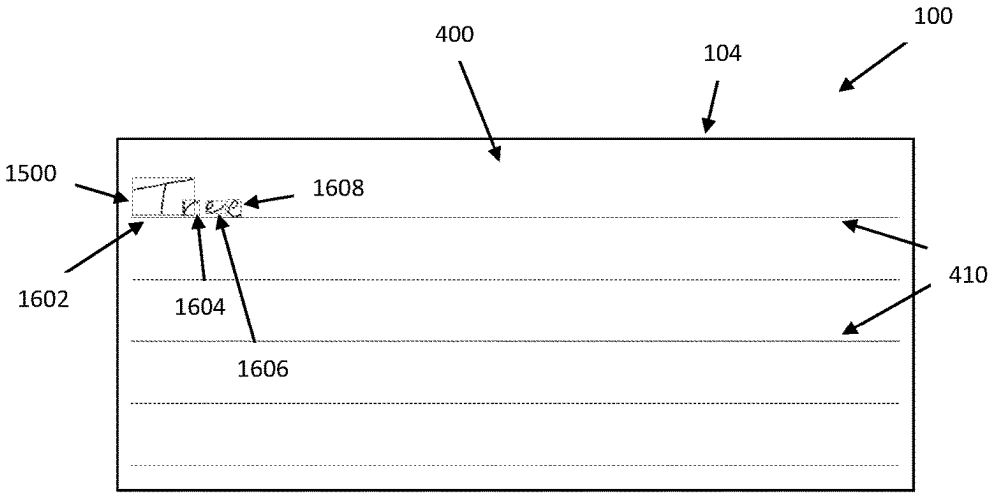


FIG. 16

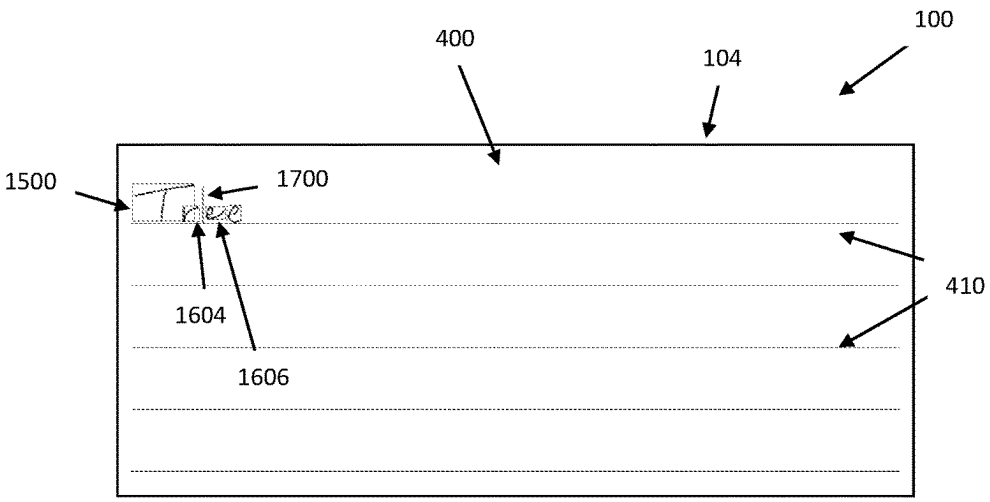


FIG. 17A

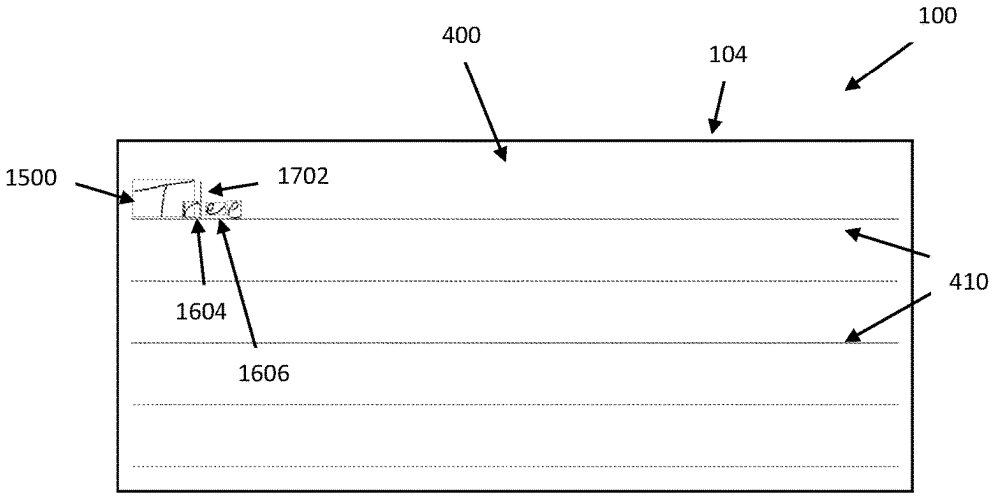


FIG. 17B

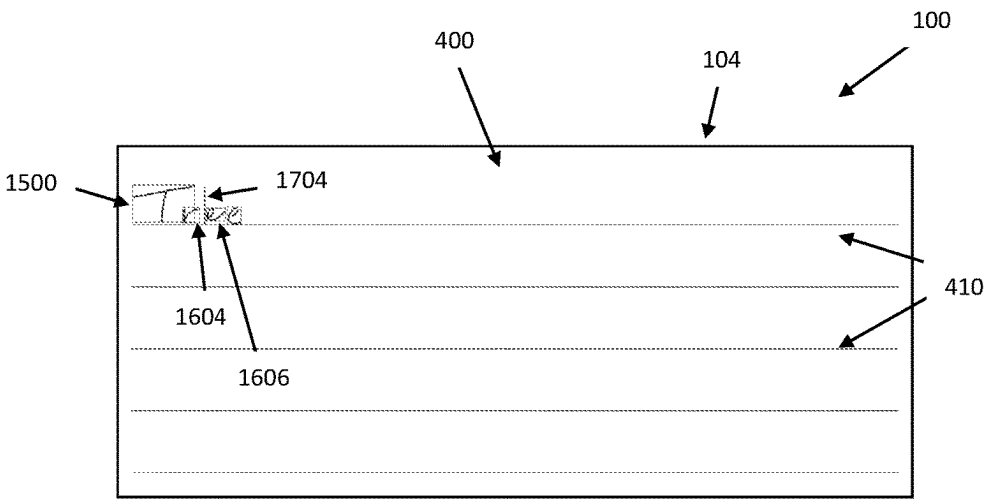


FIG. 17C

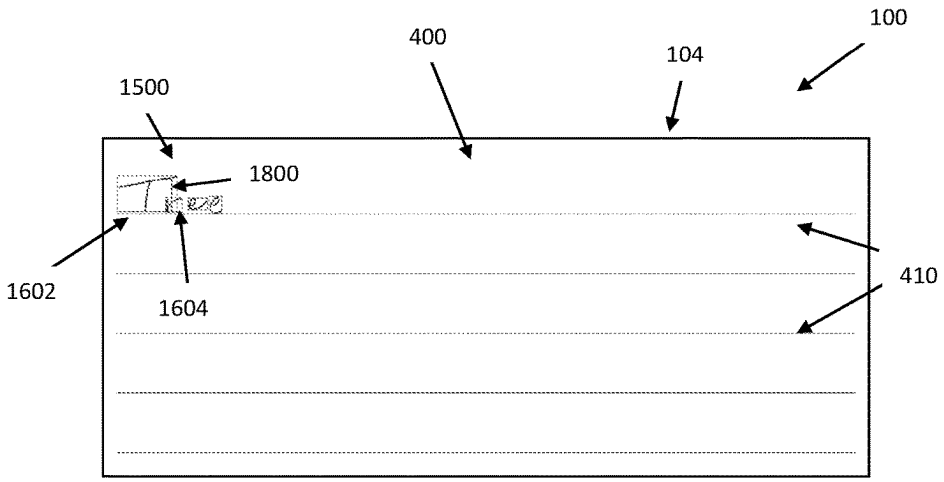


FIG. 18A

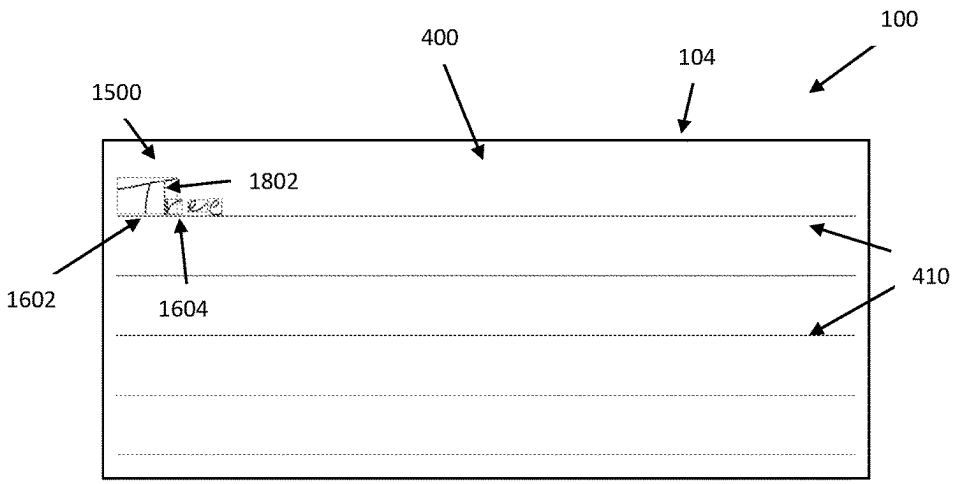


FIG. 18B

SYSTEM AND METHOD FOR BEAUTIFYING DIGITAL INK

CROSS REFERENCE TO RELATED APPLICATIONS

This application claims priority to European Application No. 15 290 303.5 filed on Dec. 8, 2015, the entire contents of which is incorporated herein.

TECHNICAL FIELD

The present description relates generally to the field of handwriting recognition systems and methods using computing device interfaces. The present description relates more specifically to systems and methods for beautifying digital ink rendered from handwriting input to such computing device interfaces and providing editing of the digital ink on the computing device interfaces.

BACKGROUND

Computing devices continue to become more ubiquitous to daily life. They take the form of computer desktops, laptop computers, tablet computers, hybrid computers (2-in-1s), e-book readers, mobile phones, smartphones, wearable computers (including smartwatches, smart glasses/headsets), global positioning system (GPS) units, enterprise digital assistants (EDAs), personal digital assistants (PDAs), game consoles, and the like. Further, computing devices are being incorporated into vehicles and equipment, such as cars, trucks, farm equipment, manufacturing equipment, building environment control (e.g., lighting, HVAC), and home and commercial appliances.

Computing devices generally consist of at least one processing element, such as a central processing unit (CPU), some form of memory, and input and output devices. The variety of computing devices and their subsequent uses necessitate a variety of interfaces and input devices. One such input device is a touch sensitive surface such as a touch screen or touch pad wherein user input is received through contact between the user's finger or an instrument such as a pen or stylus and the touch sensitive surface. Another input device is an input surface that senses gestures made by a user above the input surface. A further input device is a position detection system which detects the relative position of either touch or non-touch interactions with a non-touch physical or virtual surface. Any of these methods of input can be used generally for drawing or inputting text. When user input is text the user's handwriting is interpreted using a handwriting recognition system or method.

One application of handwriting recognition in non-portable and portable computing devices, such as smartphones, phablets and tablets, is for the input of text into various applications run by the computing devices in a manner similar to that traditionally done with keyboard, either physical or virtual. However, unlike keyboard input which is governed by strict layout rules, such as text entry on visible or invisible lines in accordance with the position of a displayed cursor, handwriting can be input virtually anywhere on the input surface of the device. While this substantially unconstrained input has some effect on the recognition accuracy of the handwriting, currently available advanced recognition algorithms are generally able to deal with such relatively 'free' positioning of textual characters.

Of more effect is a user's ability to interact with the input once it has been rendered as so-called 'digital ink' on the display. This is because, unlike typed or 'typeset ink' text which has characters relatively located in a uniform and known manner, the characters of the digital ink text are not uniform or in relatively known positions, generally due to the irregularity inherent in handwriting. Accordingly, conventional systems and methods typically only allow further interaction to be made on the typeset ink rendered from the recognized text, which is generally done at a different position of the display from the area where input is allowed.

It is desired by users however to be able to interact directly with the handwriting as well, such as to edit the input (e.g., add or delete characters or words), so that the creative flow of handwriting input is maintained. Some available applications provide alignment and scaling of the digital ink through normalization and the like. However, these do not take into account certain geometrical features of the handwriting and therefore the scaling and aligning is not properly performed to allow full interaction or faithful reproduction of the original digital ink. Further, they do not use recognition to provide these operations, and as such do not provide interaction capabilities within the input signal. Accordingly, editing operations can only be performed at the word level, and not at the character level, for example.

SUMMARY

The examples of the present invention that are described herein below provide systems and methods for beautifying and providing editing of digital ink on computing devices.

In one example, the present invention includes a system for beautifying digital ink on computing devices. Each computing device may include a processor and at least one non-transitory computer readable medium for recognizing handwriting input under control of the processor. The non-transitory computer readable medium may be configured to cause display of, on a display interface of a computing device, first digital ink in accordance with the handwriting input, determine, from first geometrical information of the handwriting input determined through the recognition of the handwriting input and second geometrical information of an alignment structure of the display interface, a structuring transformation of the first digital ink, and cause display of, on the display interface, second digital ink in accordance with the structuring transformation.

The structuring transformation may be determined from at least two comparisons of the first and second geometrical information. The at least two comparisons may provide first and second ratio values of the first and second geometrical information and the structuring transformation is determined from the minimum of the first and second ratio values.

The structuring transformation may scale the first digital ink to produce the second digital ink. The structuring transformation may align the first digital ink with the alignment structure to produce the second digital ink. The alignment structure may be a line pattern.

In another example, the present invention includes a method for beautifying digital ink on computing devices. Each computing device may include a processor and at least one non-transitory computer readable medium for recognizing handwriting input under control of the processor. The method may include the steps of displaying, on a display interface of a computing device, first digital ink in accordance with the handwriting input, determining, from geometrical information of the handwriting input determined through the recognition of the handwriting input, a structuring transformation of the first digital ink, and displaying,

on the display interface, second digital ink in accordance with the structuring transformation.

The method may determine the structuring transformation from at least two comparisons of the first and second geometrical information. The at least two comparisons may provide first and second ratio values of the first and second geometrical information and the method may determine the structuring transformation from the minimum of the first and second ratio values.

The structuring transformation may scale the first digital ink to produce the second digital ink. The structuring transformation may align the first digital ink with the alignment structure to produce the second digital ink. The alignment structure may be a line pattern.

In another example, the present invention includes a non-transitory computer readable medium having a computer readable program code embodied therein. The computer readable program code may be adapted to be executed to implement a method for beautifying digital ink on a computing device. The computing device may include a processor and at least one system non-transitory computer readable medium for recognizing handwriting input under control of the processor. The method may include the steps of displaying, on a display interface of a computing device, first digital ink in accordance with the handwriting input, determining, from geometrical information of the handwriting input determined through the recognition of the handwriting input, a structuring transformation of the first digital ink, and displaying, on the display interface, second digital ink in accordance with the structuring transformation.

The method may determine the structuring transformation from at least two comparisons of the first and second geometrical information. The at least two comparisons may provide first and second ratio values of the first and second geometrical information and the method may determine the structuring transformation from the minimum of the first and second ratio values.

The structuring transformation may scale the first digital ink to produce the second digital ink. The structuring transformation may align the first digital ink with the alignment structure to produce the second digital ink. The alignment structure may be a line pattern.

In another example, the present invention includes a system for providing editing of digital ink on computing devices. Each computing device may include a processor and at least one non-transitory computer readable medium for recognizing handwriting input under control of the processor. The non-transitory computer readable medium may be configured to cause display of, on a display interface of a computing device, digital ink in accordance with the handwriting input, determine, from segmentation information of the handwriting input determined through the recognition of the handwriting input, ranges of elements of the digital ink, and cause display of, on the display interface, an input cursor in relation to the determined digital ink element ranges.

The elements of the digital ink may include text characters and the at least one non-transitory computer readable medium may be configured to display the input cursor at a position between first and second characters in relation to the determined ranges of the first and second characters.

The determined ranges of the first and second characters may overlap, and the at least one non-transitory computer readable medium may be configured to compare geometrical features of the determined ranges of the first and second characters, and cause the display of the input cursor on a border of the determined range of the first character or the determined range of the second character within the region of overlap in accordance with the comparison. The geometrical features may include height and width of the determined ranges.

In another example, the present invention includes a method for providing editing of digital ink on computing devices. Each computing device may include a processor and at least one non-transitory computer readable medium for recognizing handwriting input under control of the processor. The method may include the steps of displaying, on a display interface of a computing device, digital ink in accordance with the handwriting input, determining, from segmentation information of the handwriting input determined through the recognition of the handwriting input, ranges of elements of the digital ink, and displaying, on the display interface, an editing cursor in relation to the determined digital ink element ranges.

The elements of the digital ink may include text characters and the method may display the input cursor at a position between first and second characters in relation to the determined ranges of the first and second characters.

The determined ranges of the first and second characters may overlap, and the method may compare geometrical features of the determined ranges of the first and second characters, and display the input cursor on a border of the determined range of the first character or the determined range of the second character within the region of overlap in accordance with the comparison. The geometrical features may include height and width of the determined ranges.

In another example, the present invention includes a non-transitory computer readable medium having a computer readable program code embodied therein. The computer readable program code may be adapted to be executed to implement a method for providing editing of digital ink on a computing device. The computing device may include a processor and at least one system non-transitory computer readable medium for recognizing handwriting input under control of the processor. The method may include the steps of displaying, on a display interface of a computing device, digital ink in accordance with the handwriting input, determining, from segmentation information of the handwriting input determined through the recognition of the handwriting input, ranges of elements of the digital ink, and displaying, on the display interface, an editing cursor in relation to the determined digital ink element ranges.

The elements of the digital ink may include text characters and the method may display the input cursor at a position between first and second characters in relation to the determined ranges of the first and second characters.

The determined ranges of the first and second characters may overlap, and the method may compare geometrical features of the determined ranges of the first and second characters, and display the input cursor on a border of the determined range of the first character or the determined range of the second character within the region of overlap in accordance with the comparison. The geometrical features may include height and width of the determined ranges.

BRIEF DESCRIPTION OF THE DRAWINGS

The present system and method will be more fully understood from the following detailed description of the examples thereof, taken together with the drawings. In the drawings like reference numerals depict like elements. In the drawings:

FIG. **1** shows a block diagram of a computing device in accordance with an example of the present system and method;

FIG. **2** shows a block diagram of a system for handwriting recognition in accordance with an example of the present system and method;

FIG. **3** shows a block diagram illustrating detail of the handwriting recognition system of FIG. **2** in accordance with an example of the present system and method;

FIG. **4** shows a schematic view of an example visual rendering of an ink input area on a portion of an input surface of the computing device;

FIG. **5** shows the example input area of FIG. **4** having example handwritten input rendered as digital ink;

FIG. **6** shows the example digital ink of FIG. **5** with geometrical information provided by the handwriting recognition system;

FIG. **7** shows the example input area of FIG. **4** with geometrical information provided by the handwriting recognition system;

FIG. **8** shows example beautified digital ink rendered on a line pattern;

FIG. **9** shows the example input area of FIG. **4** having example handwritten input rendered as digital ink;

FIG. **10** shows the example digital ink of FIG. **9** with geometrical information provided by the handwriting recognition system;

FIG. **11** shows the example beautified digital ink of FIG. **8** with a cursor;

FIG. **12** shows the example digital ink of FIG. **5** with character bounding boxes determined from character segmentation of the example handwritten input provided by the handwriting recognition system;

FIG. **13** shows the example input area of FIG. **4** having example handwritten input rendered as digital ink;

FIG. **14** shows the example digital ink of FIG. **13** with character bounding boxes determined from character segmentation of the example handwritten input provided by the handwriting recognition system;

FIG. **15** shows example beautified digital ink rendered on the line pattern;

FIG. **16** shows the example beautified digital ink of FIG. **15** with character bounding boxes determined from character segmentation of the example handwritten input provided by the handwriting recognition system;

FIG. **17**A shows a first example cursor position for non-overlapping characters of the example beautified digital ink of FIG. **15**;

FIG. **17**B shows a second example cursor position for non-overlapping characters of the example beautified digital ink of FIG. **15**;

FIG. **17**C shows a third example cursor position for non-overlapping characters of the example beautified digital ink of FIG. **15**;

FIG. **18**A shows a fourth example cursor position for overlapping characters of the example beautified digital ink of FIG. **15**; and

FIG. **18**B shows a fifth example cursor position for overlapping characters of the example beautified digital ink of FIG. **15**.

DETAILED DESCRIPTION

In the following detailed description, numerous specific details are set forth by way of examples in order to provide a thorough understanding of the relevant teachings. However, it should be apparent to those skilled in the art that the present teachings may be practiced without such details. In other instances, well known methods, procedures, components, and/or circuitry have been described at a relatively high-level, without detail, in order to avoid unnecessarily obscuring aspects of the present teachings.

Reference to and discussion of directional features such as up, down, above, below, lowest, highest, horizontal, vertical, etc., are made with respect to the Cartesian coordinate system as applied to the input surface on which the input to be recognized is made. Further, terms such as left and right are made in relation to the reader's frame of reference when viewing the drawings. Furthermore, the use of the term 'text' in the present description is understood as encompassing all alphanumeric characters, and strings thereof, in any written language and common place non-alphanumeric characters, e.g., symbols, used in written text. Further still, the examples shown in these drawings are in a left-to-right written language context, and therefore any reference to positions can be adapted for written languages having different directional formats.

The various technologies described herein generally relate to capture, processing and management of hand-drawn and handwritten content on portable and non-portable computing devices in a manner which retains the inputted style of the content while allowing conversion to a faithful typeset or beautified version of that content. The systems and methods described herein may utilize recognition of users' natural writing and drawing styles input to a computing device via an input surface, such as a touch sensitive screen, connected to, or of, the computing device or via an input device, such as a digital pen or mouse, connected to the computing device or via a physical or virtual surface monitored by a position detection system. Whilst the various examples are described with respect to recognition of handwriting input using so-called online recognition techniques, it is understood that application is possible to other forms of input for recognition, such as offline recognition in which images rather than digital ink are recognized. The terms hand-drawing and handwriting are used interchangeably herein to define the creation of digital content by users through use of their hands either directly onto a digital or digitally connected medium or via an input tool, such as a hand-held stylus. The term "hand" is used herein to provide concise description of the input techniques, however the use of other parts of a users' body for similar input is included in this definition, such as foot, mouth and eye.

FIG. **1** shows a block diagram of an example computing device **100**. The computing device may be a computer desktop, laptop computer, tablet computer, e-book reader, mobile phone, smartphone, wearable computer, digital watch, interactive whiteboard, global positioning system (GPS) unit, enterprise digital assistant (EDA), personal digital assistant (PDA), game console, or the like. The computing device **100** includes components of at least one processing element, some form of memory and input and/or output (I/O) devices. The components communicate with each other through inputs and outputs, such as connectors, lines, buses, cables, buffers, electromagnetic links, networks, modems, transducers, IR ports, antennas, or others known to those of ordinary skill in the art.

The computing device **100** has at least one display **102** for outputting data from the computing device such as images, text, and video. The display **102** may use LCD, plasma, LED, iOLED, CRT, or any other appropriate technology that is or is not touch sensitive as known to those of ordinary skill in the art. At least some of the display **102** may be co-located with at least one input surface **104**. The input surface **104**

may employ technology such as resistive, surface acoustic wave, capacitive, infrared grid, infrared acrylic projection, optical imaging, dispersive signal technology, acoustic pulse recognition, or any other appropriate technology as known to those of ordinary skill in the art to receive user input. The input surface **104** may be bounded by a permanent or video-generated border that clearly identifies its boundaries.

In addition to the input surface **104**, the computing device **100** may include one or more additional I/O devices (or peripherals) that are communicatively coupled via a local interface. The additional I/O devices may include input devices such as a keyboard, mouse, scanner, microphone, touchpads, bar code readers, laser readers, radio-frequency device readers, or any other appropriate technology known to those of ordinary skill in the art. Further, the I/O devices may include output devices such as a printer, bar code printers, or any other appropriate technology known to those of ordinary skill in the art. Furthermore, the I/O devices may include communications devices that communicate both inputs and outputs such as a modulator/demodulator (modem; for accessing another device, system, or network), a radio frequency (RF) or other transceiver, a telephonic interface, a bridge, a router, or any other appropriate technology known to those of ordinary skill in the art. The local interface may have additional elements to enable communications, such as controllers, buffers (caches), drivers, repeaters, and receivers, which are omitted for simplicity but known to those of skill in the art. Further, the local interface may include address, control, and/or data connections to enable appropriate communications among the other computer components.

The computing device **100** also includes a processor **106**, which is a hardware device for executing software, particularly software stored in the memory **108**. The processor can be any custom made or commercially available general purpose processor, a central processing unit (CPU), a semiconductor based microprocessor (in the form of a microchip or chipset), a macroprocessor, microcontroller, digital signal processor (DSP), application specific integrated circuit (ASIC), field programmable gate array (FPGA) or other programmable logic device, discrete gate or transistor logic, discrete hardware components, state machine, or any combination thereof designed for executing software instructions known to those of ordinary skill in the art. Examples of suitable commercially available microprocessors are as follows: a PA-RISC series microprocessor from Hewlett-Packard Company, an 80x86 or Pentium series microprocessor from Intel Corporation, a PowerPC microprocessor from IBM, a Sparc microprocessor from Sun Microsystems, Inc., a 68xxx series microprocessor from Motorola Corporation, DSP microprocessors, or ARM microprocessors.

The memory **108** can include any one or a combination of volatile memory elements (e.g., random access memory (RAM, such as DRAM, SRAM, or SDRAM)) and nonvolatile memory elements (e.g., ROM, EPROM, flash PROM, EEPROM, hard drive, magnetic or optical tape, memory registers, CD-ROM, WORM, DVD, redundant array of inexpensive disks (RAID), another direct access storage device (DASD)). Moreover, the memory **108** may incorporate electronic, magnetic, optical, and/or other types of storage media. The memory **108** can have a distributed architecture where various components are situated remote from one another but can also be accessed by the processor **106**. Further, the memory **108** may be remote from the device, such as at a server or cloud-based system, which is remotely accessible by the computing device **100**. The memory **108** is coupled to the processor **106**, so the processor **106** can read information from and write information to the memory **108**. In the alternative, the memory **108** may be integral to the processor **106**. In another example, the processor **106** and the memory **108** may both reside in a single ASIC or other integrated circuit.

The software in the memory **108** includes an operating system **110** and an application **112**. The software optionally further includes a handwriting recognition (HWR) system **114** which may each include one or more separate computer programs. Each of these has an ordered listing of executable instructions for implementing logical functions. The operating system **110** controls the execution of the application **112** (and the HWR system **114**). The operating system **110** may be any proprietary operating system or a commercially available operating system, such as WEBOS, WINDOWS®, MAC and IPHONE OS®, LINUX, and ANDROID. It is understood that other operating systems may also be utilized.

The application **112** includes one or more processing elements related to detection, management and treatment of user input (discussed in detail later). The software may also include one or more other applications related to handwriting recognition, different functions, or both. Some examples of other applications include a text editor, telephone dialer, contacts directory, instant messaging facility, computer-aided design (CAD) program, email program, word processing program, web browser, and camera. The application **112**, and the other applications, include program(s) provided with the computing device **100** upon manufacture and may further include programs uploaded or downloaded into the computing device **100** after manufacture.

The HWR system **114**, with support and compliance capabilities, may be a source program, executable program (object code), script, application, or any other entity having a set of instructions to be performed. When a source program, the program needs to be translated via a compiler, assembler, interpreter, or the like, which may or may not be included within the memory, so as to operate properly in connection with the operating system. Furthermore, the handwriting recognition system with support and compliance capabilities can be written as (a) an object oriented programming language, which has classes of data and methods; (b) a procedure programming language, which has routines, subroutines, and/or functions, for example but not limited to C, C++, Pascal, Basic, Fortran, Cobol, Perl, Java, Objective C, Swift, and Ada; or (c) functional programing languages for example but no limited to Hope, Rex, Common Lisp, Scheme, Clojure, Racket, Erlang, OCaml, Haskell, Prolog, and F#. Alternatively, the HWR system **114** may be a method or system for communication with a handwriting recognition system remote from the device, such as server or cloud-based system, but is remotely accessible by the computing device **100** through communications links using the afore-mentioned communications I/O devices of the computing device **100**. Further, the application **112** and the HWR system **114** may operate together or be combined as a single application.

Strokes entered on or via the input surface **104** are processed by the processor **106** as digital ink. A user may enter a stroke with a finger or some instrument such as a pen or stylus suitable for use with the input surface. The user may also enter a stroke by making a gesture above the input surface **104** if technology that senses motions in the vicinity of the input surface **104** is being used, or with a peripheral device of the computing device **100**, such as a mouse or joystick. A stroke is characterized by at least the stroke initiation location, the stroke termination location, and the

path connecting the stroke initiation and termination locations. Because different users may naturally write the same object, e.g., a letter, a shape, a symbol, with slight variations, the HWR system accommodates a variety of ways in which each object may be entered whilst being recognized as the correct or intended object.

FIG. **2** is a schematic pictorial of an example of the HWR system **114**, in either its local (i.e., loaded on the device **100**) or remote (i.e., remotely accessible by the device **100**) forms. The HWR system **114** includes stages such as preprocessing **116**, recognition **118** and output **120**. The preprocessing stage **116** processes the digital ink to achieve greater accuracy and reducing processing time during the recognition stage **118**. This preprocessing may include normalizing of the path connecting the stroke initiation and termination locations by applying size normalization and/or methods such as B-spline approximation to smooth the input. The preprocessed strokes are then passed to the recognition stage **118** which processes the strokes to recognize the objects formed thereby. The recognized objects are then output **120** to the display **102** generally as a typesetted version of the handwritten elements/characters.

The recognition stage **118** may include different processing elements or experts. FIG. **3** is a schematic pictorial of the example of FIG. **2** showing schematic detail of the recognition stage **118**. Three experts, a segmentation expert **122**, a recognition expert **124**, and a language expert **126**, are illustrated which collaborate through dynamic programming to generate the output **120**.

The segmentation expert **122** defines the different ways to segment the input strokes into individual element hypotheses, e.g., alphanumeric characters and mathematical operators, text characters, individual shapes, or sub expression, in order to form expressions, e.g., words, mathematical equations, or groups of shapes. For example, the segmentation expert **122** may form the element hypotheses by grouping consecutive strokes of the original input to obtain a segmentation graph where each node corresponds to at least one element hypothesis and where adjacency constraints between elements are handled by the node connections. Alternatively, the segmentation expert **122** may employ separate experts for different input types, such as text, drawings, equations, and music notation.

The recognition expert **124** provides classification of the features extracted by a classifier **128** and outputs a list of element candidates with probabilities or recognition scores for each node of the segmentation graph. Many types of classifiers exist that could be used to address this recognition task, e.g., Support Vector Machines, Hidden Markov Models, or Neural Networks such as Multilayer Perceptrons, Deep, Convolutional or Recurrent Neural Networks. The choice depends on the complexity, accuracy, and speed desired for the task.

The language expert **126** generates linguistic meaning for the different paths in the segmentation graph using language models (e.g., grammar or semantics). The expert **126** checks the candidates suggested by the other experts according to linguistic information **130**. The linguistic information **130** can include a lexicon, regular expressions, etc. The language expert **126** aims at finding the best recognition path. In one example, the language expert **126** does this by exploring a language model such as finite state automaton (FSA) representing the content of linguistic information **130**. In addition to the lexicon constraint, the language expert **126** may use statistical information modeling for how frequent a given sequence of elements appears in the specified language or is used by a specific user to evaluate the linguistic likelihood of the interpretation of a given path of the segmentation graph.

The present system and method make use of the HWR system **114** in order to recognize handwritten input to the device **100**. As discussed earlier, unlike keyboard input which is governed by strict layout rules, such as text entry on visible or invisible lines in accordance with the position of a displayed cursor, the present system and method allows handwriting to be input virtually anywhere on the input surface **104** of the computing device **100**, and this input is rendered as digital ink in the input position, or in an offset position if the handwriting input is offset from the rendering position. The HWR system **114** is able to recognize this relatively 'freely' positioned handwriting. The present system and method further allows users to continue to interact with the digital ink itself and provide meaningful guidance and results of that interaction through beautifying the digital ink. Beautification is achieved by properly scaling and aligning the digital ink in accordance with an alignment structure of the input surface **104** by taking into account certain geometrical features of the handwriting (described in detail later). Interaction is assisted by the performance of segmentation of strokes in the recognition process and using information on this segmentation to allow management of an input or editing cursor that acts as a pointer for character level interactions and editing operations. These and other aspects of the present system and method are now described.

FIG. **4** shows a schematic view of an example visual rendering of an ink input or capture area **400** on a portion of the input surface **104** of an example computing device **100**. The input area **400** is provided as a constraint-free canvas that allows users to create object blocks (blocks of text, drawings, etc.) anywhere without worrying about sizing or alignment. However, as can be seen an alignment structure in the form of a line pattern background **410** can be provided for guidance of user input and the alignment of digital and typeset ink objects. In any case, as users may input handwriting that is not closely aligned to the line pattern, or may desire to ignore the line pattern and write in a unconstrained manner, such as diagonally or haphazardly, the recognition of the handwriting input is performed by the HWR system **114** without regard to the line pattern. An example alignment pattern is described in United States Patent Application Publication No. 2017/0060819 titled "System and Method of Digital Note Taking" filed in the name of the present Applicant and Assignee, the entire content of which is incorporated by reference herein.

For example, from this incorporated by reference patent application, the line pattern **410** has horizontal lines separated by a multiple of the vertical rhythm height unit being the density independent pixel (dp). Regardless of the particular device **100**, the vertical distance between the horizontal lines **410** is defined by a line pattern unit (LPU) and the vertical rhythm height unit provides a graduated measure of the LPU on a particular device. For example, the LPU may be set at about one centimeter for any device being a certain multiple of the vertical rhythm height unit or users may be allowed to customize the line pattern unit to a different multiple of the vertical rhythm height unit according to their writing style. Alternatively, the vertical rhythm may be based on typeset text size (e.g., the minimum text size) and the LPU is provided as a multiple of this typeset text size. All lines **410** are displayed with the same light and subtle color, e.g., grey, that is visible but faded with respect to the rendering of the content itself. In this way the line

pattern is noticeable but unobtrusive so as to guide the handwriting input without distracting from the content entry.

In order to assist user input, the application **112** can render the digital ink according to the user input in the position at which the input is made. For example, FIG. **5** shows the example input area **400** having handwritten input rendered as (first) digital ink **500**. The digital ink **500** includes text characters or symbols, "H" **501**, "o" **502**, "w" **503**, "a" **504**, "r" **505**, "e" **506**, "y" **507**, "o" **508**, "u" **509** and "?" **510**, forming the question phrase "How are you?". As can be seen, the input represented by the digital ink text **500** has been input by a user so as not to be in accord with the line pattern **410**. That is, the characters **501** to **510** of the digital ink text **500** have not been written directly on any one of the lines **410**. This input can be rendered on the line pattern however by applying a structuring transformation on the strokes of the input to beautify the digital ink through proper scaling and alignment. In order to do this appropriately information regarding the relative geometry of the digital ink text **500** is determined, such as the descending part of the handwriting, e.g., the distance between base- and mid-lines of each handwritten character, as described in detail below.

The (first) geometrical information is provided by the HWR system **114**. That is, in the recognition process, the recognition engine **118** determines information on the lower and upper extrema of each character of the input to the application **112** as recognized by the HWR system **114**. The lower extremum is the lowest point at which the stroke (or strokes) of the character pass. The upper extremum is the highest point at which the stroke (or strokes) of the character pass. The recognition process also determines from these extremum writing lines at the levels of character, word (e.g., a series of characters), sentence or phrase (e.g., a series of characters and/or words), and text line (e.g., a series of characters, words and/or sentences). The writing lines include top-, bottom-, base- and mid-lines.

The top-line is defined as the horizontal line passing through a point related to the upper extremum across the horizontal extent of the character, word, sentence or text line. The bottom-line is defined as the horizontal line passing through a point related to the lower extremum across the horizontal extent of the character, word, sentence or text line. The base-line is defined as the horizontal line passing through a point related to the bottom-line across the horizontal extent of the character, word, sentence or text line, and the mid-line is defined as the horizontal line passing through a point related to the top- and base-lines across the horizontal extent of the character, word, sentence or text line. The determination of the points through which these lines pass is particular to the recognition process and language model(s) used by the HWR system **114**.

For example, for languages having all characters and words made from those characters in a sentence or phrase written with a similar vertical extent, such as pictographic languages, for characters within a word, sentence or text line, the character having the highest upper extrema may be used as the point for either (or both) the top- or mid-line, and the character having the lowest lower extrema may be used as the point for the either (or both) bottom- and base-line, for example.

On the other hand, for languages having characters or letters written in a sentence or phrase with different sizes, such as Latin-based languages, e.g., upper and lower casing, letters with ascenders and/or descenders (e.g., "h", "g"), diacritics (e.g., "i", "j") and punctuation symbols (e.g., "?"), for characters within a word, sentence or text line, the character having the highest upper extrema may be used as the point for the top-line, the character having the lowest lower extrema may be used as the point for the bottom-line, the character having the lowest lower extrema of characters which do not have a descender may be used as the point for the base-line, and the lower case character having the highest upper extrema of lower case characters which do not have an ascender may be used as the point for the mid-line, for example.

Alternatively, instead of setting the writing lines on absolute values, average or mean values can be taken, such as the average or mean of the lower and upper extrema across the text line, sentence or words, or the strokes of the characters themselves, so that local variations and anomalies, such as unduly offset characters, are not given undesired weight.

FIG. **6** shows the example input of FIG. **5** with the above-described geometrical information of the base- and mid-(or top-)lines provided by the HWR system **114** illustrated. That is, the characters **501-503** are recognized as belonging to one word **600**, e.g., "How", with base-line B**0** and mid-line M**0** determined for the word **600**, the characters **504-506** are recognized as belonging to one word **602**, e.g., "are", with base-line B**2** and mid-line M**2** determined for the word **602**, the characters **507-509** are recognized as belonging to one word **604**, e.g., "you", with base-line B**4** and mid-line M**4** determined for the word **604**, and the character **510** is recognized as being a single symbol **606**, with base-line B**6** determined for the symbol **606**. The present system and method uses the geometrical information to provide a structuring transformation through proper scaling and alignment of the strokes of the input as follows.

The structuring transformation uses a ratio which is the minimum between a number of different ratio values related to comparisons of (second) geometrical information of the line pattern **410** and the (first) geometrical information of the input at character, word, sentence or text line level. The level at which first, second and third ratios (described below) are all determined may be, for example, pre-determined or settable via the UI of the application **112**. In the examples discussed herein, the word level is depicted, but it is understood that similar description is applicable to the other levels.

The geometrical information of the line pattern **410** is determined by the application and includes top-, base- and mid-lines, as shown in FIG. **7**. For example, the base-line BL of each line **410** of the line pattern is defined as that line **410**, the top-line TL of each line **410** is defined as being at an (first) offset vertical distance from the base-line BL of that line **410** (e.g., between about 60% and about 80% of the LPU), and the mid-line ML of each line **410** is defined as being at an (second) offset vertical distance from the base-line BL (and/or top-line TL) of that line **410** (e.g., between about 30% and about 70% of the distance between the base-line BL and top-line TL). In this way, sufficient space for ascenders and descenders, upper casing, and grammatical symbols is provided once the digital ink has been scaled onto the line pattern **410**.

A first ratio value is the ratio of the (first) distance (i.e., height) of the line pattern, i.e., one LPU, and the (first) distance (i.e., height) of each word of the input. The height of a word is defined as the vertical extent between the bottom-line and top-line of that word. A similar definition is applicable to the character level, as these are like words having a single character, such as the symbol **606**. For sentences and text lines the height may be determined from the absolute values for the extrema, i.e., the lowest lower extremum and the highest upper extremum, and/or from the base- and top-lines, which may or may not be based on

average, mean or adjusted (e.g., upper case or ascender/descender characters discarded) values.

Thus, in the example of FIG. **6**, the height of the word **600** is from the highest point of the word, being the highest point of the character **501**, to the lowest point of the word, being the lowest point of the character **503**, the height of the word **602** is from the highest point of the word (e.g., the highest point of the character **504**) to the lowest point of the word, (e.g., the lowest point of the character **508**), the height of the word **604** is from the highest point of the word (e.g., the highest point of the character **507**) to the lowest point of the word (e.g., the lowest point of the character **507**), and the height of the symbol **606** is from the highest point of the character **510** to the lowest point of the character **510**, for example. The first ratio of each word therefore provides a comparison of the total height of that word, as set by the characters of the word (i.e., including casing, ascenders, descenders and grammatical symbols), and the LPU.

A second ratio value is the ratio of the (second) distance (e.g., top height) from the base-line BL to the top-line TL of the line pattern **410** and the (second) distance (e.g., top height) from the base-line to the top-line of each word of the input. Thus, in the example of FIG. **6**, the top height of the word **600** is from the top-line of the word (e.g., passing through the highest point of the character **501**) to the base-line B**0**, the top height of the word **602** is from the top-line of the word (e.g., passing through the highest point of the character **504**) to the base-line B**2**, the top height of the word **604** is from the top-line of the word (e.g., passing through the highest point of the character **507**) to the base-line B**4**, and the top height of the symbol **606** is from the top-line of the symbol **606** (e.g., passing through the highest point of the character **510**) to the base-line B**6**. The second ratio of each word therefore provides a comparison of the top-line to base-line height of that word and the top-line height of the lines **410**.

The third ratio is the ratio of the (third) distance (e.g., mid height) from the base-line BL to the mid-line ML of the line pattern **410** and the (third) distance (e.g., mid height) from the base-line to the mid-line of each word of the input. Thus, in the example of FIG. **6**, the mid height of the word **600** is from the mid-line M**0** to the base-line B**0**, the mid height of the word **602** is from the mid-line M**2** to the base-line B**2**, the mid height of the word **604** is from the mid-line M**4** to the base-line B**4**, and the mid height of the symbol **606** is from the mid-line of the symbol **606** (e.g., the same as the top-line) to the base-line B**6**. The third ratio of each word therefore provides a comparison of the mid-line to base-line height of that word and the mid-line height of the lines **410**.

Two or more of these ratios are compared in order to determine the minimum value. For example, the first and second ratios may be compared, or the first and third, or the second and third, or all three ratios. Further, other ratios may be determined and used in the comparison. The comparison allows a minimum character, word, sentence or text line size to be used for the structuring transformation. In the structuring transformation, the rendered strokes of the scaled digital ink are the smallest compared to the original input. However, if the minimum value of each compared ratio is greater than 1, then the returned minimum value is capped at 1, such that no scaling is performed. This especially applies to some punctuation marks which do not require scaling, such as periods.

Once the comparison is made, the elements of the (non-scaled or original) digital ink (e.g., the characters, words, etc. depending on the level used for the comparison) are scaled by the minimum value and are rendered on one of the lines **410** of the line pattern so that the base-line of the scaled element is at the base-line of that line **410** (i.e., on the line **410**). FIG. **8** shows an example in which the minimum ratio values for each of the words **600** to **606** of the digital ink **500** are determined from a comparison of the first, second and third ratios described above in order to perform the structuring transformation of the digital ink **500** to structured (second) digital ink **800**. The structured digital ink text **800** includes a word **801** being the scaled version of the word **600**, a word **802** being the scaled version of the word **602**, a word **804** being the scaled version of the word **604**, and a word **806** being the scaled version of the word **606**. The structured digital ink text **800** is rendered on the uppermost (first) line of the line pattern **410** so that the base-line of each scaled word **801** to **806** is positioned in the first line **410**.

The HWR system **114** may have some issues with determining the base-lines of elements of the input since there may be many ambiguities leading to improbable base-line recognition. Accordingly, the application **112** may be configured to handle the translation of the scaled strokes depending on the elements/characters that are recognized by the HWR system **114**. For example, the HWR system **114** may not be able to set a base-line for a single horizontal line recognized as a hyphen or minus sign, accordingly the application **112** may display the scaled strokes with an offset from the line **410**, such as to be centered vertically in the line pattern of the rendering area.

The positioning of each element on the multiple lines of the line pattern **410** is made based on certain pre-defined or settable criteria, such as, for left-to-right written languages, aligned to the first available left-most section of a first available line from the top of the rendering area of the device **100** which can accommodate the scaled element. For example, in the example of FIG. **8**, the rendering area is overlapped or coextensive with the input area **400** such that the scaled digital ink text **800** is left-aligned on the first line **410**, since there is no other content (e.g., digital or typeset ink) displayed on any of the lines **410**. Another example of criteria for positioning of the transformed ink can be the usage of a cursor as an insertion point of the new input, e.g., allowing users to complete previous inputs (described in detail later). Such positioning may therefore include reflowing of the input over multiple ones of the lines **410**, for example, if the next available position is on a line which is below the line previously used for the structured digital ink. In this way, paragraphs of text can be formed from a series of inputs of single lines of text (input as described below) and editing operations can be performed on the structured digital ink including those that trigger reflowing events (as described later).

With respect to the input and rendering areas or zones of the application **112**, it is noted that these areas may be coextensive, as in the example of FIG. **8**, such that once input has been received and displayed/rendered as the original digital ink (e.g., the digital ink **500**) after a set amount of time, as measured by a clock of the computing device **100** in conjunction with the memory **108** for example, the input area is cleared of the original digital ink and the structured digital ink (e.g., the digital ink **800**) is displayed/rendered on the rendering area. Alternatively, the input and rendering areas may be separated into different zones of the interface **104** or on separate displays or display devices, such that display of the original digital ink may persist and be scrolled, for example, to allow further input, or be omitted after a certain time period or receipt of new input. For example, superimposed input in which characters or words or parts of words are input over the top of one another with

fading or other omission of previously entered input may be possible in either case of separated or overlapped display, such as described in United States Patent Publication No. 2015/0286886 filed in the name of the present Applicant and Assignee and United States Patent Publication No. 2015/0356360 filed in the name of the present Applicant and Assignee, the entire contents of which are incorporated by reference herein.

Further, in the structuring transformation, the spaces between characters and between words in the input are transformed to a fixed spacing, such as shown in FIG. **8**. The fixed spacing may be set based on the LPU (e.g., about 5% to about 15% of the LPU) or the mid-line height of the line pattern (e.g., about 40% to about 60% of the mid height), or on other criteria. The fixed spacing is provided so that generally normalized and beautified digital ink output is provided aligned to the line pattern, particularly with respect to the above-mentioned new input which may be rendered as scaled digital ink on the same line as previously scaled input. Alternatively, the spacing between the characters/words of the input may be scaled in relation to the scaling of the associated characters/words.

The above-discussed example includes input which is generally horizontally provided in the left-to-right (or right-to-left) written context. However, as is understood as the geometry of elements of the input text, such as words, are considered for the structuring transformation, other manners of input are also handled by the present system and method. For example, FIG. **9** shows the example input area **400** having handwritten input rendered as digital ink text **900**. The digital ink text **900** includes the 'words' **600**, **602**, **604** and **606**, as in the above-described example, however these words are vertically input so that the word **600** is above the other words, the word **602** is above the words **604** and **606**, and the words **604** and **606** are on the same text line. As such the digital ink **900** is on multiple text lines which are not aligned with any of the lines **410**. As with the afore-described examples, the base-line B**0** and mid-line M**0** are determined for the word **600**, the base-line B**2** and mid-line M**2** are determined for the word **602**, the base-line B**4** and mid-line M**4** are determined for the word **604**, and the base-line B**6** is determined for the symbol **606**, as shown in FIG. **10**. Thus, as the geometrical information is the same, the structuring transformation is the same as described earlier, such that the non-structured digital ink **900** is transformed into the scaled and aligned digital ink **800** shown in FIG. **8**. As can be understood, similar transformation can be applied to other alignments and manners of inputting the handwriting.

The beautification of the non-beautified or 'raw' digital ink according to the handwritten input to the device **100** into the beautified digital ink provides a way of displaying user handwritten input in a normalized and standard fashion thereby aiding legibility, usability and interactability with the content. It is considered that guidance for users for such interactions with the beautified digital ink using mechanisms similar to those expected in document processing applications is beneficial, as this enhances the user experience with the application **112**. One such mechanism is the use of an input or editing cursor that acts as a pointer for character level editing operations, such as carriage return, space insertion, character deletion or insertion. For example, FIG. **11** shows a cursor **1100** provided after the beautified digital ink **800**.

It is understood that beyond guiding handwritten interaction with the beautified content, the cursor also provides positioning of other forms of input, such as typing via keyboard or text input by voice recognition. As can be seen, the cursor **1100** is displayed with sizing and alignment similar to the structured digital ink. The cursor may be displayed by the application **112**, and editing functions performed, in response to user interaction with the digital ink within the rendering or editing area. This interaction may be through known gestures on touch and hover devices (such as tap, double tap, press, long press, swipe, pinch), specific digital ink editing gestures (such as those described in European Patent Application No. 15 290 296.1 and U.S. patent application Ser. No. 14/989,217 both titled "System and Method for Note Taking with Gestures" and both filed in the name of the present Applicant and Assignee, the entire contents of which are incorporated by reference herein), or hardware tools (such as keyboard, mouse). Alternatively, or additionally, the cursor may be displayed by the application **112** in a (set) default position, such as at the end of the (most recent) beautified digital ink (e.g., as shown in FIG. **11**), for example, once the digital ink has been beautified. The cursor remains displayed, for example, until new handwriting input is detected by the application **112**.

As mentioned earlier, the present system and method provides the cursor management through segmentation in the recognition process. This aspect of the present system and method is now described. As discussed above, segmentation is performed by the HWR system **114** during the recognition process. This segmentation information is retrieved by the application **112** to determine the ranges of each recognized element within the handwritten input, such as for example, characters of handwritten text, shapes of handwritten drawings, etc. For example, FIG. **12** shows the determined ranges, illustrated as character bounding boxes **511** to **520** for an example segmentation of the characters **501** to **510** of the input, respectively. The present system and method uses these ranges to display the cursor at the bounding box of each character. For example, in FIG. **11** the cursor **1100** is displayed at the right edge of the bounding box **520** of the character **510**.

However, unlike typeset or fontified ink, digital ink, even the beautified digital ink, is not regularly provided on a character-by-character basis. This is because, handwriting is a form of personal font of the writing user, and while basic standards are followed in handwritten languages, many variations will exist. As such, many of the characters will overlap horizontally, or vertically, or both, due to capitalization, ascenders, descenders, slanted writing, or just untidy handwriting. While character overlap is handled by the HWR system **114** to properly recognize the characters, the character positions themselves are not adjusted through the segmentation or the beautification described above. Such adjustment could be made, but it is difficult to provide general rules because as stated above, the manner of handwriting is very personal. Accordingly, character overlap causes overlap of the bounding boxes. For example, as can be seen in FIG. **12** the bounding boxes **515** and **516** overlap dues to overlap of the characters **505** and **506**. As such, rendering a cursor on an overlapped bounding box may cause confusion for users. Examples of how the present system and method handle this situation is described below in relation to FIGS. **13** to **18**.

FIG. **13** shows the example input area **400** having handwritten input rendered as digital ink text **1300**. The digital ink text **1300** includes the characters or symbols, "T" **1302**, "r" **1304**, "e" **1306**, and "e" **1308** forming the capitalized word "Tree". As before, the input represented by the digital ink text **1300** has been input by a user so as not to be in accord with the line pattern **410**. FIG. **14** shows the digital

ink text **1300** with the determined ranges of each character as segmented by the HWR system **114** illustrated as bounding boxes **1402** to **1408** corresponding to the characters **1302** to **1308**, respectively.

FIG. **15** shows the digital ink **1300** of FIG. **13** after the structuring transformation to produce scaled digital ink **1500** which is properly aligned with the line pattern **410** by the earlier described criteria. It is understood that since the beautified digital ink of the present example is a scaled and aligned version of the 'raw' digital ink without adjustment within words, such as inter-character spacing and character sizing, the structuring transformation is equally applicable to the ranges of the characters as segmented. Thus, as shown in FIG. **16**, the bounding boxes **1402** to **1408** of the digital ink **1300** are scaled to bounding boxes **1602** to **1608**, respectively, for the structured digital ink **1500**. These scaled bounding boxes are therefore used by the application **112** to set the cursor position through consideration of geometrical features of the bounding boxes in one or more of the following ways.

In the case that the ranges of two consecutive characters do not overlap, several strategies can be applied by the present system and method in order to place the cursor:

- The cursor is positioned between those two characters at the center between the range borders. For example, FIG. **17**A shows a centered (first) cursor **1700** positioned in the center between the bounding boxes **1604** and **1606** of the non-overlapping respective (first) character **1304** and (second) character **1306**.
- The cursor is positioned between those two characters at the right range border of the left-side character. For example, FIG. **17**B shows a left-aligned (second) cursor **1702** positioned on the right edge of the bounding box **1604** of the character **1304** between the non-overlapping characters **1304** and **1306**.
- The cursor is position between those two characters at the left range border of the right-side character. For example, FIG. **17**C shows a right-aligned (third) cursor **1704** positioned on the left edge of the bounding box **1606** of the character **1306** between the non-overlapping characters **1304** and **1306**.

Whereas, in the case that the ranges of two consecutive characters do overlap, several strategies can be applied by the present system and method in order to place the cursor:

- The cursor is positioned between those two characters at the center of the area of overlap defined by the ranges of the overlapped characters. For example, FIG. **18**A shows a centered (fourth) cursor **1800** positioned in the center of the overlapping area of the (first) character **1302** and (second) character **1304**. As can be seen however, this setting may cause confusion for users as to which character the cursor **1800** is related.
- The cursor is positioned between those two characters in consideration of the other geometrical features of the ranges of the overlapped characters, such as the height, the width, or the area of the bounding boxes to be at the range border of the character with either the smallest or the highest value of these geometrical features which is in the overlapping area. For example, FIG. **18**B shows a left-aligned (fifth) cursor **1802** positioned on the left edge of the bounding box **1604** of the character **1304** between the overlapping characters **1302** and **1304**, where the character having the bounding box of smallest height is used as the cursor positioning criteria.

As with any document editing application, the cursor position can be set and reset by user interaction within the editing area, such as with a tap or a click, provided by the application **112**, as discussed earlier. However, in the afore-described example of the present system and method in which the input or capture area and the rendering or editing area are overlapped, the receipt of a gesture may cause ambiguity between setting of the cursor position and the input of new content. The present system and method accounts for such ambiguity to identify a 'set cursor position' action by evaluating certain criteria with respect to received user interaction. For example, when:

- the input is constituted by a unique stroke, until the end of the writing session; and
- the new input is a single-point gesture, such as a tap or other unique sampling point, defining a user interaction point; and
- content, such as a character or word, is present in the rendering area at the user interaction point, then:

the user interaction is considered as a request to set the cursor position.

The present system and method properly beautifies 'freely' (unstructured) input digital ink through scaling or minimizing and aligning the digital ink on alignment elements of an editing area by taking into account the relative geometrical information of the handwriting, such as the descending part of the handwriting, e.g., the distance between the base-line and the mid-line of the handwriting as determined by the recognition process for recognizing the handwriting input. Further, the present system and method considers character segmentation determined by the recognition process for recognizing the handwriting input to allow management of an editing cursor position for the beautified digital ink so that editing operations can be performed on a character basis, and not only on a word, sentence, line or block basis, for example.

While the foregoing has described what is considered to be the best mode and/or other examples, it is understood that various modifications may be made therein and that the subject matter disclosed herein may be implemented in various forms and examples, and that they may be applied in numerous other applications, combinations, and environments, only some of which have been described herein. Those of ordinary skill in that art will recognize that the disclosed aspects may be altered or amended without departing from the true spirit and scope of the subject matter. Therefore, the subject matter is not limited to the specific details, exhibits, and illustrated examples in this description. It is intended to protect any and all modifications and variations that fall within the true scope of the advantageous concepts disclosed herein.

We claim:

1. A system for providing editing of digital ink on computing devices, each computing device comprising a processor and at least one non-transitory computer readable medium for recognizing handwriting input under control of the processor, the at least one non-transitory computer readable medium configured to:

- cause display of, on a display interface of a computing device, digital ink in accordance with the handwriting input;
- determine, from segmentation information of the handwriting input determined through the recognition of the handwriting input, ranges of elements of the digital ink;
- cause display of, on the display interface, an input cursor in relation to the determined digital ink element ranges; and

cause said display of the input cursor within a region of overlap of ranges of elements of the digital ink determined in the step of determining ranges of elements of the digital ink.

2. A system according to claim **1**, wherein the elements of the digital ink include text characters, the at least one non-transitory computer readable medium configured to display the input cursor at a position between first and second characters in relation to the determined ranges of the first and second characters.

3. A system according to claim **1**, wherein the geometrical features include height and width of the determined ranges.

4. A method for providing editing of digital ink on computing devices, each computing device comprising a processor and at least one non-transitory computer readable medium for recognizing handwriting input under control of the processor, the method comprising:

displaying, on a display interface of a computing device, digital ink in accordance with the handwriting input;

determining, from segmentation information of the handwriting input determined through the recognition of the handwriting input, ranges of elements of the digital ink;

displaying, on the display interface, an editing cursor in relation to the determined digital ink element ranges; and

causing said display of the editing cursor within a region of overlap of ranges of elements of the digital ink determined in the step of determining ranges of elements of the digital ink.

5. A method according to claim **4**, wherein the elements of the digital ink include text characters, the input cursor being displayed at a position between first and second characters in relation to the determined ranges of the first and second characters.

6. A method according to claim **4**, wherein the geometrical features include height and width of the determined ranges.

7. A non-transitory computer readable medium having a computer readable program code embodied therein, said computer readable program code adapted to be executed to implement a method for providing editing of digital ink on a computing device, the computing device comprising a processor and at least one system non-transitory computer readable medium for recognizing handwriting input under control of the processor, the method comprising:

displaying, on a display interface of a computing device, digital ink in accordance with the handwriting input;

determining, from segmentation information of the handwriting input determined through the recognition of the handwriting input, ranges of elements of the digital ink;

displaying, on the display interface, an editing cursor in relation to the determined digital ink element ranges; and

causing said display of the editing cursor within a region of overlap of ranges of elements of the digital ink determined in the step of determining ranges of elements of the digital ink.

8. A non-transitory computer readable medium according to claim **7**, wherein the elements of the digital ink include text characters, the input cursor being displayed at a position between first and second characters in relation to the determined ranges of the first and second characters.

9. A non-transitory computer readable medium according to claim **7**, wherein the geometrical features include height and width of the determined ranges.

* * * * *

EXHIBIT L

US010181076B2

(12) **United States Patent**
Melinand et al.

(10) **Patent No.: US 10,181,076 B2**
(45) **Date of Patent: *Jan. 15, 2019**

(54) **SYSTEM AND METHOD FOR RECOGNIZING GEOMETRIC SHAPES**

(71) Applicant: **MyScript**, Nantes (FR)

(72) Inventors: **Robin Melinand**, Carquefou (FR); **Florent Buisson**, Nantes (FR)

(73) Assignee: **MyScript**, Nantes (FR)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

This patent is subject to a terminal disclaimer.

(21) Appl. No.: **15/146,654**

(22) Filed: **May 4, 2016**

(65) **Prior Publication Data**

US 2016/0247040 A1 Aug. 25, 2016

Related U.S. Application Data

(63) Continuation of application No. 14/558,181, filed on Dec. 2, 2014, now Pat. No. 9,489,572.

(30) **Foreign Application Priority Data**

Nov. 27, 2015 (WO) PCT/EP2015/002400

(51) **Int. Cl.**
G06K 9/36 (2006.01)
G06K 9/00 (2006.01)
(Continued)

(52) **U.S. Cl.**
CPC ***G06K 9/00422*** (2013.01); ***G06F 3/04845*** (2013.01); ***G06F 3/04883*** (2013.01);
(Continued)

(58) **Field of Classification Search**
CPC G06K 9/66; G06K 9/6215; G06K 9/6256; G06K 9/52; G06K 9/6267; G06K 2009/4666; G06T 7/73
(Continued)

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,544,265 A 8/1996 Bozinovic et al.
5,596,656 A 1/1997 Goldberg
(Continued)

FOREIGN PATENT DOCUMENTS

EP 1355275 10/2003
EP 1030263 5/2004
(Continued)

OTHER PUBLICATIONS

Written Opinion and International Search Report issued for PCT/EP2015/002400 dated Mar. 24, 2016.

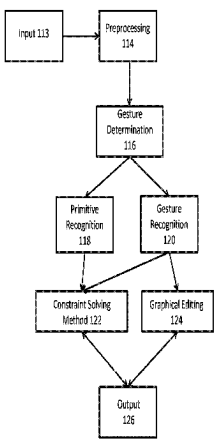
Primary Examiner — Sheela C Chawan

(74) *Attorney, Agent, or Firm* — McDermott, Will & Emery LLP

(57) **ABSTRACT**

A system and method that is able to recognize a user's natural drawing of geometric shapes. The system and method is able to process single-stroke and multi-stroke geometric shapes. It can also apply implicit and user defined constraints to the geometric shapes. The system and method applies these constraints at the vector component level rather than the primitive level. It does this by breaking down both the geometric shapes and constraints on a vector component level. This allows the system and method to handle a larger number of geometric shapes and constraints. After applying the constraints to the geometric shapes at the vector component level, the system and method outputs smooth geometric shapes that incorporated both the implicit and user defined constraints.

27 Claims, 12 Drawing Sheets



(51) **Int. Cl.**

G06F 3/0488	(2013.01)
G06T 11/60	(2006.01)
G06T 11/20	(2006.01)
G06F 3/0484	(2013.01)
G06K 9/48	(2006.01)
G06K 9/52	(2006.01)
G06T 7/60	(2017.01)

(52) **U.S. Cl.**
CPC ***G06K 9/00402*** (2013.01); ***G06K 9/00416*** (2013.01); ***G06K 9/481*** (2013.01); ***G06K 9/52*** (2013.01); ***G06T 7/60*** (2013.01); ***G06T 11/203*** (2013.01); ***G06T 11/60*** (2013.01)

(58) **Field of Classification Search**
USPC 382/156, 189, 107, 120, 123, 188, 203, 382/214, 253, 290, 309, 321; 345/418, 345/419, 420, 440, 441
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,809,267	A	9/1998	Moran et al.	
5,880,743	A *	3/1999	Moran	G06T 13/80 345/473
6,233,351	B1	5/2001	Feeney et al.	
6,292,190	B1	9/2001	Corn	
6,411,732	B1	6/2002	Saund	
6,459,442	B1	10/2002	Edwards et al.	
6,476,834	B1	11/2002	Doval et al.	
7,256,773	B2 *	8/2007	Kolmykov-Zotov	G06F 3/04883 178/18.01
7,324,691	B2	1/2008	Li et al.	
7,352,902	B2	4/2008	Li et al.	
7,385,595	B2	6/2008	Bryborn et al.	
7,639,250	B2	12/2009	Xu et al.	
8,102,383	B2	1/2012	Cohen et al.	
8,159,501	B2	4/2012	Rao	
8,564,620	B2	10/2013	Bellamy et al.	
8,670,623	B2	3/2014	Oota	
8,718,375	B2	5/2014	Ouyang et al.	
2005/0275622	A1	12/2005	Patel et al.	
2008/0198146	A1	8/2008	Bryborn et al.	
2009/0278848	A1	11/2009	Robertson et al.	
2010/0100866	A1	4/2010	Kamper et al.	
2010/0149109	A1	6/2010	Elias	
2010/0251188	A1	9/2010	Fu et al.	
2011/0185272	A1	7/2011	Rajkumar	
2011/0279455	A1	11/2011	McDaniel	
2012/0176416	A1	7/2012	Dondurur et al.	
2012/0221938	A1	8/2012	Patterson et al.	

FOREIGN PATENT DOCUMENTS

EP	1519300	*	7/2009
EP	1454225		1/2013
WO	2012115853		8/2012
WO	2014046302		3/2014

* cited by examiner

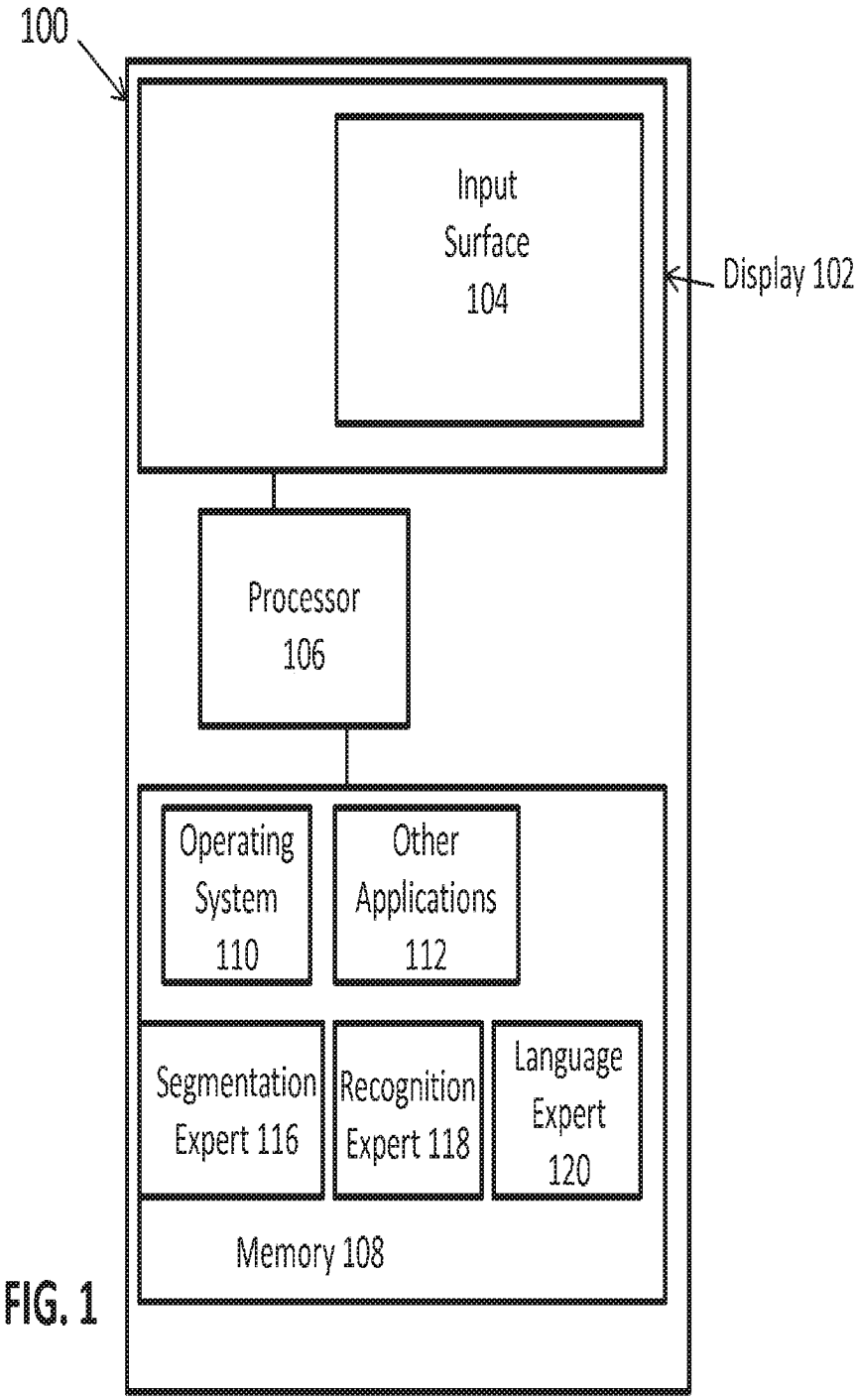

100
Input Surface 104
Display 102
Processor 106
Operating System 110
Other Applications 112
Segmentation Expert 116
Recognition Expert 118
Language Expert 120
Memory 108

FIG. 1

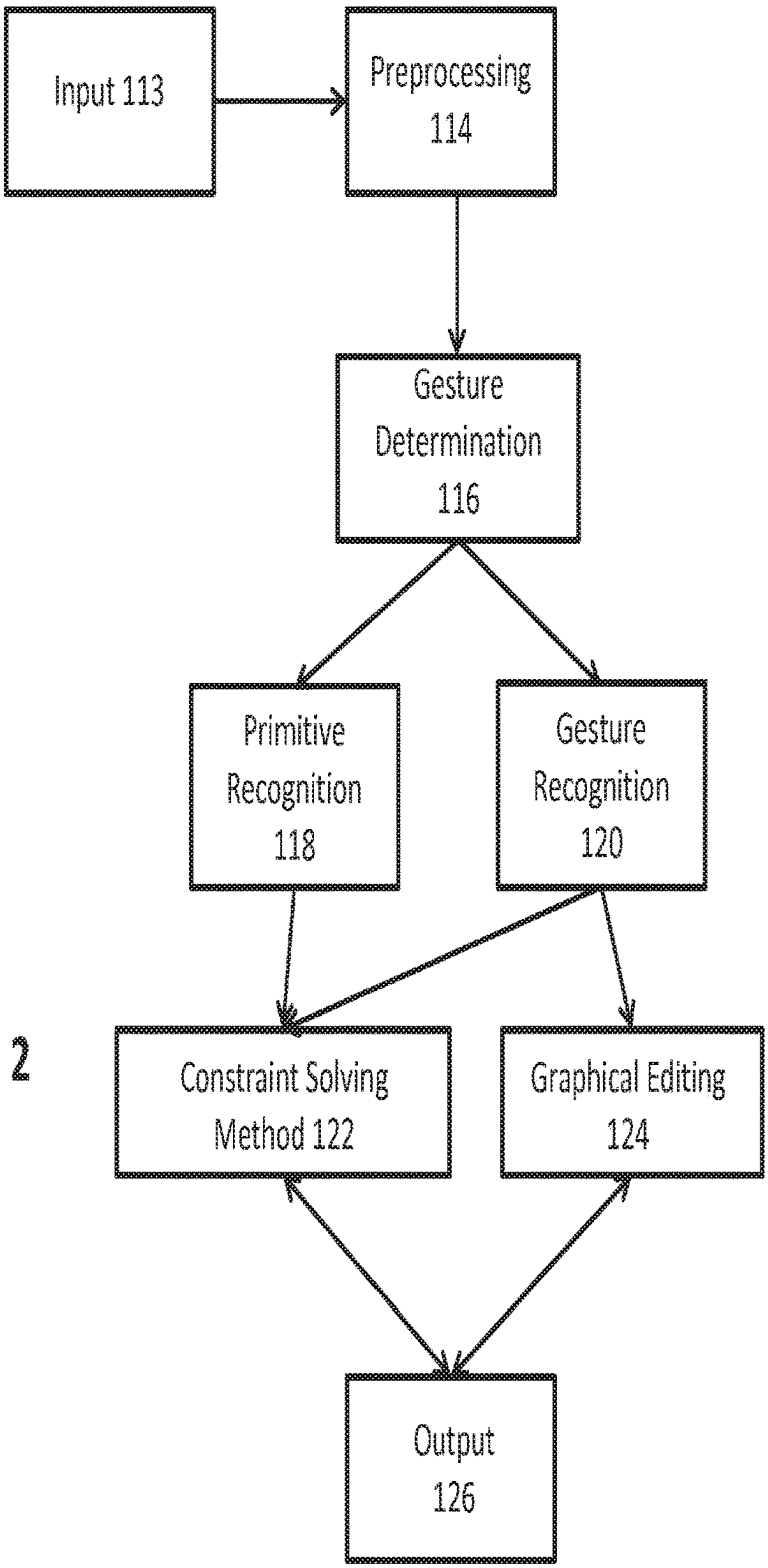

Input 113
Preprocessing 114
Gesture Determination 116
Primitive Recognition 118
Gesture Recognition 120
Constraint Solving Method 122
Graphical Editing 124
Output 126

FIG. 2

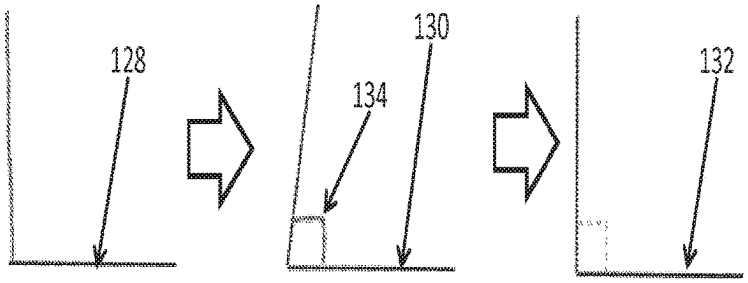


FIG. 3A

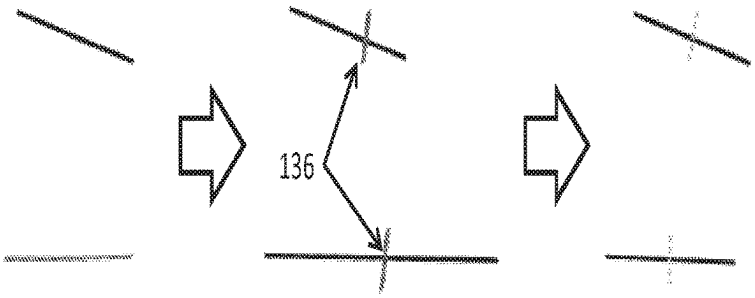


FIG. 3B

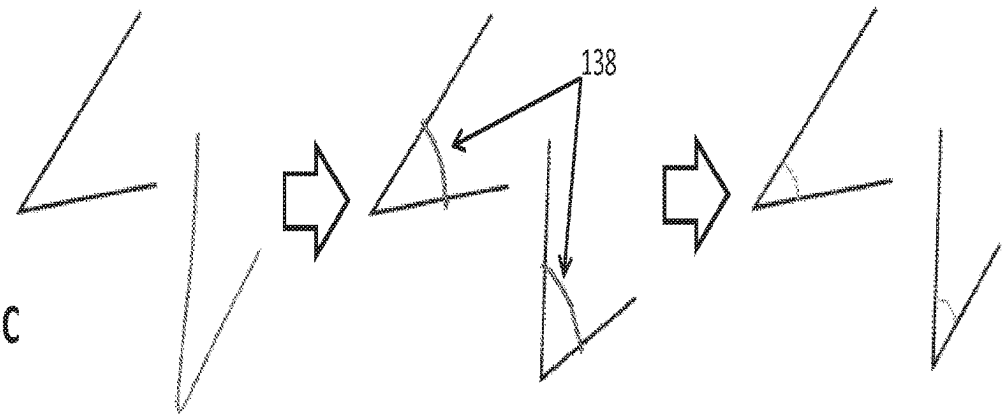


FIG. 3C

FIG. 3D

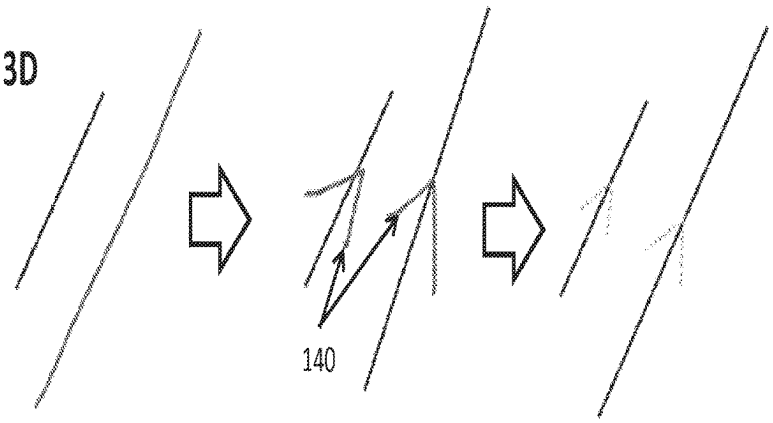


FIG. 3E

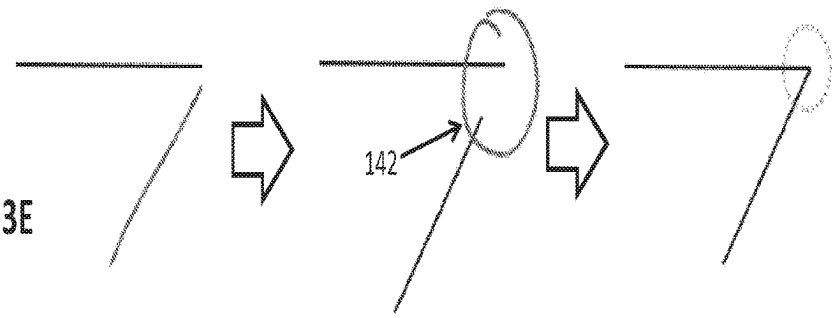


FIG. 3F

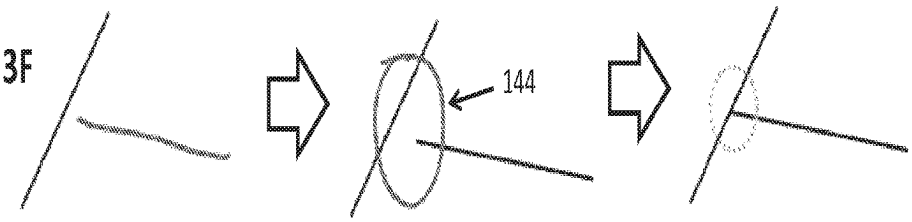
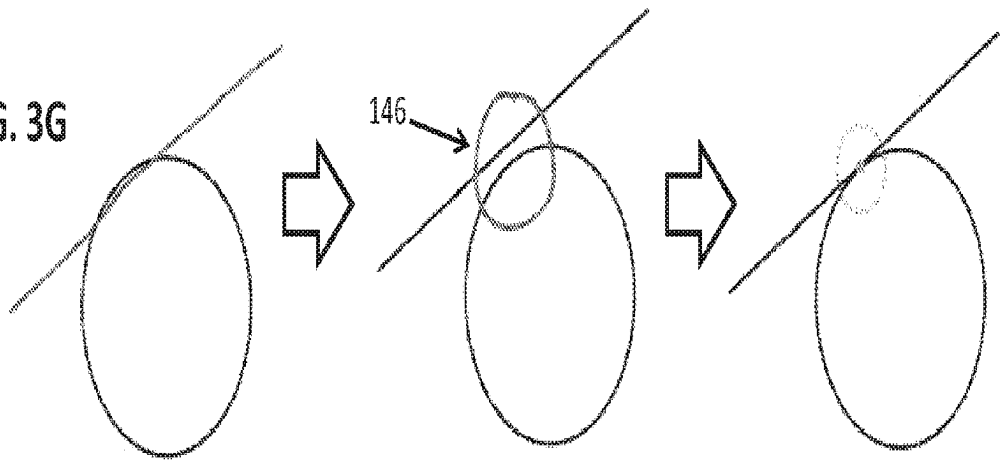


FIG. 3G



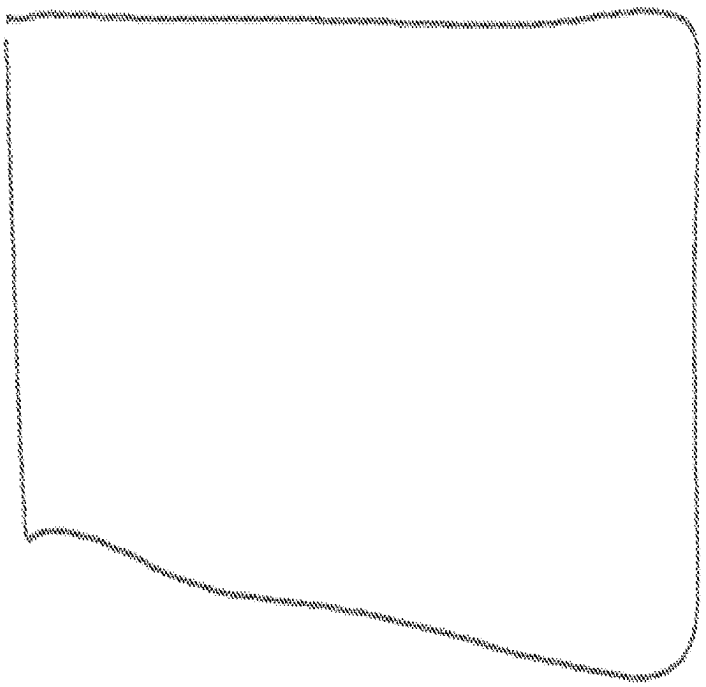


FIG. 4A

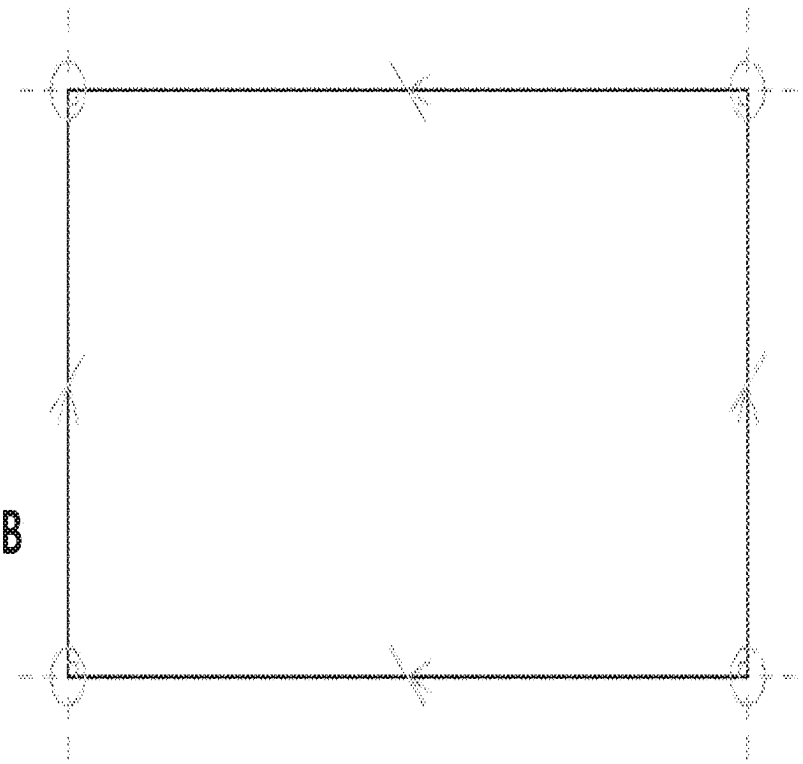


FIG. 4B

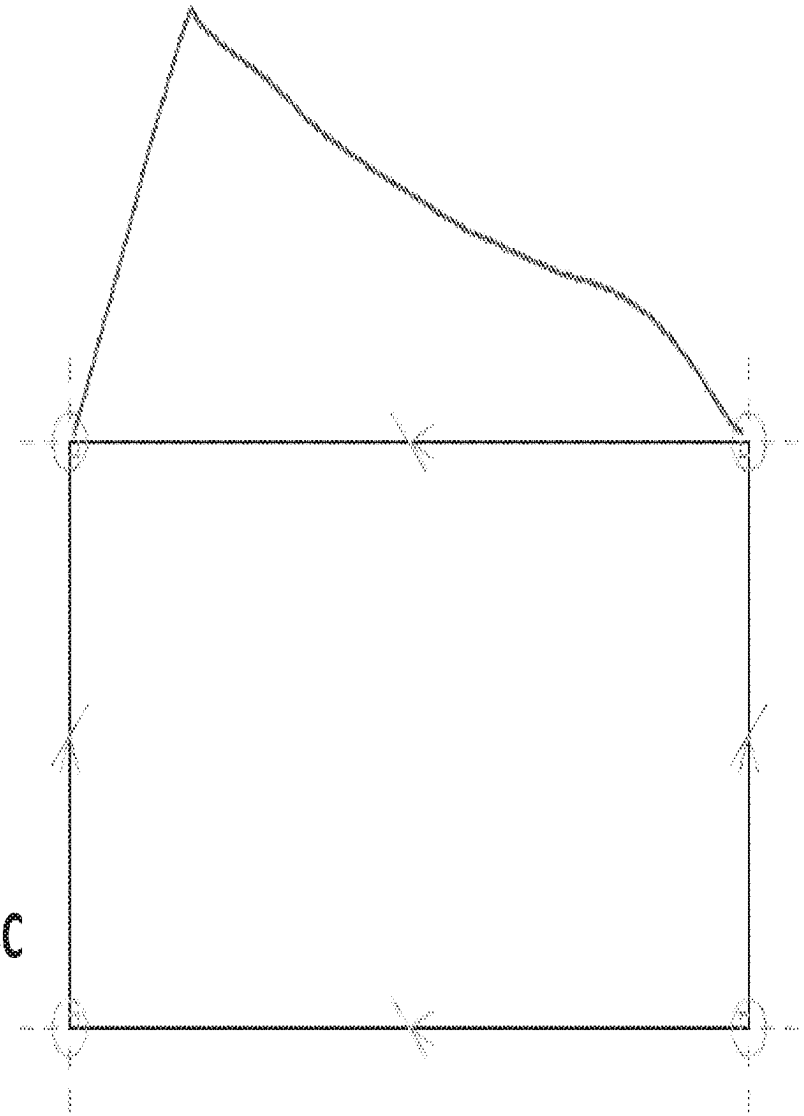


FIG. 4C

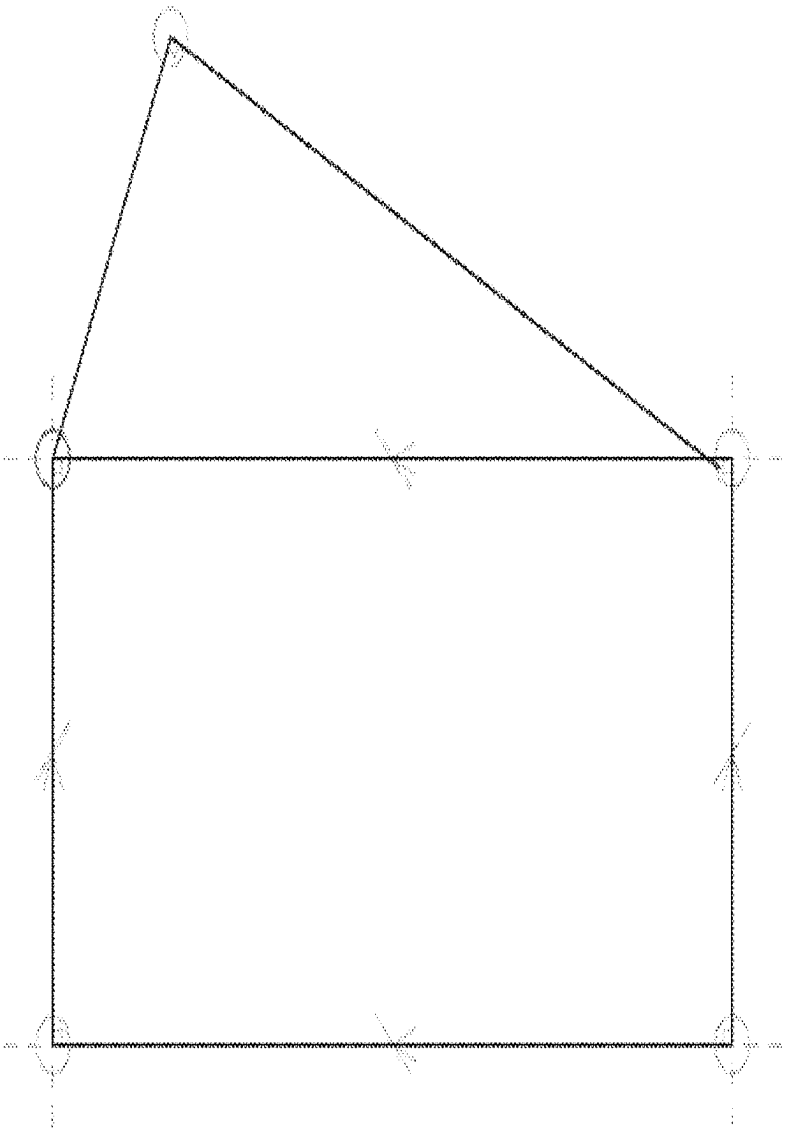


FIG. 4D

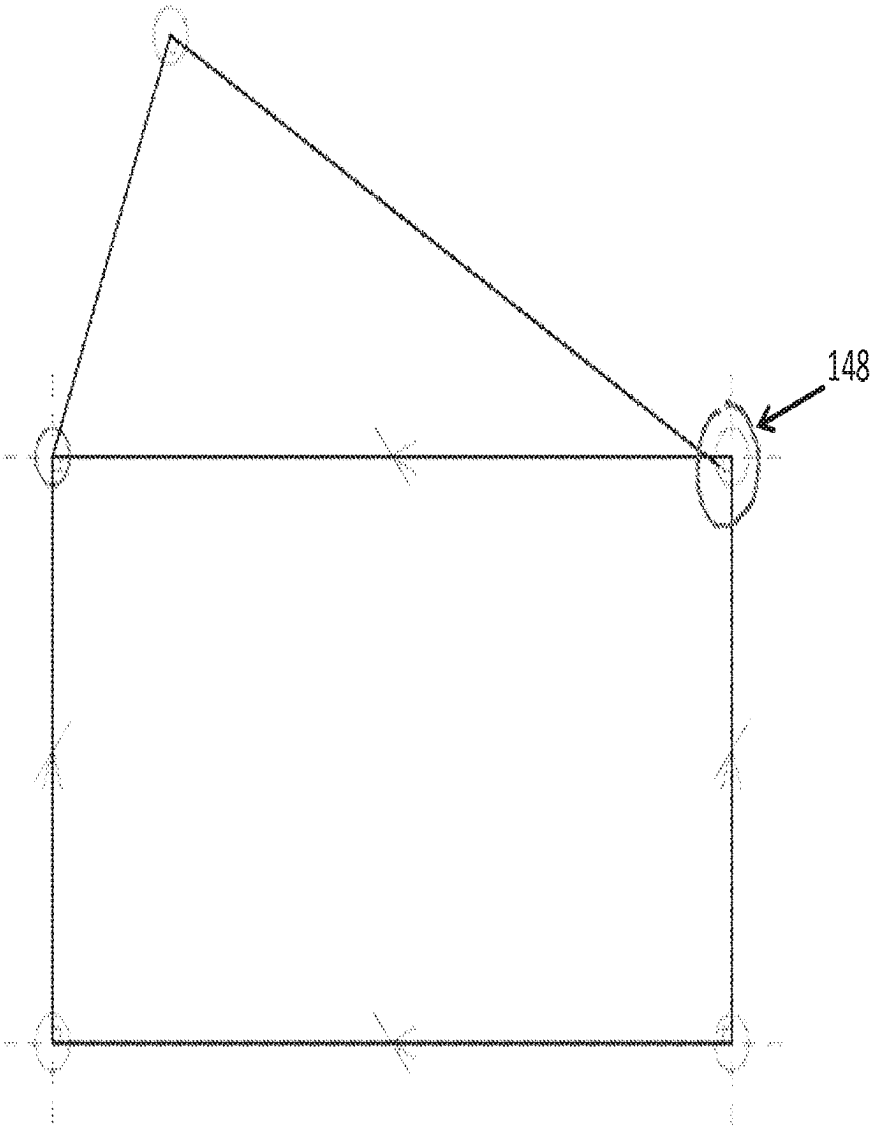


FIG. 4E

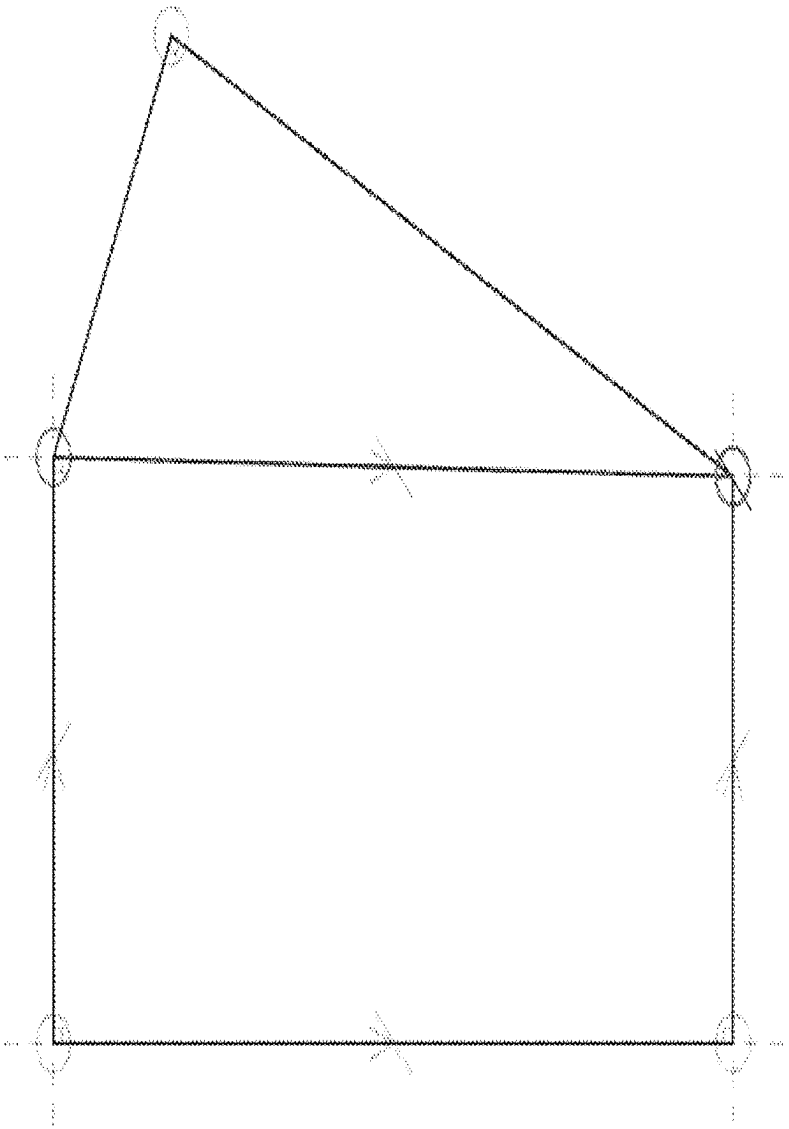


FIG. 4F



FIG. 5A

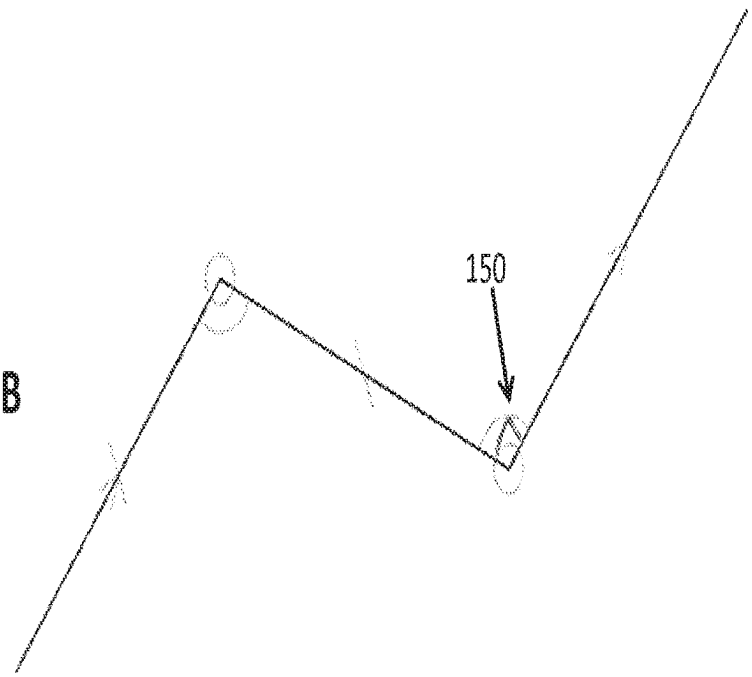


FIG. 5B

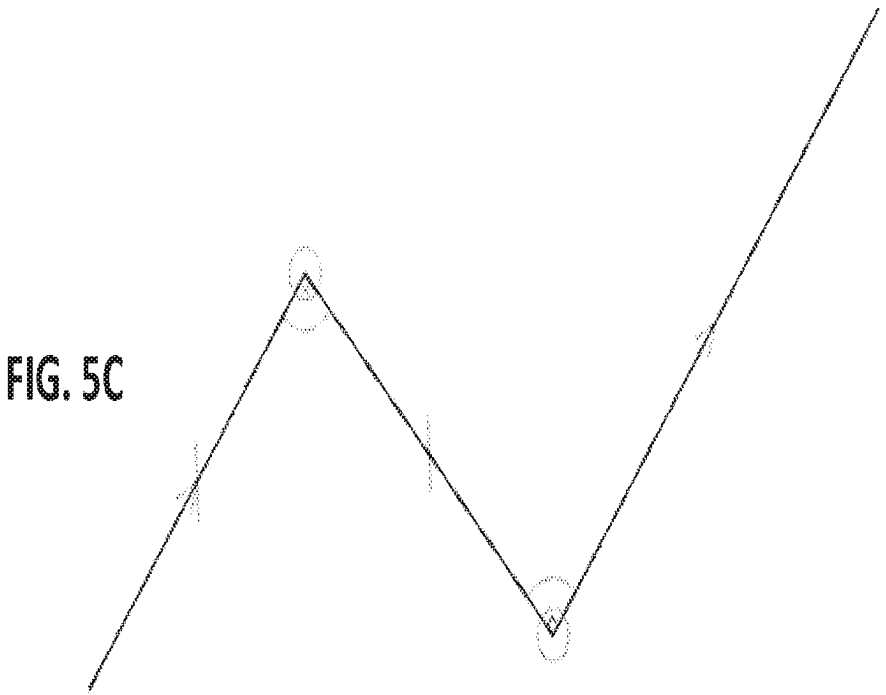


FIG. 5C

SYSTEM AND METHOD FOR RECOGNIZING GEOMETRIC SHAPES

CROSS-REFERENCE TO RELATED APPLICATIONS

This application is a Continuation of U.S. application Ser. No. 14/558,181 filed on Dec. 2, 2014. This application also claims priority to PCT/EP2015/002400 filed on Nov. 27, 2015. The entire contents of these applications are incorporated by reference herein.

TECHNICAL FIELD

The present invention relates generally to the field of computing device interfaces capable of recognizing user input of geometric shapes.

BACKGROUND

Computing devices continue to become more ubiquitous to daily life. Such devices take the form of computer desktops, laptops, tablet PCs, e-book readers, interactive whiteboards, mobile phones, smartphones, wearable computers, global positioning system (GPS) units, enterprise digital assistants (EDAs), personal digital assistants (PDAs), game consoles, and the like. Further, computing devices are being incorporated into cars, trucks, farm equipment, manufacturing equipment, building environment control (e.g., lighting, HVAC), and home and commercial appliances.

Computing devices generally consist of at least one processing element, such as a central processing unit (CPU), some form of memory, and input and output devices. The numerous varieties of computing devices as well as their subsequent uses necessitate a variety of input devices. One type of input device is a touch sensitive surface such as a touch screen or touch pad wherein user input is received through contact between the user's finger or an instrument such as a pen or stylus and the touch sensitive surface. Another type of input device is an input surface that senses motions made by a user above the input surface and without touching the input surface. Either of these methods of input can be used generally for drawing or inputting shapes. When the user input is a shape, the computing device must interpret the user's natural drawing technique using, for example, an on-line geometric shape recognition system or method.

Generally, on-line geometric shape recognition systems or methods monitor the initiation of a stroke, such as when the user contacts a touch sensitive surface (pen-down); the termination of a stroke, such as when the user stops contacting a touch sensitive surface (pen-up); and any movements (i.e. the stroke) the user makes with his or her finger or pen between the initiation and termination of the stroke.

On-line geometric shape recognition systems or methods may be classified as single-stroke or multi-stroke. Single-stroke recognition analyzes each single-stroke as a separate primitive. A primitive is a geometric shape. The system may only utilize single-stroke primitives or may utilize shorthand for primitives that would otherwise require multiple strokes. These single-stroke systems or methods typically have less input errors but may require users to memorize stroke patterns for multi-stroke primitives if shorthand is utilized. Multi-stroke recognition systems can recognize natural drawing and are more intuitive to users because one does not have to learn shorthand.

Whether one uses a single or multi-stroke system, on-line geometric shape recognition systems or methods usually consist of preprocessing, primitive recognition, and constraint solving. In practice, on-line geometric shape recognition systems or methods may include these steps along with additional steps. Further, on-line geometric shape recognition systems or methods may not clearly delineate each step or may perform the steps in an alternate order.

Preprocessing typically includes discarding irrelevant input data and normalizing, sampling, and removing noise from relevant data.

Primitive recognition specifies the different ways to break down the input data into individual lines, shapes, or other segments based on a set of basic models (e.g., lines, circles, ellipses, arcs, and points). Primitive recognition generally associates the segments with possible primitive candidates. Primitives can be any type of shape or curve, from basic (e.g., line segment, circle, ellipse, arc of circle, arc of ellipse, and point) to more complex (e.g., sinusoidal, spline, Bezier, and parametric function plots). Primitive candidates can comprise single-stroke, multi-stroke, and/or geometric variations of shapes. Primitive recognition may also be preceded by gesture determination that determines whether the input data is geometric shape or a gesture (e.g., an input stroke that provides additional information as to the relation of already input geometric shapes).

Constraint solving generally includes checking the primitives against a set of constraints (e.g., length equality, angle equality, parallelism, symmetry, alignment, tangency, connection, and perpendicularity). Typically, constraints are defined on a primitive level, meaning they are associated with specific primitives. Constraints can be implicit or explicit. Implicit constraints are such that the system infers a constraint based on user input of primitives. For example, the system or method may detect that two lines are close enough to being perpendicular and infer that a perpendicular restraint should apply, outputting two perpendicular lines. In this example, perpendicularity is an example of an implicit constraint. Explicit constraints are such that the user explicitly specifies additional constraints to apply. For instance, the system or method may not detect that two lines were intended to be perpendicular. A user may input an additional gesture to provide this explicit constraint. An on-line geometric shape recognition system or method may recognize both implicit and explicit constraints. Once the set of primitives and set of constraints are built, a constraint solving method enforces the maximum number of possible constraints. Following constraint solving, the on-line geometric shape recognition system or method outputs smooth geometric shapes.

Constraint solving at the primitive level is usually limited in the number of primitives and constraints that a system can analyze because constraints are generally defined at the primitive level and closely linked to primitive types. Each new primitive or constraint introduced into the system causes exponentially more relationships between primitives and constraints, creating exponentially more possibilities to analyze during constraint solving.

The present system and method provides improved results for user input recognition by defining the constraints at a vector component level rather than primitive level. By defining constraints on the vector component level, the system and method is able to better handle increasingly large sets of primitives and constraints.

SUMMARY

The examples of the present invention that are described herein below provide methods, systems, and software for

use in on-line geometric shape recognition. These permit a user to enter geometric shapes into a computing device as one would on a piece of paper. The present shape recognition system and method includes a computing device connected to an input device in the form of an input surface. A user is able to provide input by applying pressure to or gesturing above the input surface using either his or her finger or an instrument such as a stylus or pen. The present system and method monitors the input strokes. After preprocessing the input strokes, the present system and method determines whether or not the input stroke is a gesture. If the input stroke is a gesture, the present system and method matches the gesture with an instruction (e.g., add an additional constraint or delete a drawing) to apply to a preexisting drawing. If the input is not a gesture, the present system and method determines which primitive closely matches the input stroke. Finally, the present system and method applies the set of constraints to the set of primitives to output smooth geometric shapes. After the output is displayed as a drawing on the output device, the user may proceed to adjust or otherwise enrich the output drawing with more shapes or constraints.

An object of the disclosed system and method is to provide an on-line geometric shape recognition system and method that can interpret a user's natural drawing style and output cleaned-up geometric shapes. This can be done by providing a system and method whereby the input strokes are associated with either primitives or gestures that the user can use to explicitly define constraints. Using the primitives and constraints associated with the user input, the system is able to output smooth geometric shapes.

Yet a further object of the disclosed system and method is to provide a simpler definition of constraints. This can be done by providing a system and method that defines constraints on a vector component level rather than on a primitive level.

Another object of the disclosed system and method is to relate primitives and constraints at the vector component level. This can be done by providing a system and method that breaks down primitives into sets of vector components and analyzes the relationship between the constraints and primitives at the vector component level.

An object of the disclosed system and method is also to provide an on-line geometric shape recognition system and method that allows further editing of the output. This can be done by providing a system and method that recognizes certain input editing gestures.

The present system and method will be more fully understood from the following detailed description of the examples thereof, taken together with the drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** shows a block diagram of a computing device in accordance with an example of the present system.

FIG. **2** shows a block diagram of a system for on-line geometric shape recognition in accordance with an example of the present system.

FIG. **3**A shows a pictorial illustration of a possible perpendicularity constraint in accordance with an example of the present system.

FIG. **3**B shows a pictorial illustration of a possible length equality constraint in accordance with an example of the present system.

FIG. **3**C shows a pictorial illustration of a possible angle equality constraint in accordance with an example of the present system.

FIG. **3**D shows a pictorial illustration of a possible parallelism constraint in accordance with an example of the present system.

FIG. **3**E shows a pictorial illustration of a possible junction constraint involving ends of line segments in accordance with an example of the present system.

FIG. **3**F shows a pictorial illustration of a possible junction constraint involving the end of a line segment and line segment in accordance with an example of the present system.

FIG. **3**G shows a pictorial illustration of a possible junction constraint involving a line segment and a portion of a circle in accordance with an example of the present system.

FIG. **4**A shows a pictorial illustration of a possible user input in accordance with an example of the present system.

FIG. **4**B shows a pictorial illustration of a possible output in accordance with an example of the present system.

FIG. **4**C shows a pictorial illustration of a possible output with possible additional user input in accordance with an example of the present system.

FIG. **4**D shows a pictorial illustration of a possible output in accordance with an example of the present system.

FIG. **4**E shows a pictorial illustration of a possible output with possible additional user input in accordance with an example of the present system.

FIG. **4**F shows a pictorial illustration of a possible output in accordance with an example of the present system.

FIG. **5**A shows a pictorial illustration of a possible user input in accordance with an example of the present system.

FIG. **5**B shows a pictorial illustration of a possible output with possible additional user input in accordance with an example of the present system.

FIG. **5**C shows a pictorial illustration of a possible output in accordance with an example of the present system.

DETAILED DESCRIPTION

In the following detailed description, numerous specific details are set forth by way of examples in order to provide a thorough understanding of the relevant teachings. However, it should be apparent to those skilled in the art that the present teachings may be practiced without such details. In other instances, well known methods, procedures, components, and/or circuitry have been described at a relatively high-level, without detail, in order to avoid unnecessarily obscuring aspects of the present teachings.

The various technologies described herein generally relate to on-line geometric shape recognition. The system and method described herein may be used to recognize a user's natural drawing style input through the processes of primitive recognition and constraint solving. The process of constraint solving can include the breaking down and comparison of primitives and constraints at the vector component level.

FIG. **1** shows a block diagram of an exemplary computing device **100**. The computing device may be a computer desktop, laptop, tablet PC, e-book reader, mobile phone, smartphone, wearable computer, digital watch, interactive whiteboard, global positioning system (GPS) unit, enterprise digital assistant (EDA), personal digital assistant (PDA), game console, or the like. Device **100** includes at least one processing element, some form of memory and input and/or output (I/O) devices. The components communicate with each other through inputs and outputs, such as connectors, lines, buses, cables, buffers, electromagnetic links, net-

works, modems, transducers, IR ports, antennas, or others known to those of ordinary skill in the art.

The device **100** includes at least one input surface **104**. The input surface **104** may employ technology such as resistive, surface acoustic wave, capacitive, infrared grid, infrared acrylic projection, optical imaging, dispersive signal technology, acoustic pulse recognition, or any other appropriate technology as known to those of ordinary skill in the art. The input surface **104** may be bounded by a permanent or video-generated border that clearly identifies its boundaries.

In addition to the input surface **104**, the device **100** may include one or more additional I/O devices (or peripherals) that are communicatively coupled via a local interface. The local interface may have additional elements to enable communications, such as controllers, buffers (caches), drivers, repeaters, and receivers, which are omitted for simplicity but known to those of skill in the art. Further, the local interface may include address, control, and/or data connections to enable appropriate communications among the other computer components.

One such I/O device may be at least one display **102** for outputting data from the computing device such as images, text, and video. The display **102** may use LCD, plasma, CRT, or any other appropriate technology as known to those of ordinary skill in the art. At least some or all of display **102** may be co-located with the input surface **104**. Other additional I/O devices may include input devices such as a keyboard, mouse, scanner, microphone, touchpads, bar code readers, laser readers, radio-frequency device readers, or any other appropriate technology known to those of ordinary skill in the art. Furthermore, the I/O devices may also include output devices such as a printer, bar code printers, or any other appropriate technology known to those of ordinary skill in the art. Finally, the I/O devices may further include devices that communicate both inputs and outputs such as a modulator/demodulator (modem; for accessing another device, system, or network), a radio frequency (RF) or other transceiver, a telephonic interface, a bridge, a router, or any other appropriate technology known to those of ordinary skill in the art.

The device **100** also includes a processor **106**, which is a hardware device for executing software, particularly software stored in the memory **108**. The processor can be any custom made or commercially available general purpose processor, a central processing unit (CPU), a semiconductor based microprocessor (in the form of a microchip or chipset), a macroprocessor, microcontroller, digital signal processor (DSP), application specific integrated circuit (ASIC), field programmable gate array (FPGA) or other programmable logic device, discrete gate or transistor logic, discrete hardware components, state machine, or any combination thereof designed for executing software instructions known to those of ordinary skill in the art. Examples of suitable commercially available microprocessors are as follows: a PA-RISC series microprocessor from Hewlett-Packard Company, an 80x86 or Pentium series microprocessor from Intel Corporation, a PowerPC microprocessor from IBM, a Sparc microprocessor from Sun Microsystems, Inc., a 68xxx series microprocessor from Motorola Corporation, DSP microprocessors, or ARM microprocessors.

The memory **108** can include any one or a combination of volatile memory elements (e.g., random access memory (RAM, such as DRAM, SRAM, SDRAM, etc.)) and nonvolatile memory elements (e.g., ROM, EPROM, flash PROM, EEPROM, hard drive, magnetic or optical tape, memory registers, CD-ROM, WORM, DVD, redundant array of inexpensive disks (RAID), another direct access storage device (DASD), etc.). Moreover, memory **108** may incorporate electronic, magnetic, optical, and/or other types of storage media. The memory **108** can have a distributed architecture where various components are situated remote from one another but can also be accessed by the processor **106**. The memory **108** is coupled to a processor **106**, so the processor **106** can read information from and write information to the memory **108**. In the alternative, the memory **108** may be integral to the processor **106**. In another example, the processor **106** and the memory **108** may both reside in a single ASIC or other integrated circuit.

The software in memory **108** includes the on-line geometric shape recognition computer program, which may include one or more separate programs, each of which comprises an ordered listing of executable instructions for implementing logical functions. The operating system **110** controls the execution of the on-line geometric shape recognition computer program. The operating system **110** may be any proprietary operating system or a commercially available operating system, such as WEBOS, WINDOWS®, MAC and IPHONE OS®, LINUX, and ANDROID. It is understood that other operating systems may also be utilized without departing from the spirit of the system and method disclosed herein.

The memory **108** may include other application programs **112** related to geometric shape recognition as described herein, totally different functions, or both. The applications **112** include programs provided with the device **100** upon manufacture and may further include programs downloaded into the device **100** after manufacture. Some examples include a text editor, telephone dialer, contacts directory, instant messaging facility, computer-aided design (CAD) program, email program, word processing program, web browser, and camera.

The on-line geometric shape recognition computer program with support and compliance capabilities may be a source program, executable program (object code), script, or any other entity comprising a set of instructions to be performed. When a source program, the program needs to be translated via a compiler, assembler, interpreter, or the like, which may or may not be included within the memory, so as to operate properly in connection with the operating system. Furthermore, the on-line geometric shape recognition computer program with support and compliance capabilities can be written as (a) an object oriented programming language, which has classes of data and methods; (b) a procedure programming language, which has routines, subroutines, and/or functions, for example but not limited to C, C++, Pascal, Basic, Fortran, Cobol, Perl, Java, and Ada; or (c) functional programming languages for example but no limited to Hope, Rex, Common Lisp, Scheme, Clojure, Racket, Erlang, OCaml, Haskell, and F#.

The system may be initiated when processor **106** detects a user entered stroke via the input surface **104**. The user may enter a stroke with a finger or some instrument such as a pen or stylus. The user may also enter a stroke by making gesture above input surface **104** if technology that senses motions above input surface **104** is being used. A stroke is characterized by at least the stroke initiation location, the stroke termination location, and the path upon which the user connects the stroke initiation and termination locations.

FIG. **2** is a schematic pictorial of a system for on-line geometric shape recognition in accordance with an example of the present system. Once initiated with at least one input stroke **113**, the system preprocesses **114** the strokes. Preprocessing **114** can include normalizing the path connecting

the stroke initiation and termination locations by applying size normalization and/or methods such as B-spline approximation to smooth the input.

After preprocessing, the system determines whether the stroke is a gesture or not in the context of an existing drawing. A gesture is an input stroke that provides additional information regarding the existing drawing. If the gesture determination **116** recognizes a gesture, the system then passes the input to gesture recognition **120**. The system may then recognize the input as an annotation defining an explicit constraint the user may wish to impose on existing geometric shapes. In one embodiment, the gesture recognition **120** may use manually tuned heuristics based on polyline approximation of the gesture strokes and their position compared to the primitives to detect the gestures and distinguish between them in context of the existing drawing. However, any gesture recognition **120** method known to those of ordinary skill in the art may be used. The recognized gesture is passed on to the constraint solving method **122**. The constraint solving method **122** will be discussed below. The gesture may also be associated with graphical editing instructions. In this case, the gesture is passed onto the graphical editing **124**.

The graphical editing **124** may be any method known to those of ordinary skill in the art to edit existing output **126**. Such graphical editing **124** typically uses information from the gesture recognition **120** and the output **126** to adjust the current drawing by erasing it, filling in shapes with patterns or colors, changing the thickness or highlighting certain lines, or otherwise editing the current output **126** in ways known to those of ordinary skill in the art.

If gesture determination **116** does not detect a gesture, the input is passed to the primitive recognition **118**. Primitives may include the basic geometric shapes (e.g., line segment, circle, ellipse, arc of circle, arc of ellipse, and point) or additionally more complex shapes (e.g., sinusoidal, spline, Bezier, and parametric function plots). The system may also define a "junk" primitive, which cause the system to discard the input as an unrecognized primitive. Depending on the number of primitives defined in the system, primitive recognition **118** may include single-stroke primitive recognition that treats each stroke as a separate primitive to process. Primitive recognition **118** may also include multi-stroke recognition that treats multiple consecutive input strokes as single primitive.

The system may employ any combination of single and multi-stroke recognition methods known to those of ordinary skill in the art to determine which primitive most closely matches the input **113**. One example of primitive recognition **118** is to fit the input **113** with all primitives, giving a score for each fitting attempt. Dynamic programming may then be used to select the best set of primitives to encompass all of the input strokes based on the best overall fitting scores. Once the best set of primitives has been chosen, the set of primitives is sent to the constraint solving method **122**.

The constraint solving method **122** may receive input from the primitive recognition **118**, gesture recognition **120**, and/or output **126**. Constraints link primitives together (e.g., length equality, angle equality, parallelism, symmetry, alignment, tangency, connection, and perpendicularity). FIGS. **3**A-**3**G are pictorial illustrations of possible constraints. In each case, the far left illustration is the input **113** and the far right illustration is the output **126**. The middle illustration is the application of constraints by the constraint solving method **122**. Constraints can be implicit, deduced by the constraint solving method **122** depending on the primitives provided by the primitive recognition **118**, or can be explicit, defined by the user through gestures. For instance, the constraint solving method **122** may deduce the two line segments input **128** by the user in FIG. **3**A were intended to be perpendicular and automatically apply the perpendicularity constraint, outputting **132** the far right illustration of FIG. **3**A. In this example, the middle illustration **130** represents the constraint solving method **122** adding the perpendicularity constraint internally.

In another instance, the constraint solving method **122** may not detect that the user intended two perpendicular line segments in FIG. **3**A and output the non-perpendicular line segments in the middle illustration **130**. The user may then annotate the junction of the two line segments with a square **134**, which the system would recognize as a gesture. The gesture recognition **120** would pass this gesture onto the constraint solving method **122**, which would adjust the output to be two perpendicular line segments, as seen in the output **132** of far right illustration in FIG. **3**A.

In FIGS. **3**A-**3**G, the constraints can be deduced by the constraint solving method **122** or explicitly annotated by the user. The annotations of these figures (i.e., a square **134** for perpendicularity in FIG. **3**A, two small line segments **136** for length equality in FIG. **3**B, two arcs **138** for angle equality in FIG. **3**C, two arrows **140** for parallelism in FIG. **3**D, a circle **142** around the ends of line segments to form an junction in FIG. **3**E, a circle **144** around the end of a line segment and line segment to form a junction in FIG. **3**F, and a circle **146** around a line segment and portion of a circle for a junction or tangency in FIG. **3**G) are one embodiment, and the present system and method may use any gestures for these constraints as well as others. Further, constraints naturally associated with primitives may be deduced during primitive recognition **118** prior to the constraint solving method **122**. For instance, the primitive recognition **118** may recognize a square from the input and also apply the constraints of four equal sides, four internal right angles, and parallel opposite sides. If primitive recognition **118** recognizes a polyline, it may also apply the junction constraint where the lines intersect. However, for clarity this specification will only refer to the constraint solving method **122** as applying all constraints even though it is envisioned that the primitive recognition **118** may handle this function in some situations.

The constraint solving method **122** includes defining constraints and primitives at the vector component level rather than at a primitive level. When defining constraints at the primitive level, constraints are associated with specific primitives. However, by defining constraints at the vector component level a user may add additional constraints and primitives without adding exponentially more work. For example, when the junction constraint is defined and applied at the primitive level with two line segments, it must first be associated with the beginning or ending of a line segment. Once a circle primitive is added to the present system and method, the junction constraint must also be associated with the circle center together with other line segment externalities. This constraint becomes even more complicated once ellipses and other more complex primitives are added to the present system and method, even though the constraint is still defining just a junction between two points.

To avoid this exponential increase in complexity, the constraint solving method **122** breaks to down both primitives and constraints at the vector component level, such that the method focuses on the relationships between points. In one embodiment, the constraint solving method **122** may define at least three items: points, lengths, and slope angles.

To define primitives as items, the constraint solving method **122** may break down the primitives into their items. For example, a line segment primitive may be defined by the two extremity points, a slope, and a length. Another example is a circle primitive defined by a central point and radius length. With the primitives broken down to their items, the constraint solving method **122** applies vector component constraints. Even though the constraint solving method **122** may include breaking down the primitives into their vector component items, in other embodiments, the primitive recognition **118** may break down the primitives into the vector components prior to passing the information to the constraint solving method **122**. In this alternative embodiment, the primitive items from the primitive recognitions **118** would still have constraints applied to them by the constraint solving method **122**.

Unlike constraints defined on the primitive level, constraints defined at the vector component level link items together. Vector component constraints may include:

- Slope Angle Constraint defined between two points and one slope angle;
- Length Constraint defined between two points (A, B) and one length (L) so that ||AB||=L;
- Projection Constraint defined between two points (A, B), a length (L) and a slope(s) so that B is the projection of A following slope(s) at distance (L) and vice-versa;
- Junction Constraint defined between two points (A, B) to ensure A=B;
- Slope Angle Difference Constraint defined between two slope angles (s**1**, s**2**) and a given value (v) so that s**2**−s**1**=v;
- Length Ratio Constraint defined between two lengths (L**1**, L**2**) and a given value (r) so that L**1** /L**2** =r;
- Length Sum Constraint defined between three points (A, B, C) and one length (L) so that ||AC||+||BC||=L; and
- Existence Constraint defined for any point, and useful if it is isolated and not linked by any other constraint

Other constraints may be defined in a similar manner as would be known to those of skill in the art. Further, any constraint may be defined as a combination of vector component constraints. One benefit of using vector component constraints is that the constraint solving method **122** may deduce the value of an item if some of the other items are known. As an example, the Length Constraint links two point items (A, B) to a length item L. So, if A and B are known, the value of L can be determined. Conversely, if A and L are known, B can be defined as on a circle of radius L centered on A, which information is added to the intersection parameter of B. The same applies if we switch A and B. By breaking down both primitives and constraints on a vector component level the present system and method will be able to solve more situations.

Once the constraint solving method **122** has broken down the primitives into items, the constraint solving method **122** may apply all possible constraints to the primitives or may build a list of specific constraints based on priority. By placing constraints in a hierarchy, the constraint solving method **122** avoids applying contradictory constraints. For instance, constraints that maintain primitive integrity may be high priority. This ensures that primitives do not change their inherent shape (e.g., circles will remain a point and a radius). A second priority may constraints that the constraint solving method **122** implicit in the primitives such as a junction of a polyline. Other lower priority constraints may be those defined explicitly by the user (e.g., user defined parallelism) or those for pairs of primitives (e.g., a line segment tangentially intersecting to a circle). However, this hierarchy is only one example, and the present system and method is able to prioritize constraints in any manner one skilled in the art would necessitate. After the constraint solving method **122** applies the proper constraints to the primitives, it rebuilds the primitives from their vector component levels to produce output **126** in the form of smooth geometric shapes.

FIGS. **4**A-**4**F show a schematic pictorial illustration of possible inputs and outputs relating to a rectangle and polyline in accordance with an example of the present system. FIG. **4**A shows a schematic illustration of an input of a rectangle. After preprocessing **114**, the gesture determination **116** would not define this input as a gesture and pass the input on for primitive recognition **118**. After primitive recognition **118** determined the input was a rectangle, the information would be passed on to the constraint solving method **122**. The constraint solving method **122** would apply the constraints known for a rectangle (e.g., four line segments that intersect at four interior right angles, and opposite sides are parallel and of equal length). This would produce the rectangle shown in FIG. **4**B as output.

A user may further add to the rectangle with input shown FIG. **4**C. The primitive recognition **118** would recognize this as a polyline primitive. The constraint solving mechanism **122** would also detect a junction in the middle of the new polyline that is a right angle as well as junction with corner of the previous rectangle. This would produce new output shown in FIG. **4**D. A user may want another junction that was not detected and add annotation **148** to the output as shown in FIG. **4**E. This annotation **148** would be processed as a gesture, recognized as a gesture to indicate a junction, and passed on to the constraint solving method **122**. The constraint solving method would take output along with this gesture and all other relevant constraints to produce new output shown in FIG. **4**F.

FIGS. **5**A-**5**C show a schematic pictorial illustration of possible input and output relating to a polyline in accordance with an example of the present system. FIG. **5**A shows a schematic illustration of an input of a polyline. After preprocessing **114**, the gesture determination **116** would not define this input as a gesture and pass the input on for primitive recognition **118**. After primitive recognition **118** determined the input was a polyline, the information would be passed on to the constraint solving method **122**. The constraint solving method **122** would apply the constraints detected by in the primitive polyline: two junctions between three line segments, parallelism between the first and third line segments, the corresponding angle quality between the two angles, and length equality between the first and second line segments. This would produce the polyline output shown in FIG. **5**B as output. A user may further indicate perpendicularity as shown with the additional small box annotation **150** on the second angle shown FIG. **5**B. This annotation **150** would be processed as a gesture, recognized as a gesture to indicate perpendicularity, and passed on to the constraint solving method **122**. The constraint solving method would take output along with this gesture and all other relevant constraints to produce new output shown in FIG. **5**C.

Through the present on-line geometric shape recognition system and method, the best results for user input drawing recognition are provided by breaking down primitives and constraints on a vector component level. By applying constraints to primitives on the vector component level and rebuilding the altered primitives for output, the present on-line geometric shape recognition system and method is

able to handle more primitives and constraints without an exponential increase in difficulty.

While the foregoing has described what is considered to be the best mode and/or other examples, it is understood that various modifications may be made therein and that the subject matter disclosed herein may be implemented in various forms and examples, and that they may be applied in numerous other applications, combinations, and environments, only some of which have been described herein. Those of ordinary skill in that art will recognize that the disclosed aspects may be altered or amended without departing from the true spirit and scope of the subject matter. Therefore, the subject matter is not limited to the specific details, exhibits, and illustrated examples in this description. It is intended to protect any and all modifications and variations that fall within the true scope of the advantageous concepts disclosed herein.

We claim:

1. A non-transitory computer readable medium having a computer readable program code embodied therein, said computer readable program code adapted to be executed to implement a method for providing geometric shape recognition for at least one input stroke, said method comprising:

recognizing at least one shape primitive that corresponds to at least one input stroke;

defining the at least one shape primitive as a plurality of vector components;

applying at least one vector component level constraint to the vector components, the at least one constraint geometrically linking at least two of the vector components; and

providing an output based on the application of the at least one constraint to the vector components.

2. A non-transitory computer readable medium according to claim **1**, wherein the plurality of vector components include components selected from at least one point, length and slope angle.

3. A non-transitory computer readable medium according to claim **2**, wherein the at least one vector component level constraint is defined as applying to at least one point, length, or slope angle.

4. A non-transitory computer readable medium according to claim **3**, wherein the at least one vector component level constraint comprises at least one of length equality, angle equality, parallelism, symmetry, alignment, tangency, connection, and perpendicularity.

5. A non-transitory computer readable medium according to claim **1**, the method further comprising recognizing at least one gesture with respect to the at least one input stroke, wherein the output is further based on application of at least one further constraint associated with the at least one gesture to the vector components.

6. A non-transitory computer readable medium according to claim **5**, wherein the at least one gesture provides graphical editing instructions of the at least one shape primitive.

7. A non-transitory computer readable medium according to claim **1**, wherein if the at least one vector component level constraint comprises multiple constraints, the multiple constraints are applied to the vector components based on a priority order, wherein the priority order comprises constraints maintaining the at least one shape primitive's integrity, constraints implicit in the at least one shape primitive, and constraints defined by a gesture.

8. A non-transitory computer readable medium according to claim **1**, the method further comprising rebuilding the recognized at least one shape primitive from the vector components having the at least one constraint applied thereto to provide the output.

9. A non-transitory computer readable medium according to claim **8**, wherein the at least one input stroke is user drawn and the output is at least one smooth geometric shape.

10. A method for providing geometric shape recognition for at least one input stroke on a computing device, the computing device comprising a processor and at least one non-transitory computer readable medium under control of the processor, said method comprising:

recognizing, by the at least one non-transitory computer readable medium, at least one shape primitive that corresponds to at least one input stroke;

defining, by the at least one non-transitory computer readable medium, the at least one shape primitive as a plurality of vector components;

applying, by the at least one non-transitory computer readable medium, at least one vector component level constraint to the vector components, the at least one constraint geometrically linking at least two of the vector components; and

providing, by the at least one non-transitory computer readable medium, an output based on the application of the at least one constraint to the vector components.

11. A method according to claim **10**, wherein the plurality of vector components include components selected from at least one point, length and slope angle.

12. A method according to claim **11**, wherein the at least one vector component level constraint is defined as applying to at least one point, length, or slope angle.

13. A method according to claim **12**, wherein the at least one vector component level constraint comprises at least one of length equality, angle equality, parallelism, symmetry, alignment, tangency, connection, and perpendicularity.

14. A method according to claim **10**, further comprising recognizing at least one gesture with respect to the at least one input stroke, wherein the output is further based on application of at least one further constraint associated with the at least one gesture to the vector components.

15. A method according to claim **14**, wherein the at least one gesture provides graphical editing instructions of the at least one shape primitive.

16. A method according to claim **10**, wherein if the at least one vector component level constraint comprises multiple constraints, the multiple constraints are applied to the vector components based on a priority order, wherein the priority order comprises constraints maintaining the at least one shape primitive's integrity, constraints implicit in the at least one shape primitive, and constraints defined by a gesture.

17. A method according to claim **10**, further comprising rebuilding the recognized at least one shape primitive from the vector components having the at least one constraint applied thereto to provide the output.

18. A method according to claim **17**, wherein the at least one input stroke is user drawn and the output is at least one smooth geometric shape.

19. A system for providing geometric shape recognition for at least one input stroke on a computing device, the computing device comprising a processor and at least one non-transitory computer readable medium under control of the processor, the system comprising:

causing recognition, by the at least one non-transitory computer readable medium, of at least one shape primitive that corresponds to at least one input stroke;

causing definition, by the at least one non-transitory computer readable medium, of the at least one shape primitive as a plurality of vector components;

causing application, by the at least one non-transitory computer readable medium, of at least one vector component level constraint to the vector components, the at least one constraint geometrically linking at least two of the vector components; and

causing output, by the at least one non-transitory computer readable medium, based on the application of the at least one constraint to the vector components.

20. A system according to claim **19**, wherein the plurality of vector components include components selected from at least one point, length and slope angle.

21. A system according to claim **20**, wherein the at least one vector component level constraint is defined as applying to at least one point, length, or slope angle.

22. A system according to claim **21**, wherein the at least one vector component level constraint comprises at least one of length equality, angle equality, parallelism, symmetry, alignment, tangency, connection, and perpendicularity.

23. A system according to claim **19**, further comprising recognizing at least one gesture with respect to the at least one input stroke, wherein the output is further based on application of at least one further constraint associated with the at least one gesture to the vector components.

24. A system according to claim **23**, wherein the at least one gesture provides graphical editing instructions of the at least one shape primitive.

25. A system according to claim **19**, wherein if the at least one vector component level constraint comprises multiple constraints, the multiple constraints are applied to the to the vector components based on a priority order, wherein the priority order comprises constraints maintaining the at least one shape primitive's integrity, constraints implicit in the at least one shape primitive, and constraints defined by a gesture.

26. A system according to claim **19**, further comprising causing rebuilding of the recognized at least one shape primitive from the vector components having the at least one constraint applied thereto to provide the output.

27. A system according to claim **26**, wherein the at least one input stroke is user drawn and the output is at least one smooth geometric shape.

* * * * *

EXHIBIT M

US010013603B2

(12) **United States Patent**
Onis

(10) **Patent No.: US 10,013,603 B2**
(45) **Date of Patent: Jul. 3, 2018**

(54) **SYSTEM AND METHOD FOR RECOGNIZING MULTIPLE OBJECT STRUCTURE**

(71) Applicant: **MyScript**, Nantes (FR)

(72) Inventor: **Sebastien Onis**, Nantes (FR)

(73) Assignee: **MyScript**, Nantes (FR)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 53 days.

(21) Appl. No.: **15/152,762**

(22) Filed: **May 12, 2016**

(65) **Prior Publication Data**

US 2017/0206406 A1 Jul. 20, 2017

(30) **Foreign Application Priority Data**

Jan. 20, 2016 (EP) 16290015

(51) **Int. Cl.**

G06K 9/00	(2006.01)
G06K 9/72	(2006.01)
G06F 17/24	(2006.01)
G06F 3/0487	(2013.01)

(52) **U.S. Cl.**

CPC ***G06K 9/00422*** (2013.01); ***G06F 3/0487*** (2013.01); ***G06F 17/242*** (2013.01); ***G06K 9/00449*** (2013.01); ***G06K 9/723*** (2013.01); ***G06K 9/726*** (2013.01)

(58) **Field of Classification Search**

None

See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,653,107	A *	3/1987	Shojima	G06K 9/00402 382/189
5,818,963	A *	10/1998	Murdock	G06K 9/00865 382/159
5,970,170	A *	10/1999	Kadashevich	G06K 9/00872 382/179
7,447,360	B2	11/2008	Li et al.	

(Continued)

OTHER PUBLICATIONS

International Search Report issued for PCT/EP2017/000074 dated Apr. 19, 2017.

(Continued)

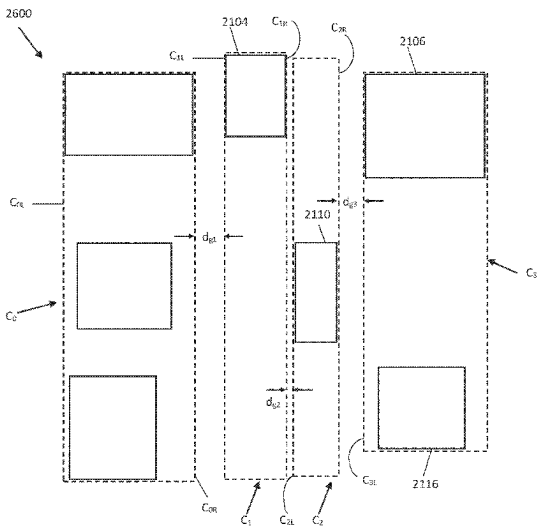
Primary Examiner — Michelle M Entezari

(74) *Attorney, Agent, or Firm* — McDermott Will & Emery LLP

(57) **ABSTRACT**

A system, method, and computer program for recognizing an arrangement of multiple objects on computing devices. The computing device may include a processor and at least one non-transitory computer readable medium for recognizing handwriting input under control of the processor. The non-transitory computer readable medium may be configured to determine with the medium at least one geometrical relationship between a plurality of recognized elements of the input and allocate with the medium the recognized elements having the at least one geometrical relationship therebetween to corresponding positions of an arrangement of the recognized elements. The positions of the arrangement may be cells of a two-dimensional structure of the recognized elements. The at least one geometrical relationship may be an overlap of the plurality of recognized elements in one or more directions of the arrangement.

18 Claims, 22 Drawing Sheets



(56) **References Cited**

U.S. PATENT DOCUMENTS

8,121,412	B2	2/2012	Predovic et al.	
2003/0215145	A1*	11/2003	Shilman	G06F 3/04883 382/195
2007/0003142	A1*	1/2007	Simard	G06K 9/00416 382/187
2009/0123073	A1*	5/2009	Nelson	G06K 9/00416 382/189
2009/0304282	A1*	12/2009	Predovic	G06K 9/00422 382/187
2012/0014601	A1*	1/2012	Jiang	G06F 1/1626 382/173
2012/0114245	A1*	5/2012	Lakshmanan	G06K 9/00409 382/186
2016/0188970	A1*	6/2016	Matsuda	G06K 9/00422 382/187
2017/0109578	A1*	4/2017	Bednarowicz	G06K 9/00476

OTHER PUBLICATIONS

Written Opinion of the International Searching Authority issued for PCT/EP2017/000074 dated Apr. 19, 2017.

Toyozumi et al., "An On-line Handwritten Mathematical Equation Recognition System that Can Process Matrix Expressions by Referring to the Relative Positions of Matrix Elements", Systems and Computers in Japan, vol. 37, No. 14, pp. 87-96, Jan. 1, 2006.

Li et al., "Online Recognition of Handwritten Mathematical Expressions with Support for Matrices", 19th International Conference on Pattern Recognition, 2008: ICPR 2008; 8-11, Tampa, FL, pp. 1-4, Dec. 2008.

Alkalai, "Table Recognition in Mathematical Documents", Jan. 1, 2015, School of Computer Science College of Engineering and Physical Sciences, The University of Birmingham, pp. 1-203, Nov. 2015.

* cited by examiner

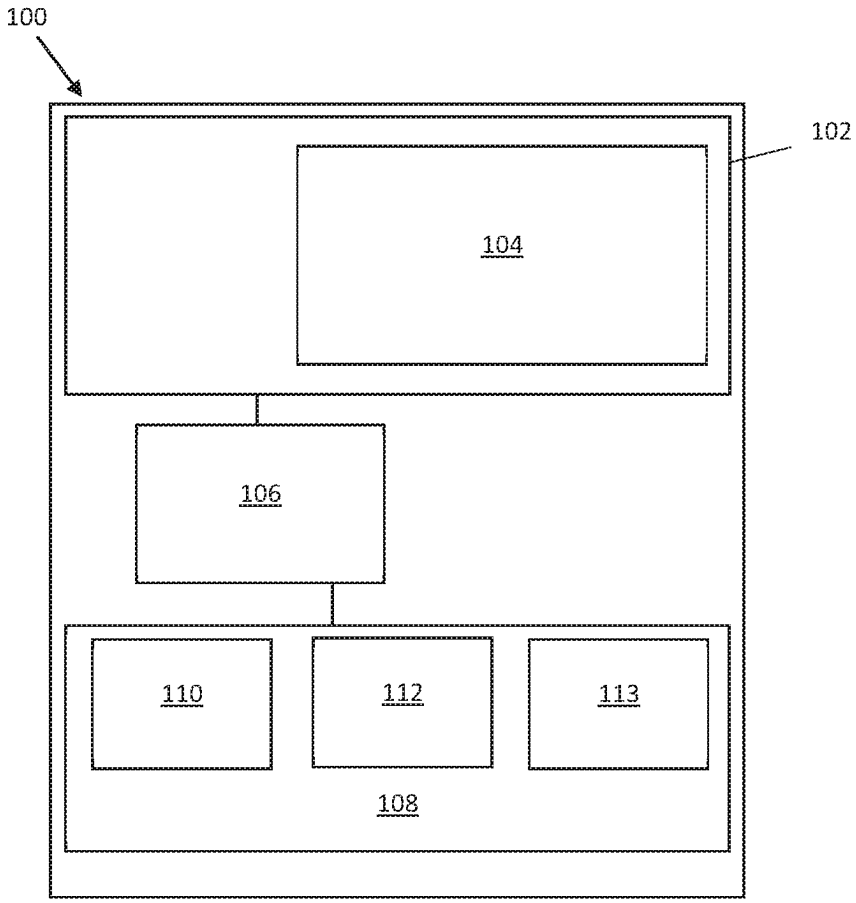


FIG. 1

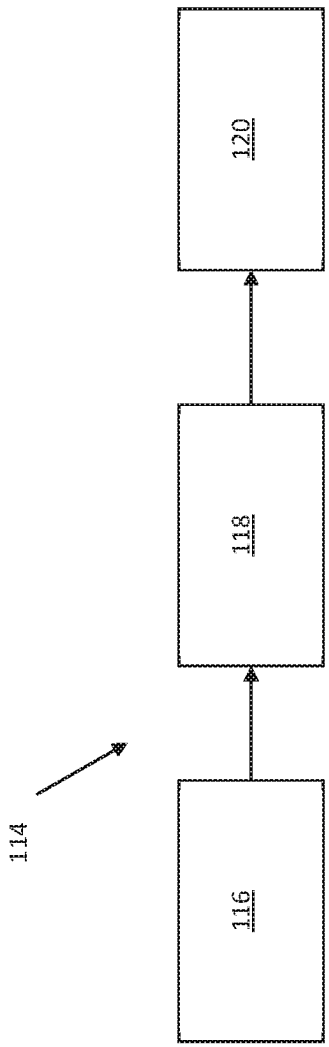


FIG. 2

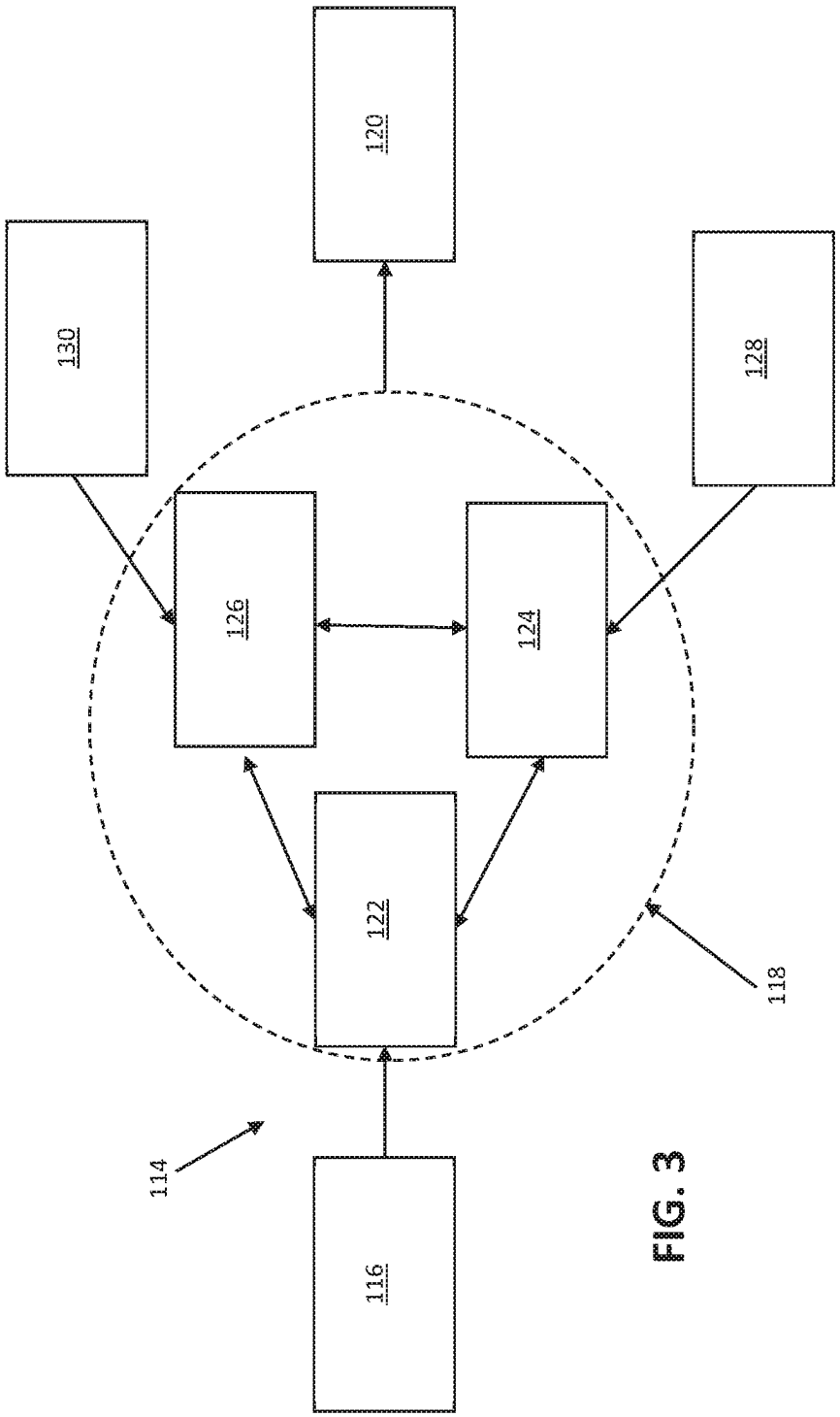


FIG. 3

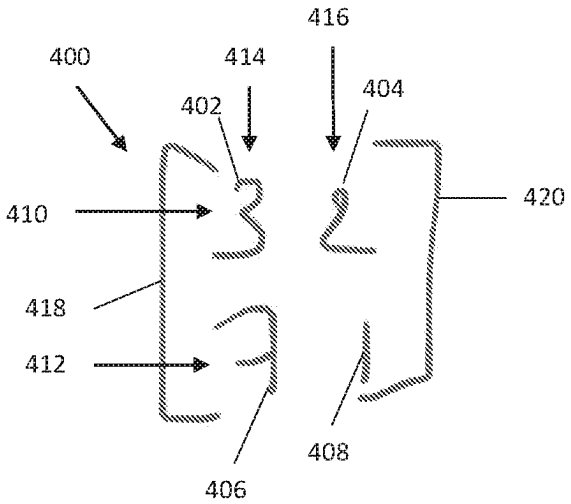

400
414
416
402
404
410
420
418
412
408
406

FIG. 4A

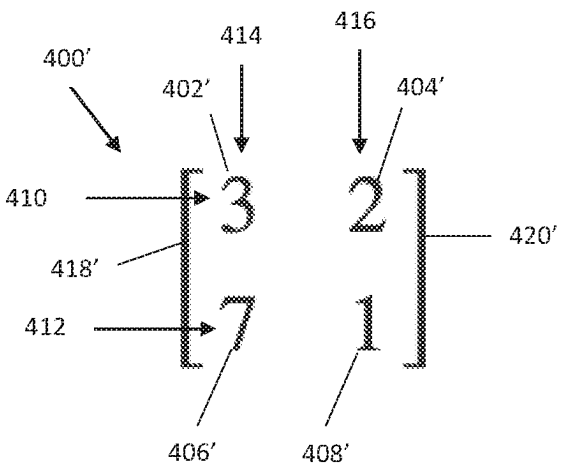

400′
414
416
402′
404′
410
3
2
420′
418′
412
7
1
406′
408′

FIG. 4B

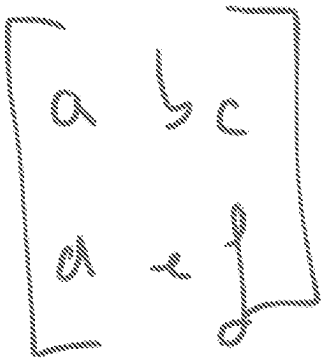


FIG. 5A

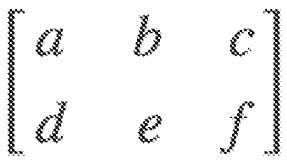


FIG. 5B



FIG. 6

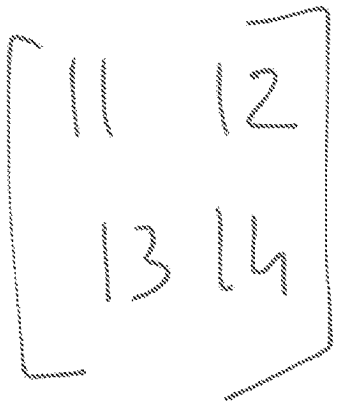


FIG. 7

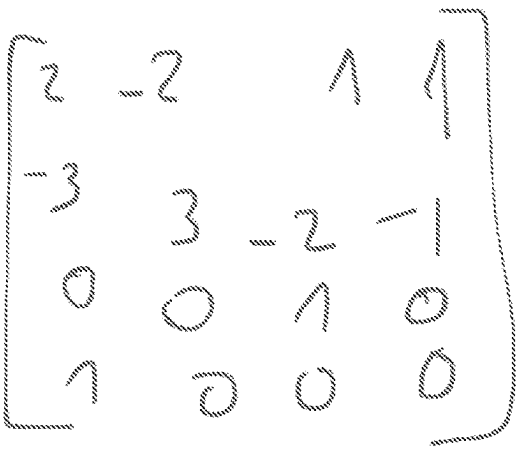


FIG. 8

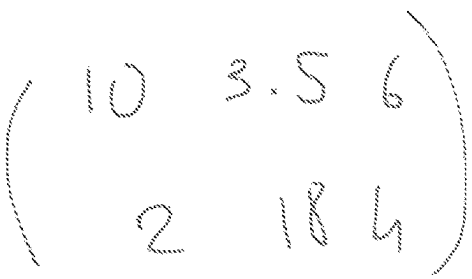


FIG. 9

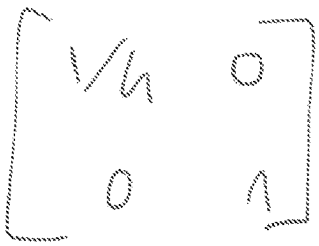


FIG. 10

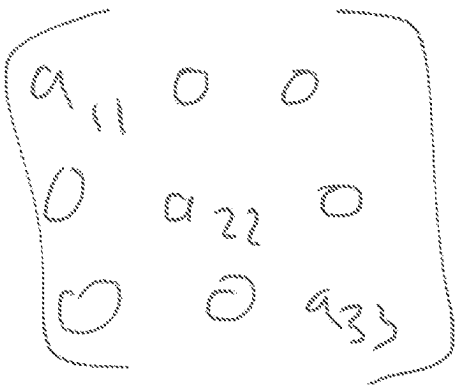


FIG. 11A

$$\begin{pmatrix} a_{11} & 0 & 0 \\ 0 & a_{22} & 0 \\ 0 & 0 & a_{33} \end{pmatrix}$$

FIG. 11B

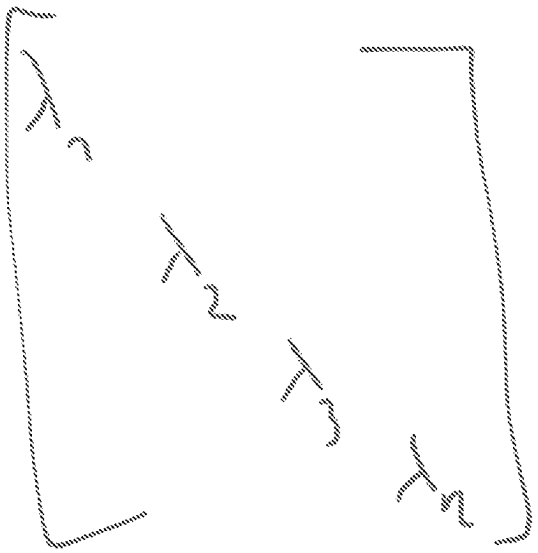


FIG. 12

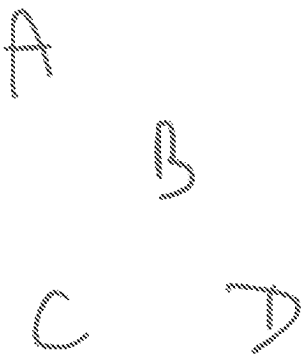


FIG. 13

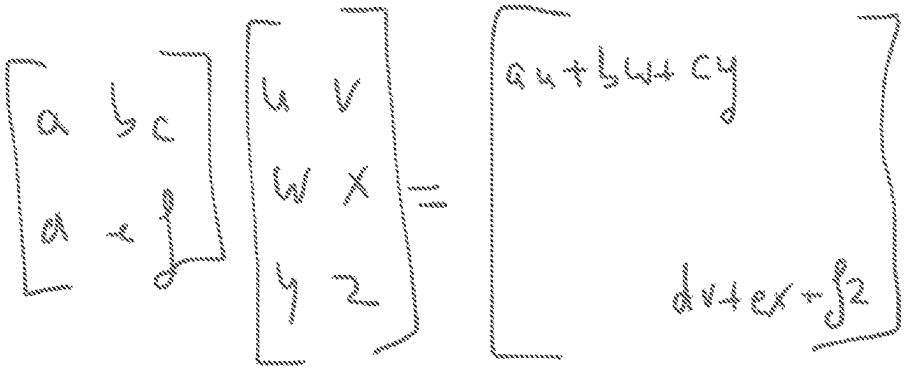


FIG. 14A

$$\begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix} \begin{bmatrix} u & v \\ w & x \\ y & z \end{bmatrix} = \begin{bmatrix} au + bw + cy & \\ & dv + ex + fz \end{bmatrix}$$

FIG. 14B

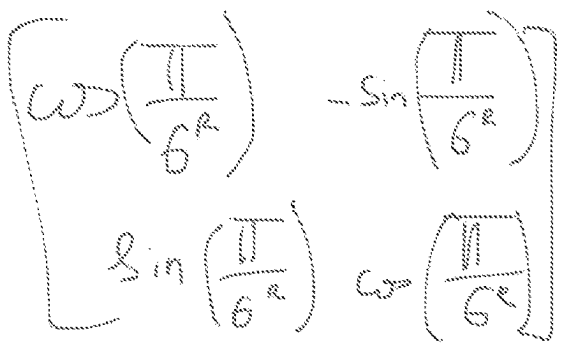


FIG. 15

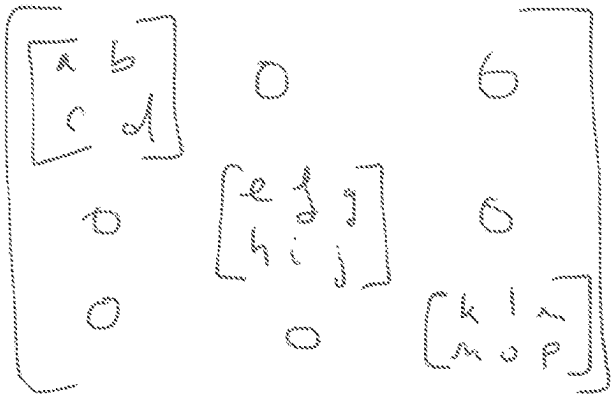


FIG. 16



FIG. 17

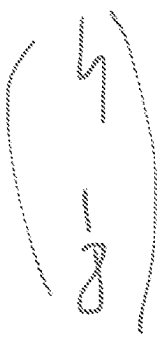


FIG. 18

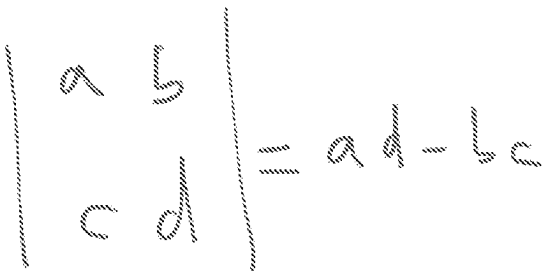


FIG. 19

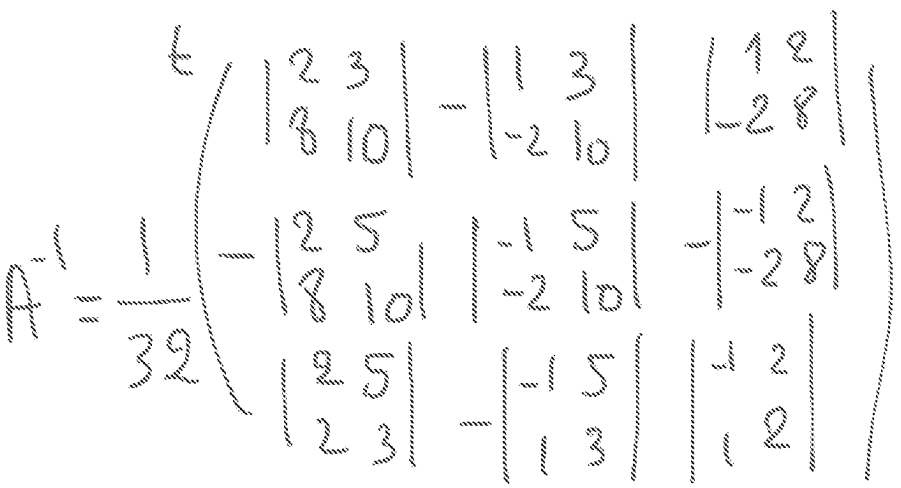


FIG. 20

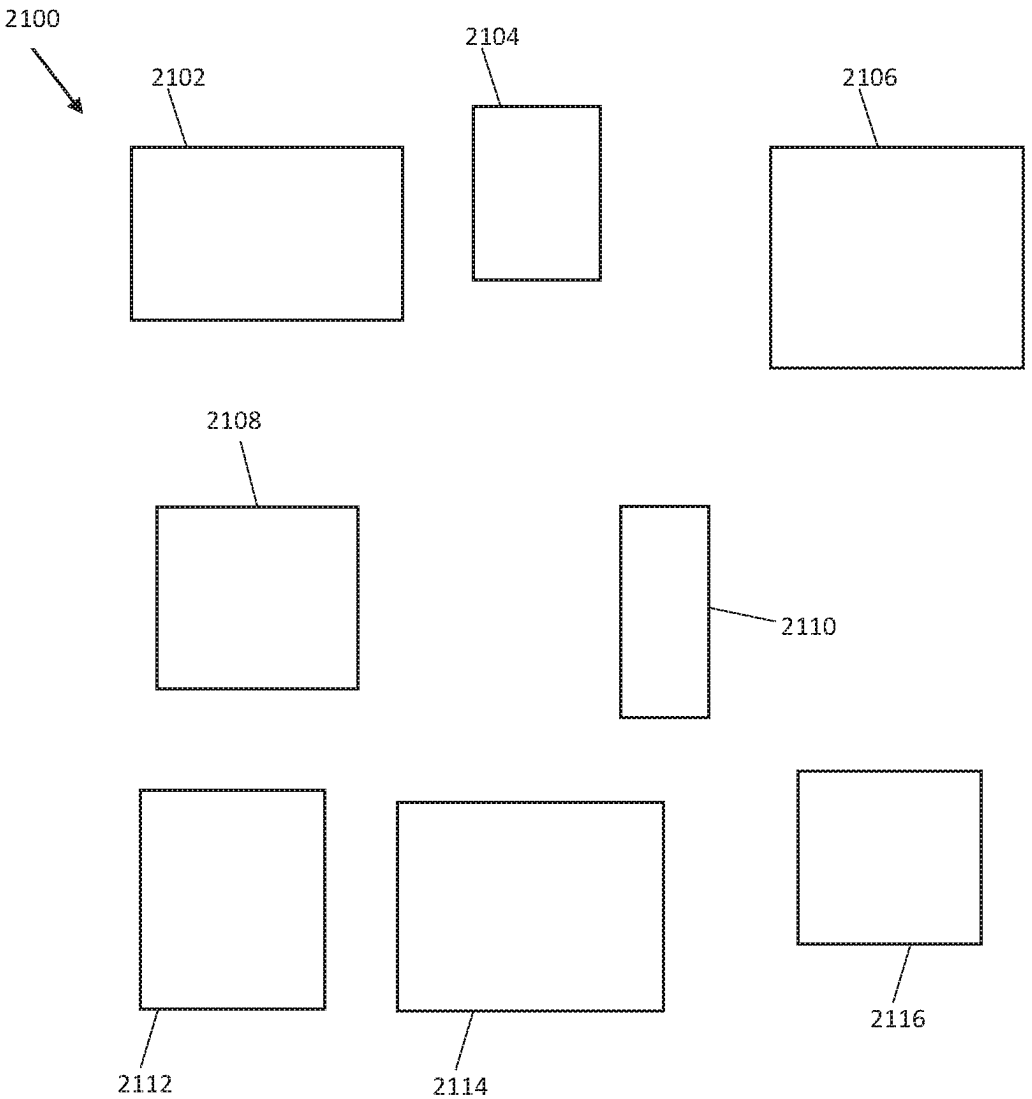


FIG. 21

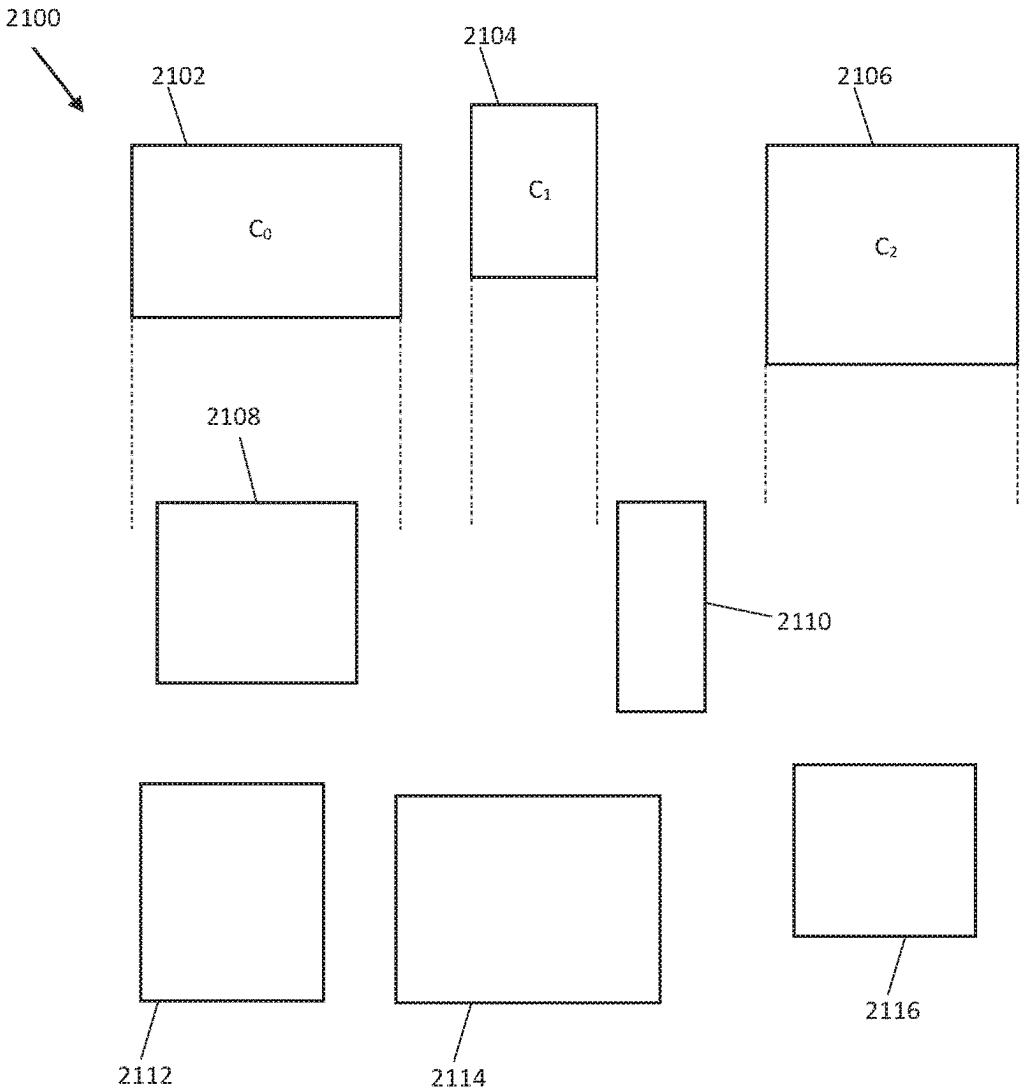


FIG. 22

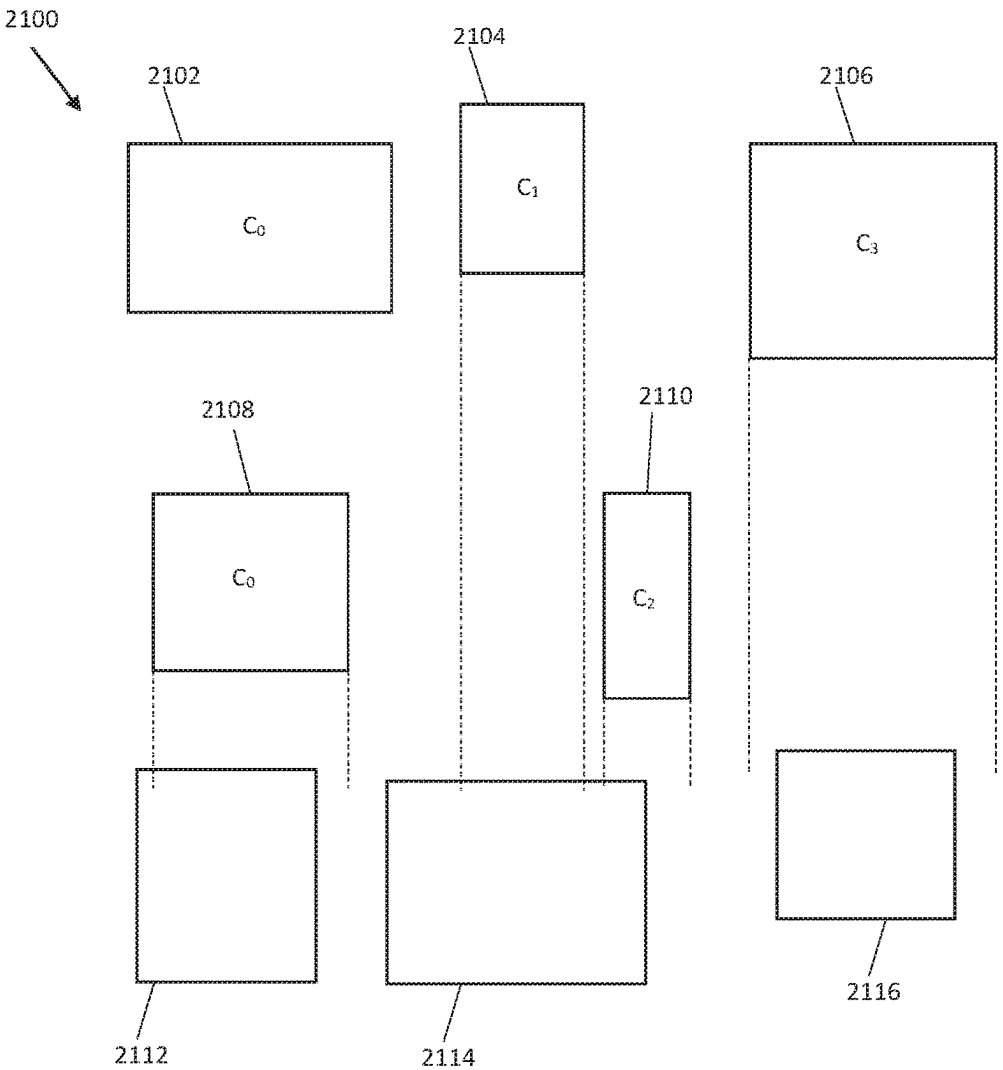


FIG. 23

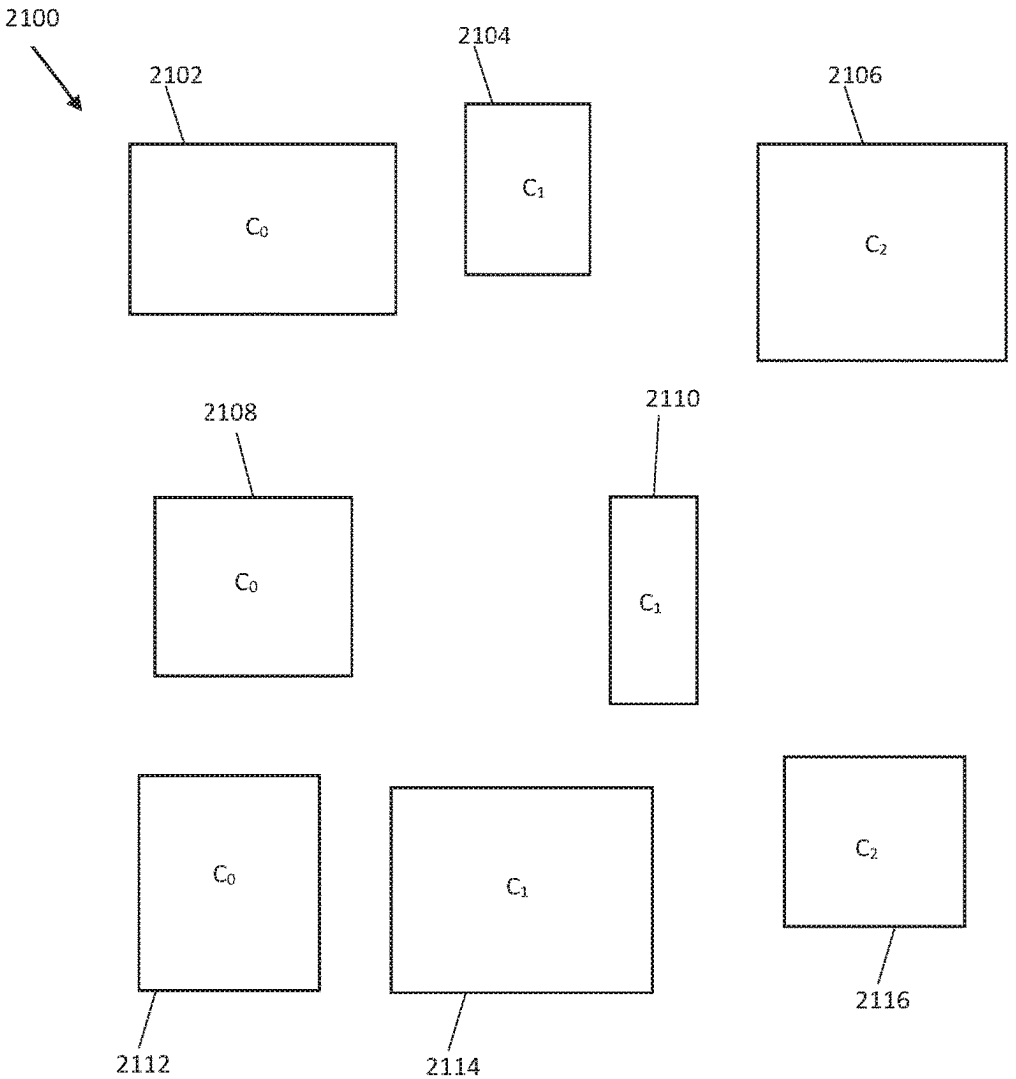

2100
2102
2104
2106
C0
C1
C2
2108
2110
C0
C1
C0
C1
C2
2112
2114
2116

FIG. 24

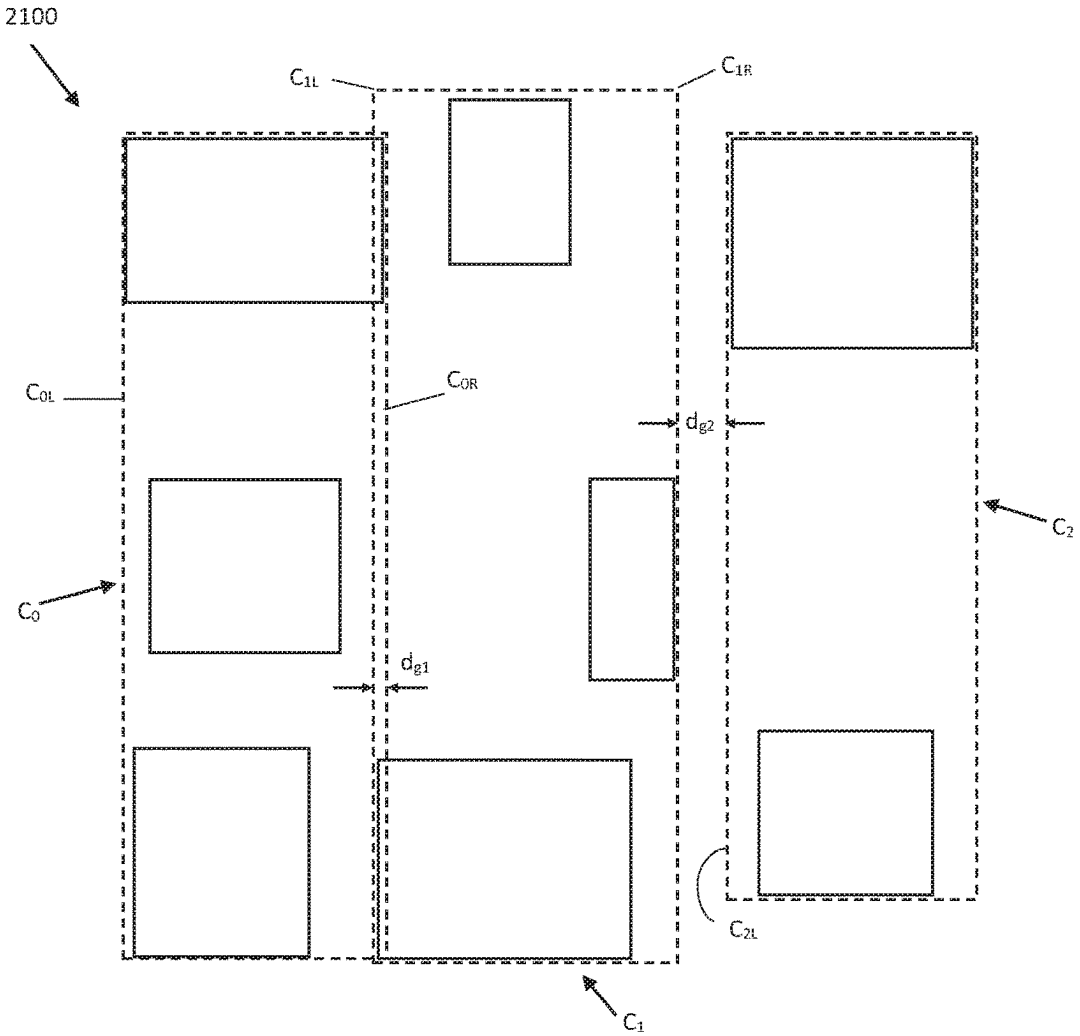


FIG. 25

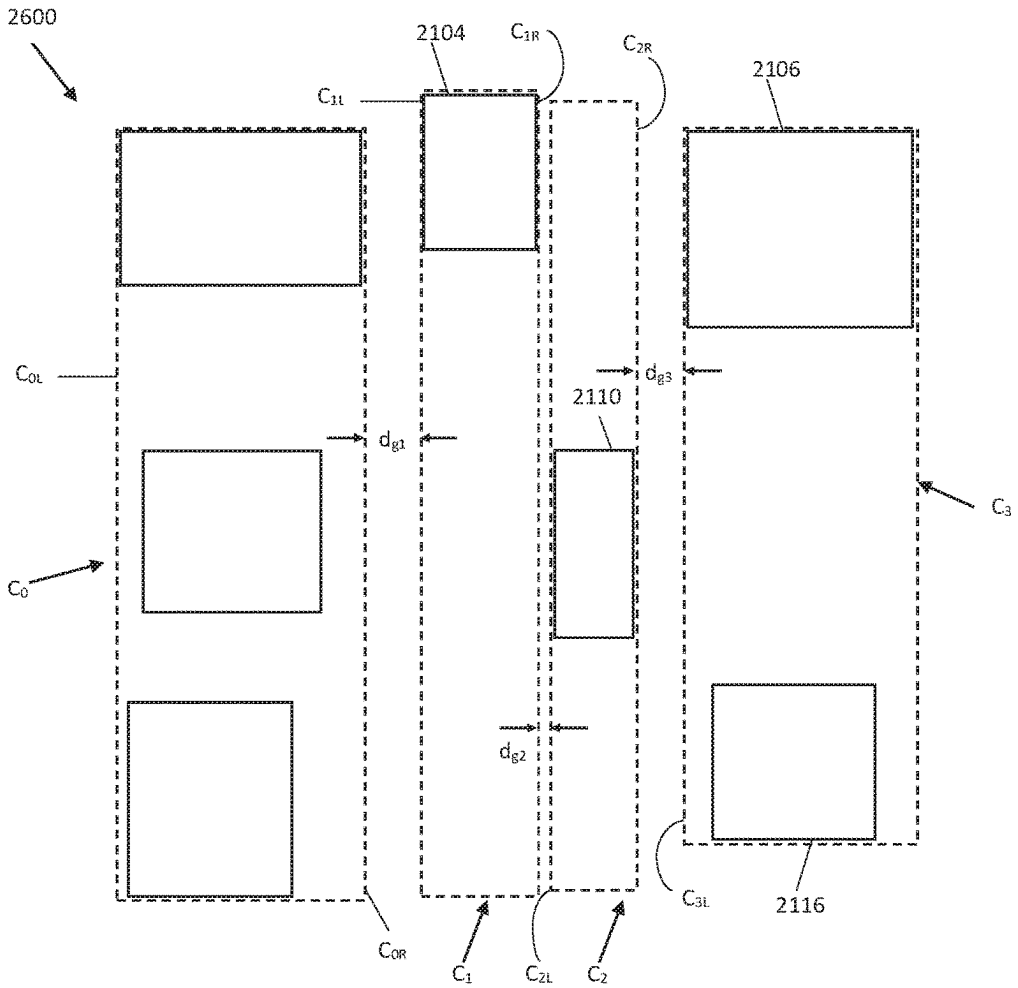

2600
2104
C_{1R}
C_{2R}
2106
C_{1L}
C_{0L}
d_{g3}
2110
d_{g1}
C_3
C_0
d_{g2}
C_{3L}
2116
C_{0R}
C_1
C_{2L}
C_2

FIG. 26

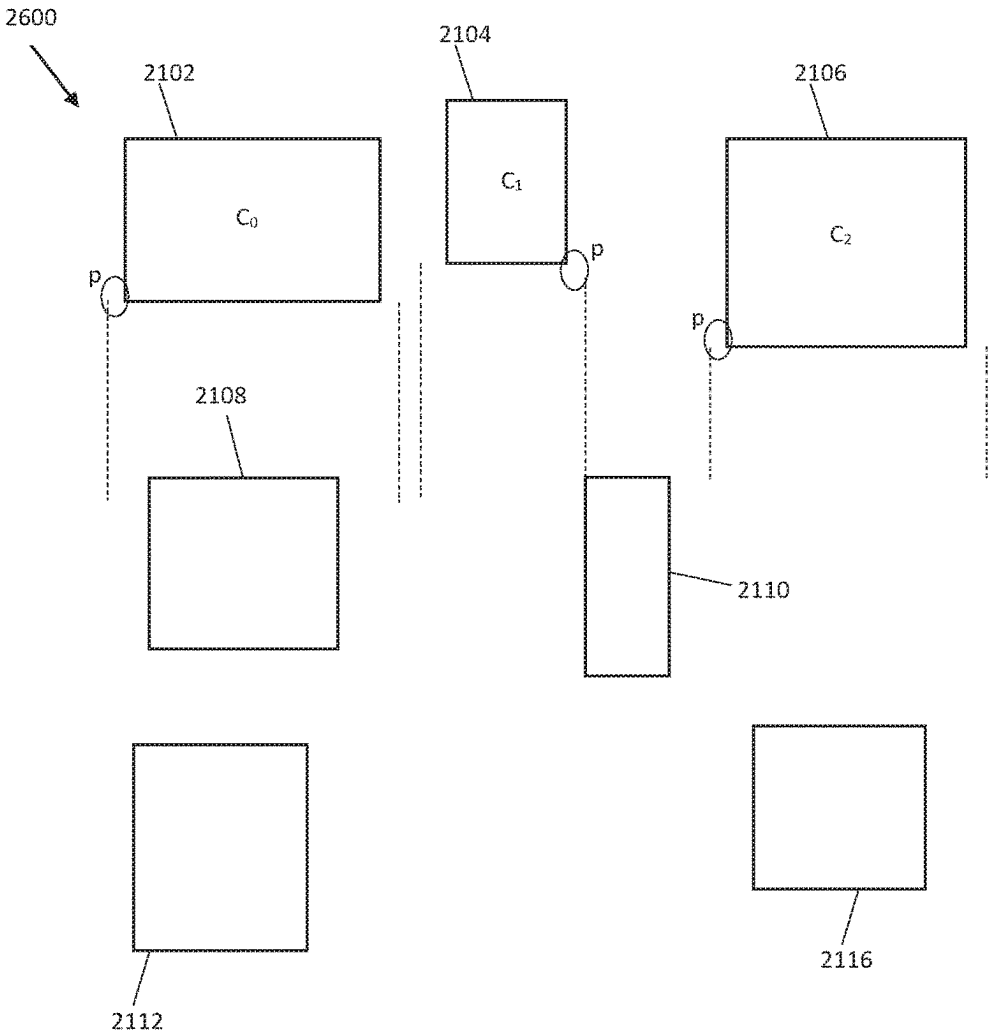


FIG. 27

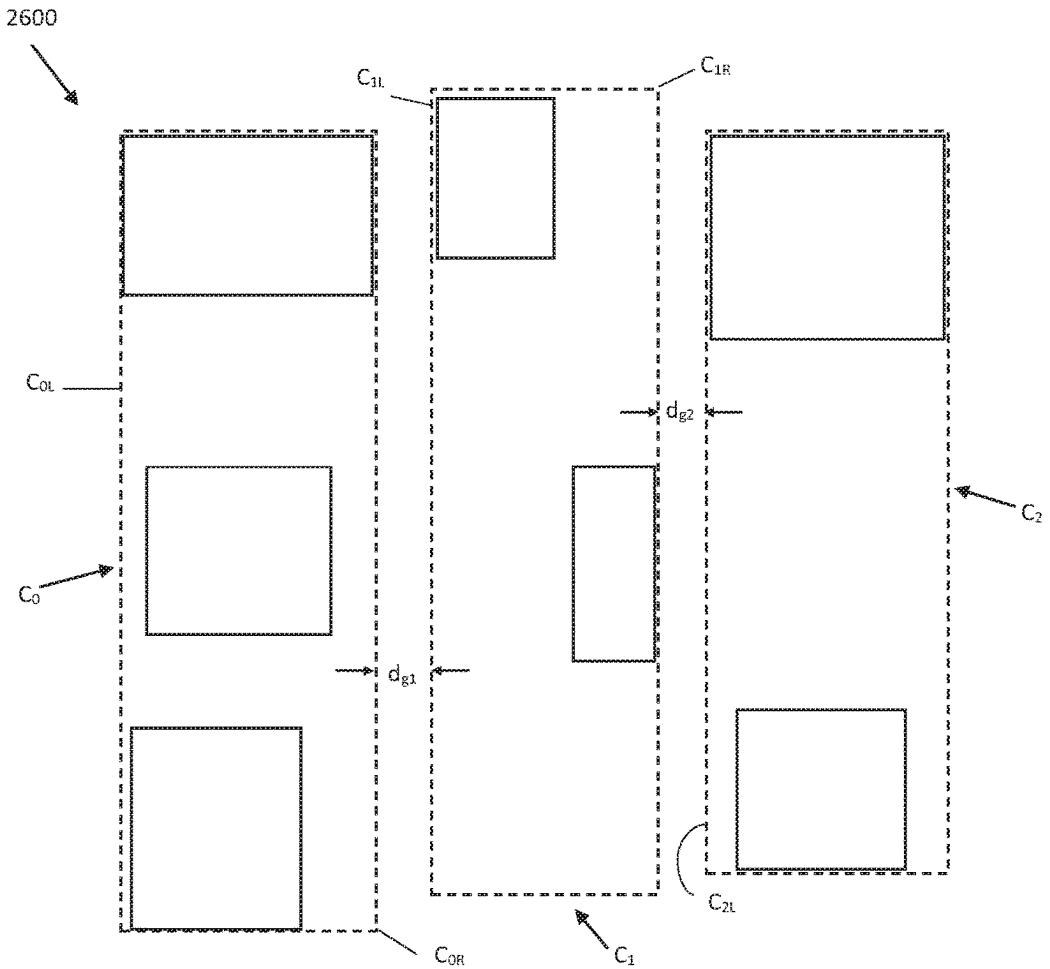


FIG. 28

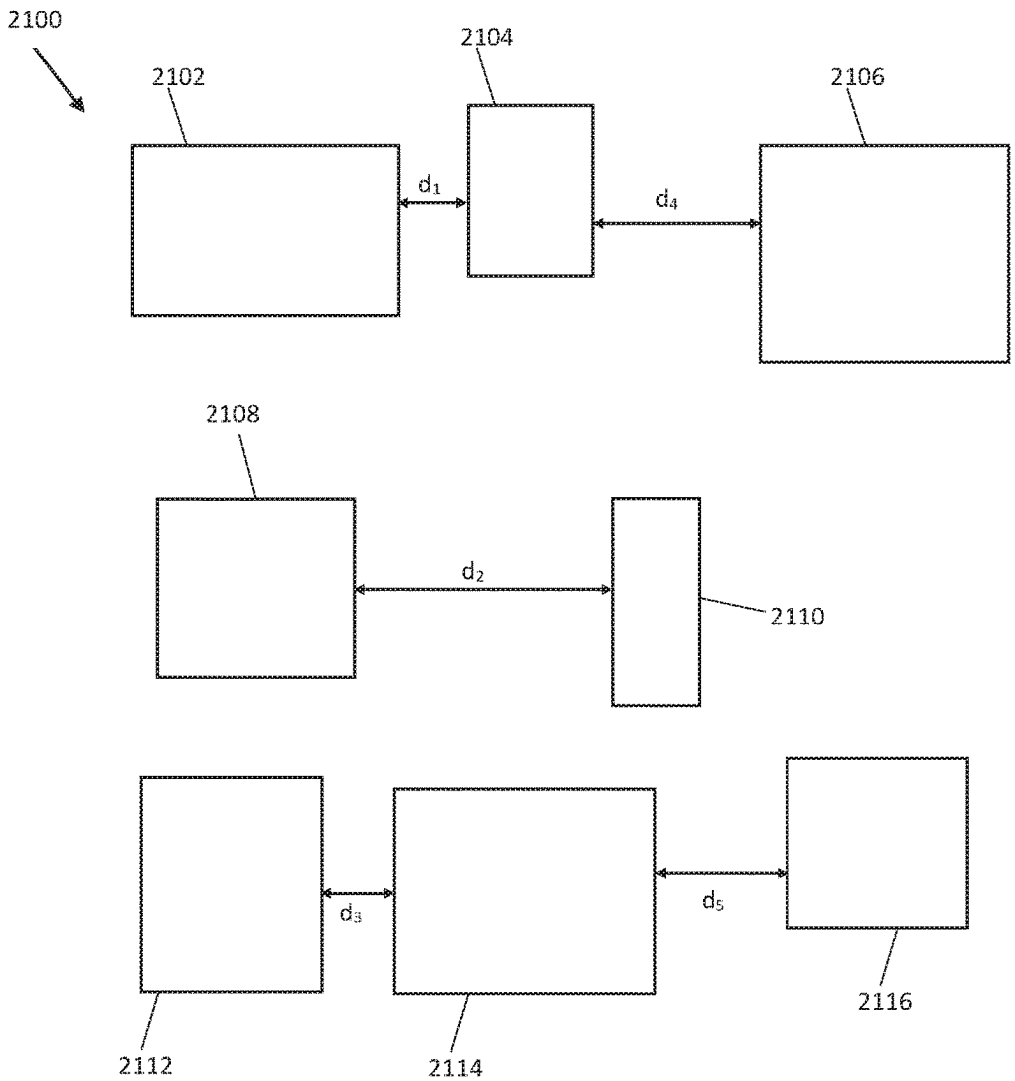


FIG. 29

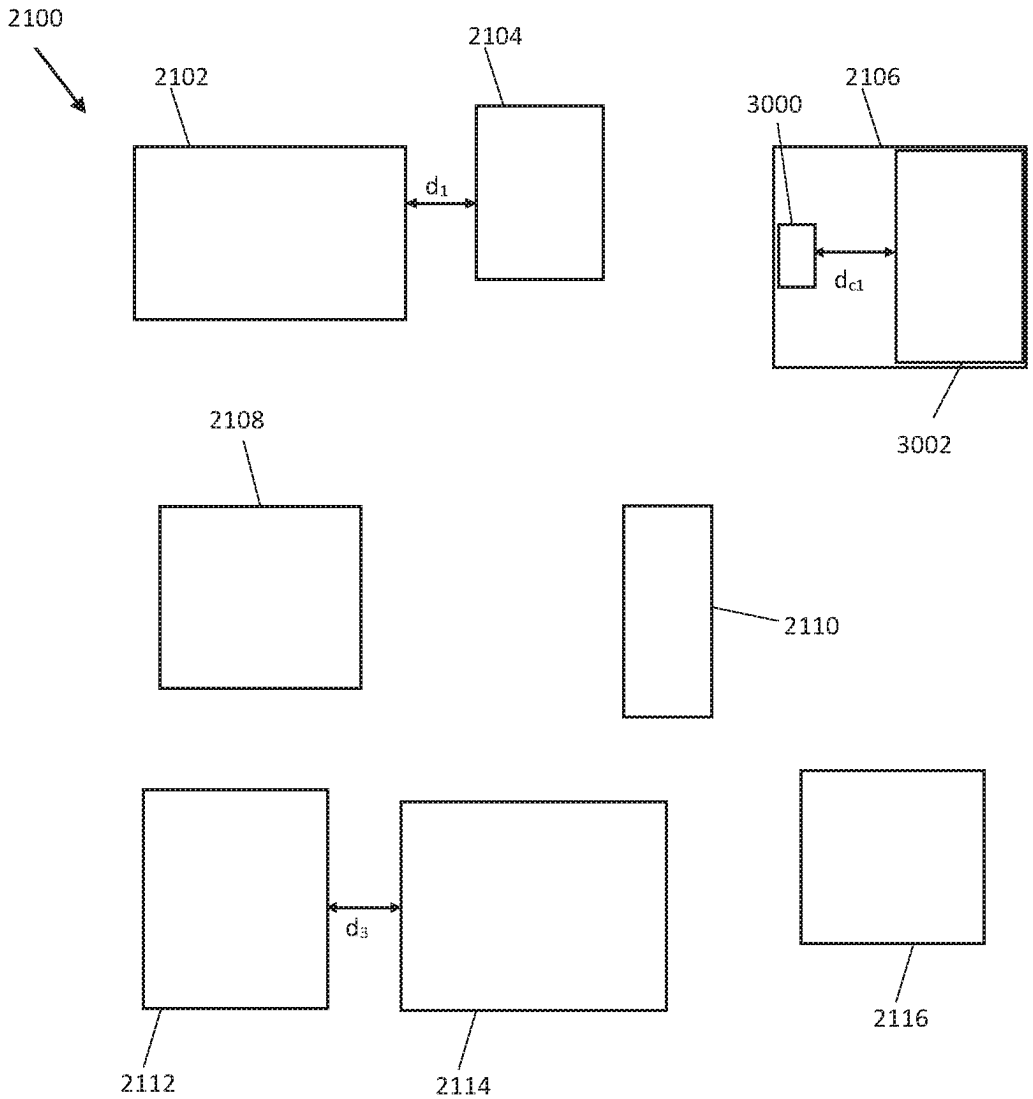

2100
2102
2104
d_1
2106
3000
d_{c1}
3002
2108
2110
d_3
2112
2114
2116

FIG. 30

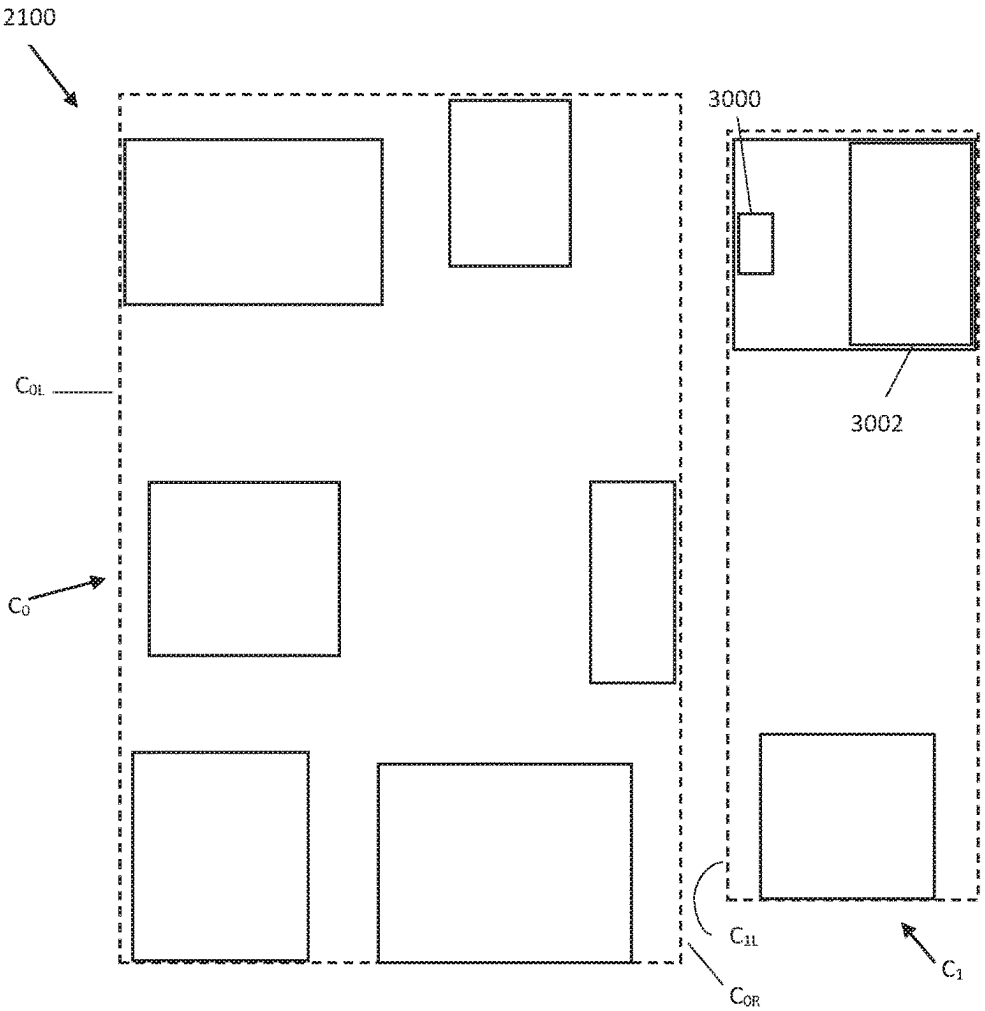


FIG. 31

SYSTEM AND METHOD FOR RECOGNIZING MULTIPLE OBJECT STRUCTURE

CROSS-REFERENCE TO RELATED APPLICATIONS

This application claims priority to European Application No. 16290015.3 filed on Jan. 20, 2016, which is incorporated herein by reference in its entirety and made a part thereof.

TECHNICAL FIELD

The present description relates generally to the field recognizing input of multiple handwritten objects within structures using computing devices. The present description relates more specifically to detecting two-dimensional structures for handwritten content through consideration of positional relationships of the input content and the handwriting recognition of the input content.

BACKGROUND

Computing devices continue to become more ubiquitous to daily life. They take the form of computer desktops, laptop computers, tablet computers, hybrid computers (2-in-1s), e-book readers, mobile phones, smartphones, wearable computers (including smartwatches, smart glasses/headsets), global positioning system (GPS) units, enterprise digital assistants (EDAs), personal digital assistants (PDAs), game consoles, and the like. Further, computing devices are being incorporated into vehicles and equipment, such as cars, trucks, farm equipment, manufacturing equipment, building environment control (e.g., lighting, HVAC), and home and commercial appliances.

Computing devices generally consist of at least one processing element, such as a central processing unit (CPU), some form of memory, and input and output devices. The variety of computing devices and their subsequent uses necessitate a variety of interfaces and input devices. One such input device is a touch sensitive surface such as a touch screen or touch pad wherein user input is received through contact between the user's finger or an instrument such as a pen or stylus and the touch sensitive surface. Another input device is an input surface that senses gestures made by a user above the input surface. A further input device is a position detection system which detects the relative position of either touch or non-touch interactions with a non-touch physical or virtual surface. Any of these methods of input can be used generally for drawing or inputting text. The user's handwriting is interpreted using a handwriting recognition system or method. Other systems for handwriting input to computing devices include electronic or digital pens which interact with paper, encoded surfaces or digitizing surfaces in order to have their movement relative to the surface tracked by a computing device, such as the systems provided by Anoto AB., Leapfrog Enterprises, Inc., and Livescribe, Inc.

Regardless of the input method used, handwriting recognition systems and methods typically involve determining the initiation of a digital ink stroke, such as when first contact with a touch sensitive surface is made (pen-down event); the termination of the stroke, such as when contact with the touch sensitive surface is ceased (pen-up event); and any movement (gestures or strokes) made between stroke initiation and termination. These determined strokes

are processed to recognize and interpret the input. The type of computing device or input surface can also determine the type of handwriting recognition system or method utilized. For instance, if the input surface is large enough (such as a tablet), the user can handwrite input anywhere on or above the input surface, as if the user was writing on paper. This however adds complexity to the recognition task, because the separate elements to be recognized may be related dependent of the relative positions of the elements or may be unrelated independent of their relative positions.

For example, for structured content, such as mathematical equations, tables and matrices, the relative positioning of the handwritten elements are necessary for defining the structure. Some systems are available for dealing with the recognition of mathematical matrices, for example, U.S. Pat. Nos. 7,447,360 and 8,121,412. These systems rely on indicative elements for recognition, such as brackets or spatial alignment, e.g., within rows and columns, and as such merely recognize the structure without regard to the content itself. Whilst such recognition is applicable to relatively simple and well-formed matrices, they are unable to deal with more complex matrices, e.g., containing complex elements like equations, sub-matrices, etc., ill-aligned matrix elements or matrices having empty element cells, e.g., row and column positions. Further, the described systems of these patents provide absolute recognition of these structures thereby influencing recognition of the content itself.

What is required is a system that recognizes matrices and like complex content structures, that do not rely on the input of specific designation elements or gestures and do not significantly increase processing time or complexity to the recognition of themselves whilst retaining sufficient recognition accuracy.

SUMMARY

The examples of the present invention that are described herein below provide systems and methods for providing a system for recognizing an arrangement of multiple objects on computing devices. In one example, the computing device may include a processor and at least one non-transitory computer readable medium for recognizing handwriting input under control of the processor. The non-transitory computer readable medium may be configured to determine with the medium at least one geometrical relationship between a plurality of recognized elements of the input and allocate with the medium the recognized elements having the at least one geometrical relationship therebetween to corresponding positions of an arrangement of the recognized elements. The positions of the arrangement may be cells of a two-dimensional structure of the recognized elements. The at least one geometrical relationship may be an overlap of the plurality of recognized elements in one or more directions of the arrangement.

The at least one non-transitory computer readable medium may also be configured to determine the overlap based on a comparison of a directional distance between one or more geometrical features of the plurality of recognized elements with at least one geometrical threshold. The at least one geometrical threshold may be based on at least one geometrical spacing of at least some of the recognized elements. The at least one geometrical spacing may also be a maximum spacing between components of the at least some of the recognized elements.

In another example, the present invention includes a method for recognizing an arrangement of multiple objects

on computing devices. Each computing device may comprise a processor and at least one non-transitory computer readable medium for recognizing handwriting input under control of the processor. The method may comprise the steps of (i) determining, with the medium, at least one geometrical relationship between a plurality of recognized elements of the input, and (ii) allocating, with the medium, the recognized elements having the at least one geometrical relationship therebetween to corresponding positions of an arrangement of the recognized elements. The positions of the arrangement may be cells of a two-dimensional structure of the recognized elements. The at least one geometrical relationship may be an overlap of the plurality of recognized elements in one or more directions of the arrangement.

The method may further comprise the step of determining the overlap based on a comparison of a directional distance between one or more geometrical features of the plurality of recognized elements with at least one geometrical threshold. The at least one geometrical threshold may be based on at least one geometrical spacing of at least some of the recognized elements. The at least one geometrical spacing may be a maximum spacing between components of the at least some of the recognized elements.

In another example, the present invention includes a non-transitory computer readable medium having a computer readable program code embodied therein. The computer readable program code may be adapted to be executed to implement a method for recognizing an arrangement of multiple objects input to a computing device. The computing device may comprise a processor and at least one system non-transitory computer readable medium for recognizing handwriting input under control of the processor. The method comprises (i) determining, with the medium, at least one geometrical relationship between a plurality of recognized elements of the input, and (ii) allocating, with the medium, the recognized elements having the at least one geometrical relationship therebetween to corresponding positions of an arrangement of the recognized elements. The positions of the arrangement may be cells of a two-dimensional structure of the recognized elements. The at least one geometrical relationship may be an overlap of the plurality of recognized elements in one or more directions of the arrangement.

The method may further comprise the step of determining the overlap based on a comparison of a directional distance between one or more geometrical features of the plurality of recognized elements with at least one geometrical threshold. The at least one geometrical threshold may be based on at least one geometrical spacing of at least some of the recognized elements. The at least one geometrical spacing may be a maximum spacing between components of the at least some of the recognized elements.

BRIEF DESCRIPTION OF THE DRAWINGS

The present system and method will be more fully understood from the following detailed description of the examples thereof, taken together with the drawings. In the drawings like reference numerals depict like elements. In the drawings:

FIG. **1** shows a block diagram of a computing device in accordance with an example of the present system and method;

FIG. **2** shows a block diagram of a system for handwriting recognition in accordance with an example of the present system and method;

FIG. **3** shows a block diagram illustrating detail of the handwriting recognition system of FIG. **2** in accordance with an example of the present system and method;

FIG. **4**A shows an example of a handwritten input in the form of a 2×2 matrix in accordance with the present system;

FIG. **4**B shows an example of the recognized output of the handwritten input of FIG. **4**A.

FIG. **5**A shows an example of a handwritten input in the form of a 2×3 matrix in accordance with the present system;

FIG. **5**B shows an example of the recognized output of the handwritten input of FIG. **5**A.

FIG. **6** shows an example of a handwritten input in the form of a 2×2 matrix in accordance with the present system.

FIG. **7** shows an example of a handwritten input in the form of a 2×2 matrix in accordance with the present system.

FIG. **8** shows an example of a handwritten input in the form of a 4×4 matrix in accordance with the present system.

FIG. **9** shows an example of a handwritten input in the form of a 2×3 matrix in accordance with the present system.

FIG. **10** shows an example of a handwritten input in the form of a 2×2 matrix in accordance with the present system.

FIG. **11**A shows an example of a handwritten input in the form of a 3×3 matrix in accordance with the present system.

FIG. **11**B shows an example of the recognized output of the handwritten input of FIG. **11**A.

FIG. **12** shows an example of a handwritten input in the form of a 3×3 matrix in accordance with the present system.

FIG. **13** shows an example of a handwritten input in accordance with the present system.

FIG. **14**A shows another example of a handwritten input in accordance with the present system.

FIG. **14**B shows an example of the recognized output of the handwritten input of FIG. **14**A.

FIG. **15** shows yet another example of a handwritten input in accordance with the present system.

FIG. **16** shows an example of a handwritten input in the form of a 3×3 matrix with nested matrices in accordance with the present system.

FIG. **17** shows an example of a handwritten input in the form of a 1×3 matrix in accordance with the present system.

FIG. **18** shows an example of a handwritten input in the form of a 3×1 matrix in accordance with the present system.

FIG. **19** shows an example of a handwritten input in accordance with the present system.

FIG. **20** shows an example of a handwritten input in accordance with the present system.

FIGS. **21-31** show an example of matrix recognition in accordance with the present system.

DETAILED DESCRIPTION

In the following detailed description, numerous specific details are set forth by way of examples in order to provide a thorough understanding of the relevant teachings. However, it should be apparent to those of ordinary skill in the art that the present teachings may be practiced without such details. In other instances, well known methods, procedures, components, and/or circuitry have been described at a relatively high-level, without detail, in order to avoid unnecessarily obscuring aspects of the present teachings. Reference to and discussion of directional features such as up, down, above, below, lowest, highest, horizontal, vertical, etc., are made with respect to the Cartesian coordinate system as applied to the input surface on which the input to be recognized is made.

The various technologies described herein generally relate to capture, processing and management of hand-

drawn and handwritten content on portable and non-portable computing devices in a manner which retains the inputted style of the content while allowing conversion to a faithful typeset or beautified version of that content. The systems and methods described herein may utilize recognition of users' natural writing and drawing styles input to a computing device via an input surface, such as a touch sensitive screen, connected to, or of, the computing device or via an input device, such as a digital pen or mouse, connected to the computing device or via a physical or virtual surface monitored by a position detection system. Whilst the various examples are described with respect to recognition of handwriting input using so-called online recognition techniques, it is understood that application is possible to other forms of input for recognition, such as offline recognition in which images rather than digital ink are recognized. The terms hand-drawing and handwriting are used interchangeably herein to define the creation of digital content by users through use of their hands either directly onto a digital or digitally connected medium or via an input tool, such as a hand-held stylus. The term "hand" is used herein to provide concise description of the input techniques, however the use of other parts of a users' body for similar input is included in this definition, such as foot, mouth and eye.

FIG. **1** shows a block diagram of an example computing device **100**. The computing device may be a computer desktop, laptop computer, tablet computer, e-book reader, mobile phone, smartphone, wearable computer, digital watch, interactive whiteboard, global positioning system (GPS) unit, enterprise digital assistant (EDA), personal digital assistant (PDA), game console, or the like. The computing device **100** includes components of at least one processing element, some form of memory and input and/or output (I/O) devices. The components communicate with each other through inputs and outputs, such as connectors, lines, buses, cables, buffers, electromagnetic links, networks, modems, transducers, IR ports, antennas, or others known to those of ordinary skill in the art.

The illustrated example of the computing device **100** has at least one display **102** for outputting data from the computing device such as images, text, and video. The display **102** may use LCD, plasma, LED, iOLED, CRT, or any other appropriate technology that is or is not touch sensitive as known to those of ordinary skill in the art. At least some of the display **102** is co-located with at least one input surface **104**. The input surface **104** may employ technology such as resistive, surface acoustic wave, capacitive, infrared grid, infrared acrylic projection, optical imaging, dispersive signal technology, acoustic pulse recognition, or any other appropriate technology as known to those of ordinary skill in the art to receive user input. The input surface **104** may be bounded by a permanent or video-generated border that clearly identifies its boundaries. Instead of, or additional to, an on-board display, the computing device **100** may have a projected display capability. Alternatively, the computing device may include the input surface independent of, or without, a display. In the case of a device having no display, input made via the input surface is not displayed by the computing device, rather the device acts as an input device only with the recognized input (discussed later) used for control input, for example, or displayed as content on a connected device or dedicated display device.

The computing device **100** may include one or more additional I/O devices (or peripherals) that are communicatively coupled via a local interface. The additional I/O devices may include input devices such as a keyboard, mouse, scanner, microphone, touchpads, bar code readers, laser readers, radio-frequency device readers, or any other appropriate technology known to those of ordinary skill in the art. Further, the I/O devices may include output devices such as a printer, bar code printers, or any other appropriate technology known to those of ordinary skill in the art. Furthermore, the I/O devices may include communications devices that communicate both inputs and outputs such as a modulator/demodulator (modem; for accessing another device, system, or network), a radio frequency (RF) or other transceiver, a telephonic interface, a bridge, a router, or any other appropriate technology known to those of ordinary skill in the art. The local interface may have additional elements to enable communications, such as controllers, buffers (caches), drivers, repeaters, and receivers, which are omitted for simplicity but known to those of skill in the art. Further, the local interface may include address, control, and/or data connections to enable appropriate communications among the other computer components.

The computing device **100** also includes a processor **106**, which is a hardware device for executing software, particularly software stored in the memory **108**. The processor can be any custom made or commercially available general purpose processor, a central processing unit (CPU), commercially available microprocessors including a semiconductor based microprocessor (in the form of a microchip or chipset), microcontroller, digital signal processor (DSP), application specific integrated circuit (ASIC), field programmable gate array (FPGA) or other programmable logic device, discrete gate or transistor logic, discrete hardware components, state machine, or any combination thereof designed for executing software instructions known to those of ordinary skill in the art.

The memory **108** can include any one or a combination of volatile memory elements (e.g., random access memory (RAM, such as DRAM, SRAM, or SDRAM)) and nonvolatile memory elements (e.g., ROM, EPROM, flash PROM, EEPROM, hard drive, magnetic or optical tape, memory registers, CD-ROM, WORM, DVD, redundant array of inexpensive disks (RAID), another direct access storage device (DASD), or any other magnetic, resistive or phase-change nonvolatile memory). Moreover, the memory **108** may incorporate electronic, magnetic, optical, and/or other types of storage media. The memory **108** can have a distributed architecture where various components are situated remote from one another but can also be accessed by the processor **106**. Further, the memory **108** may be remote from the device, such as at a server or cloud-based system, which is remotely accessible by the computing device **100**. The memory **108** is coupled to the processor **106**, so the processor **106** can read information from and write information to the memory **108**. In the alternative, the memory **108** may be integral to the processor **106**. In another example, the processor **106** and the memory **108** may both reside in a single ASIC or other integrated circuit.

The software in memory **108** includes an operating system **110**, an application **112** in the form of a non-transitory computer readable medium having a computer readable program code embodied therein, and a handwriting recognition (HWR) system **114**, which may each include one or more separate computer programs, each of which has an ordered listing of executable instructions for implementing logical functions. The operating system **110** controls the execution of the applications **112** and the HWR system **114**. The operating system **110** may be any proprietary operating system or a commercially or freely available operating system, such as WEBOS, WINDOWS®, MAC and IPHONE OS®, LINUX, and ANDROID. It is understood

that other operating systems may also be utilized. Alternatively, the application **112** of the present system and method may be provided without use of an operating system.

The application **112** may be related to handwriting recognition as described herein, different functions, or both. The application **112** includes programs provided with the computing device **100** upon manufacture and may further include programs uploaded or downloaded into the computing device **100** after manufacture. Some examples include a text editor, telephone dialer, contacts directory, instant messaging facility, computer-aided design (CAD) program, email program, word processing program, web browser, and camera.

The HWR system **114**, with support and compliance capabilities, may be a source program, executable program (object code), script, application, or any other entity having a set of instructions to be performed. When a source program, the program needs to be translated via a compiler, assembler, interpreter, or the like, which may or may not be included within the memory, so as to operate properly in connection with the operating system. Furthermore, the handwriting recognition system with support and compliance capabilities can be written as (a) an object oriented programming language, which has classes of data and methods; (b) a procedure programming language, which has routines, subroutines, and/or functions, for example but not limited to C, C++, Pascal, Basic, Fortran, Cobol, Perl, Java, Objective C, Swift, and Ada; or (c) functional programming languages for example but no limited to Hope, Rex, Common Lisp, Scheme, Clojure, Racket, Erlang, OCaml, Haskell, Prolog, and F#. Alternatively, the HWR system **114** may be a method or system for communication with a handwriting recognition system remote from the device, such as server or cloud-based system, but is remotely accessible by the computing device **100** through communications links using the afore-mentioned communications I/O devices of the computing device **100**. Further, the application **112** and the HWR system **114** may operate together or be combined as a single application. Further still, the application **112** and/or the HWR system **114** may be integrated within the operating system **110**.

Strokes entered on or via the input surface **104** are processed by the processor **106** as digital ink. A user may enter a stroke with a finger or some instrument such as a pen or stylus suitable for use with the input surface. The user may also enter a stroke by making a gesture above the input surface **104** if technology that senses motions in the vicinity of the input surface **104** is being used, or with a peripheral device of the computing device **100**, such as a mouse or joystick. A stroke is characterized by at least the stroke initiation location, the stroke termination location, and the path connecting the stroke initiation and termination locations as captured by the application **112** and/or HWR system **114**. Further information such as timing, pressure, angle at a number of sample points along the path may also be captured to provide deeper detail of the strokes. Because different users may naturally write the same object, e.g., a letter, a shape, or a symbol, with slight variations, the present system accommodates a variety of ways in which each object may be entered whilst being recognized as the correct or intended object.

FIG. **2** is a schematic pictorial of an example of the HWR system **114**. The HWR system **114** includes stages such as preprocessing **116**, recognition **118** and output **120**. The preprocessing stage **116** processes the digital ink to achieve greater accuracy and reducing processing time during the recognition stage **118**. This preprocessing may include normalizing of the path connecting the stroke initiation and termination locations by applying size normalization and/or methods such as B-spline approximation to smooth the input. The preprocessed strokes are then passed to the recognition stage **118** which processes the strokes to recognize the objects formed thereby. The recognized objects are then output **120** to the display **102** generally as a typesetted version of the handwritten elements/characters.

The recognition stage **118** may include different processing elements or experts. FIG. **3** is a schematic pictorial of the example of FIG. **2** showing schematic detail of the recognition stage **118**. Three experts (segmentation expert **122**, recognition expert **124**, and language expert **126**) are illustrated which collaborate through dynamic programming to generate the output **120**.

The segmentation expert **122** defines the different ways to segment the input strokes into individual element hypotheses, e.g., alphanumeric characters and mathematical operators, text characters, individual shapes, or sub expression, in order to form expressions, e.g., mathematical equations, words, or groups of shapes. For example, the segmentation expert **122** may form the element hypotheses by grouping consecutive strokes of the original input to obtain a segmentation graph where each node corresponds to at least one element hypothesis and where adjacency constraints between elements are handled by the node connections.

The recognition expert **124** provides classification of the features extracted by a classifier **128** and outputs a list of element candidates with probabilities or recognition scores for each node of the segmentation graph. Many types of classifiers exist that could be used to address this recognition task, e.g., Support Vector Machines, Hidden Markov Models, or Neural Networks such as Multilayer Perceptrons, Deep, Convolutional or Recurrent Neural Networks. The choice depends on the complexity, accuracy, and speed desired for the task.

The language expert **126** generates linguistic meaning for the different paths in the segmentation graph using language models (e.g., grammar or semantics). The expert **126** checks the candidates suggested by the other experts according to linguistic information **130**. The linguistic information **130** can include a lexicon, regular expressions, etc. and is the storage for all static data used by the language expert **126** to execute a language model. A language model can rely on statistical information on a given language. The linguistic information **130** is computed off-line, with or without adaption according to the results of recognition and user interactions, and provided to the linguistic expert **126**. The language expert **126** aims at finding the best recognition path. In one example, the language expert **126** does this by exploring a language model such as final state automaton (FSA) representing the content of linguistic information **130**. In addition to the lexicon constraint, the language expert **126** may use a language model with statistical information modeling for how frequent a given sequence of elements appears in the specified language or is used by a specific user to evaluate the linguistic likelihood of the interpretation of a given path of the segmentation graph.

The system and method described herein make use of the HWR system **114** in order to recognize handwritten input to the device **100**. In particular, the present system and method recognizes content input within structures, such as matrices. A matrix is defined herein as the layout of several elements which are generally related, typically numbers, characters, symbols, mathematical expressions, in one or more rows and columns without the use of row and/or column designators, like lines and headers which may be present in a table, or dot

points and numbers which may be present in a list, for example. Such matrices are typically used in mathematical operations, and may be input by one or more users of the device **100** in order to capture information and/or carry out such operations, either manually through the process of handwriting or automatically through capabilities of the application **112**, for example, to provide mathematical solutions (discussed later).

Unlike the afore-discussed known systems for recognizing structured input, such as matrices, the present system and method do not require users to input specific content to enable the recognition, such as brackets, nor use processes which fail to recognize matrices having more complex elements, like equations, sub-matrices, ill-aligned elements and missing elements. Further, the known systems use geometrical projection to detect rows and columns and only process entire structures for recognition, whereas the present system and method may use incremental recognition and geometrical relationships of all elements without projection. These and other features of the present system and method are now discussed.

FIGS. **4** to **20** illustrate various examples of handwritten or hand-drawn matrices, with some shown in typeset version also. FIG. **4**A shows handwritten input which has been made onto the input surface **104** of the device **100**, for example, rendered as a 2 by 2 (or 2×2; herein the standard use of 'number of rows'×'number of columns' is used) digital ink matrix **400**. The matrix **400** has four content elements **402**, being "3", **404** being "2", **406** being "7" and **408** being "1". The elements **402** and **404** are located in a first substantially horizontal line or row **410**, the elements **406** and **408** are located in a second substantially horizontal line or row **412**, the elements **402** and **406** are located in a first substantially vertical line or column **414**, and the elements **404** and **408** are located in a second substantially vertical line or column **416**. These content elements are contained within a first substantially vertical bracket or fence element **418** (e.g., on the left hand side of the first column **414**) and a second substantially vertical bracket or fence element **420** (e.g., on the right hand side of the second column **414**). The relative positions of the content elements in one or more rows and one or more columns provides a first characteristic of the matrix **400** based on at least one geometrical relationship of the recognized elements of the input and the containment of the content elements within one or more fence elements provides a second characteristic of the matrix **400**.

The application **112** of the present system and method detects the first characteristic either alone or in combination with the second characteristic in conjunction with the handwriting recognition performed by the HWR system **114** to determine that the input **400** is a matrix, and applies predetermined and/or user settable, e.g., via a user interface (UI) on the device **100**, formatting to output a typeset matrix **400'** as shown in FIG. **4**B. As can be seen, the typeset matrix **400'** has typeset versions **402'** to **408'** of the digital ink content elements **402** to **408**, respectively, still arranged in the rows **410**,**412** and columns **414**,**416**, and typeset versions **418'** and **420'** of the digital ink fence elements **418** and **420**, respectively. Accordingly, the present system and method displays the recognized matrix **400** in a format that is expected by users for a digital version of a matrix, for example. It is understood, that the display format of the recognized matrix made be different, and may not necessarily be in 'typeset ink', but 'beautified' digital ink; that is, the display of digital ink is retained but the natural variations of handwriting are softened or normalized by modifying the relative positions of the elements, smoothing and/or normalizing the ink elements, such as the characters themselves, etc. Whether beautification is carried out in digital ink or typeset ink (e.g., the relative positions of the matrix elements are aligned), or both, users may be provided with feedback on the recognition of the matrix and its content, such as through a preview of the recognition result or elements thereof, such as display of brackets if not already input.

The input of FIG. **4** is a relatively simple example of a matrix with single digit numbers as content elements relatively well-aligned in rows and columns and with relatively clear fences around that content. FIG. **5**A shows a similarly simple 2×3 digital ink matrix rendered from handwritten input but with alphabetic content elements. FIG. **5**B shows the recognized output of the present system and method as a typeset matrix. Like FIG. **4**, the input of FIG. **5** is a relatively simple example of a matrix with single alphabetic characters as content elements relatively well-aligned in rows and columns and with relatively clear fences around that content. FIG. **6** shows a similarly simple 2×2 digital ink matrix rendered from handwritten input with single digit numbers as content elements relatively well-aligned in rows and columns but surrounded or contained by differently shaped fence elements, e.g., round brackets instead of square brackets are used. The present system and method is configured to recognize these different, and other, fence forms in order to assist in the detection of matrix input. It is noted that while a fence element is the secondary characteristic of a matrix, other structures having a primary characteristic of content in rows and/or columns have similar secondary characteristics, such as drawn vertical and/or horizontal lines in tables, for example. Further, recognition of the first characteristic of the input matrix alone, that is without consideration of the second characteristic (e.g., fence elements may not even be present), may be sufficient for such detection, as described in detail later.

A slightly more complex example of a handwritten matrix is shown in FIG. **7**, where the input 2×2 matrix has double digit numbers as content elements with at least one slightly misaligned column, e.g., the left-hand column. This slight misalignment coupled with the spaces between the digits of each double digit element provide a challenge for correct recognition of the matrix being a 2×2 matrix, since one or more of the spaces may be considered as spaces between columns rather than between components of double digits, and as such it is conceivable that without robust recognition rules, the matrix of FIG. **7** could be detected as a 2×5 matrix of single digit elements (e.g., the "1" s of the "11", the "1" and "2" of the "12", the "1" and "3" of the "13" and the "1" and "4" of the "14" are treated separately with the first "1" of the "11" allocated to the first column, the second "1" of the "11" and the "1" of the "13" allocated to the second column, the "3" of the "13" allocated to the third column, the "1" of the "12" and the "1" of the "14" allocated to the fourth column, and the "2" of the "12" and the "4" of the "14" allocated to the fifth column. Similar results of course can occur for larger numbers. If such an incorrect recognition is made, then the consequential beautified output of the recognition would display a recognized matrix which is quite different to that intended by the user and/or any subsequent use of the data re). The present method and system employs criteria to minimize the occurrence of such incorrect matrix recognition caused by over-segmentation of the content elements which returns more columns than applicable to the matrix, which is described in detail later.

Other content element forms also provide a challenge, with or without apparent misalignment. For example, FIG. **8** shows a handwritten matrix having content elements as

single digit numbers where some have designators, e.g., minus signs, which designate a mathematical meaning to those content elements. Such designators may be formed by a wide variety of symbols, marks or characters, and like for the large(r) number example of FIG. **7**, the over-segmentation of the content elements would result in these designators being allocated into columns separate from the content which they designate. Similar issues apply also to numerical content elements involving decimals as shown in the example matrix of FIG. **9**, and fractions as shown in the example matrix of FIG. **10**.

Minimization of incorrect recognition for such matrix input is provided in the present system and method by taking the recognition of the content elements themselves into account in the matrix detection process of the application **112**. That is, for the example matrix of FIG. **8** the HWR system **114** recognizes the designators as minus signs connected to the numbers on the immediate right hand side thereof forming "−1", "−2" and "−3", for example, such that the application **112** treats these as a single content element each, for the example matrix of FIG. **9** the HWR system **114** recognizes the decimal point between adjacent numbers forming "3.5", for example, such that the application **112** treats this as a single content element, and for the example matrix of FIG. **10** the HWR system **114** recognizes the divider line between adjacent numbers forming the fraction "¼", for example, such that the application **112** treats this as a single content element. The manner in which the present system and method takes the recognition results into account for content structure detection is described in detail later.

Other matrix types include a mixture of content element types, such as numbers or alphabetic characters, and still other matrix content element types include abstract mathematical forms, such as subscripts and superscripts. For example, FIG. **11**A shows a handwritten matrix having both numerical content elements, being "0", and alphanumeric content elements having alphabetic characters designated with subscript numerals. Similar to the designator examples of FIGS. **8** to **10** described above, by taking the recognized content into account, e.g., the HWR system **114** recognizes the subscripts connected to the numbers on the immediate left hand side thereof forming "a_{11}", "a_{22}" and "a_{33}", the application **112** provides the recognized matrix shown in FIG. **11**B displayed in typeset ink, since the subscripts are treated as part of a greater content element and not as an entire content element themselves.

FIG. **12** shows a similar example input matrix with content elements including subscripts. However, unlike the earlier examples, the matrix of FIG. **12** also includes empty content element positions or cells. That is, each row and column position is considered as a cell, such that the content element position at the first row (e.g., identified as row 0, say) and the first column (e.g., identified as column 0, say) corresponds to "cell 0,0", for example. Other forms of identifying matrix element positions are possible however. Further, the application **112** and/or HWR system **114** may apply such identifiers to the recognized content as positional data, such as data tags, stored in the memory **108** of the device **100**, for example. The presence of such empty cells can cause another challenge for the proper recognition of two-dimensional structured content, such as matrices, since the identification of the rows and columns of the structure is conceivably only ascertainable through the consideration of elements from other rows and/or columns, not from within the same row and/or column of the element under consideration. For example, an element in an otherwise empty row (or column) is not aligned with any other element and therefore the identification of the row (or column) is made by considering non-aligned elements. The present system and method provides robust matrix detection even in such content scenarios, as described in detail later.

As described earlier, the presence of fence elements can be used to assist the recognition of a matrix, particularly when empty cells are present. This is because, if the presence of fence elements is interpreted to mean that the content surrounded, contained or otherwise adjacent (e.g., in a single fence scenario) the fence element(s) is laid out in one or more rows and one or more columns, the empty cells are considered as part of such row(s) and/or column(s) rather than as non-content. However, as also described earlier, the present system and method is capable of recognizing two-dimensional content structures without the presence or aide of structural indicators, such as fence elements for matrices. For example, FIG. **13** shows an example input matrix with empty cells, like the example of FIG. **12**, but unlike FIG. **12** no fence elements are present. The present system and method still correctly identifies the row and column structure of FIG. **13**, however the recognition result may or may not be identified as a matrix. That is, it may just be identified as two-dimensional structure, which may not have any practical consequential effect depending on any subsequent use of the recognition result. Even in such a case, selection by users to designate the recognized structure as a matrix may be provided, via the UI or the like, and/or through feedback or preview of the recognition result, such as the display of generated digital or typeset ink fence elements about the displayed content elements, for example. Such features of the present system and method are described in detail later.

Still further complexity in two-dimensional structures is possible. FIG. **14**A shows the example 2×3 digital ink matrix of FIG. **5**A with additional input. In particular, a 3×2 handwritten matrix is input to the immediate right of the 2×3 matrix followed by an equals sign, "=", and a further matrix which includes two mathematical equations. As can be seen, the right-most matrix includes the partial result of the multiplication of the first two matrices, that is the product of row 0 of the 2×3 matrix and column 0 of the 3×2 matrix, and the product of row 1 of the 2×3 matrix and column 1 of the 3×2 matrix, where the full result would form an expected 2×2 matrix. FIG. **14**B shows the recognized output in typeset form, and as can be seen the equations of the third matrix are properly recognized as belonging to cell 0,0 and cell 1,1 of a 2×2 matrix. This recognition results despite relatively significant overlap of the handwritten equations, as shown in FIG. **14**A. As described earlier, this recognition is achieved by the present system and method through consideration of the recognition of the equations within the rows of third matrix and may also be recognized through consideration of the whole recognized content, e.g., the expected 2×2 matrix resulting from the product of a 2×3 matrix and a 3×2 matrix. These features are described in detail later.

By treating the content elements as belonging to cells of the two-dimensional structure with regard to the one or more recognized elements of each content element based on the recognition result from the HWR system **114**, rather than treating individual input elements individually to determine cell allocations, the present system and method is able to substantially accurately detect even more complex matrix forms. For example, FIG. **15** shows an example handwritten 2×2 matrix with each content element including relatively complex mathematical functions combining many of the afore-described content features, e.g., alphabetic characters,

words, numbers, superscripts, brackets, symbols, and designators, such as a minus sign and divider lines. However, despite this complexity the present system and method detects the 2×2 matrix since the recognized composite parts of each recognized function is treated as a single content element belonging to a corresponding matrix cell. Further, FIG. **16** shows an example handwritten 3×3 matrix which includes nested matrices, e.g., cell 0,0 includes a 2×2 sub-matrix, cell 1,1 includes a 2×3 sub-matrix and cell 2,2 includes a 2×3 sub-matrix.

While relatively complex matrices present a challenge for detection and recognition, overly simple matrices, such as single row or single column matrices, also present a challenge. FIG. **17** shows an example 1×3 handwritten matrix and FIG. **18** shows an example 3×1 handwritten matrix. In either case since (similar to the empty cell issue discussed earlier) there is no inter-row relationship for the 1×3 matrix or inter-column relationship for the 3×1 matrix it may be difficult to determine the presence of multiple columns (or rows). This is because, only the content elements within that row or column itself are considerable for determining columns or rows, and as such the values of the distances or spaces between those elements in relation to set thresholds or the like for column or row determination are critical to that determination. Basically, less information is available such that determination is more problematic.

For example, in FIG. **17** depending on the threshold set for column determination, the content may be recognized as a 1×3 matrix with content elements "3", "7" and "2", or a 1×2 matrix with content elements "3" and "72", or not a matrix at all with content "372". Likewise, in FIG. **18** depending on the threshold set for row determination, the content may be recognized as a 3×1 matrix with content elements "4", "1" and "8", or a 2×1 matrix with content elements "4" and "18", or not a matrix at all with (vertically disposed) content "418". As discussed earlier, the secondary characteristic of a fence element(s) may assist with matrix detection per se, such that the no matrix result does not occur, but this does not inform the application **112** as to the internal structure of the matrix. In the present system and method, a further (or third or tertiary characteristic) can be considered with respect to certain content types to assist in this determination. For example, the space required between input elements, such as numbers, for columnar identification may be set to be less than for other input elements, such as alphabetic characters, for example. This is because, for numbers it is considered that they should be written reasonably close together when multi-digit numbers are specified. Such a criteria can also assist in more complex matrix detection, like in the example of FIG. **7**. Alternative, or additional, characteristics are also taken into account to properly detect such simple matrices as well as more complex matrices, which are discussed in detail later.

Furthermore, as discussed earlier, other forms of two-dimensional structures are also recognizable by the present system and method. For example, FIG. **19** shows a mathematical function involving a determinant of a matrix, which like a matrix has rows and columns of elemental cells. FIG. **20** shows such determinants nested within a matrix. Further, as can be seen (like some of the earlier examples) additional elements external to the two-dimensional structures may also be present, such as mathematical operators and functions. While such additional elements are recognized separately from the content of a matrix, for example, they may be used as context by the application **112** and/or HWR system **114** in order to guide recognition of the matrix content and the structure of the matrix, such as discussed above, for example.

As can be seen from the afore-described examples, the natural variation in hand-drawn matrices is wide and as such a system which robustly determines structural relationships, like these matrices, must be able to generally perform detection and recognition over this wide range of variations. Of course, some limits apply as to what should be correctly recognized as a matrix, for example, so that the general definition is adhered to. The present system and method therefore uses criteria and the results of recognition for correct detection of matrix structures as described below. It is noted that typically handwriting recognition only takes geometric relationships of recognized elements into account for recognition of these elements themselves, not the structure within which those elements are input. Accordingly, the present system and method uses further information to provide this recognition using a probability based approach which takes into account geometrical and language model features to provide coarse filtering of row and column candidates. These coarsely filtered matrix cell candidates may then be fine filtered using the results of the recognition process to recognize the actual matrix. In this way, the present system and method makes efficient use of recognition to detect matrices by pro-processing matrix element candidates. It is understood that the present system and method is applicable to recognition of other structural relationships of handwritten content in which the content is generally aligned in (horizontal or non-horizontal, e.g., diagonal) rows and (vertical or non-vertical, e.g., diagonal) columns such as tables, lists, etc., since the recognition of the structure is not reliant of specifically input indicators for the structure, but rather the relative geometry of the recognized content itself. These and other features of the present system and method are now discussed.

The following description of the examples illustrated in FIGS. **21** to **37** is directed to matrix recognition, however similar description applies to recognition of other structures as described above. Further, the following description is related to column detection within an input matrix, however as will be apparent the same process may be used to detect rows also, such that the cells of the matrix are detected. In either case, the process generally involves detecting alignments of elements in the substantially horizontal and vertical directions (or directions otherwise based on the direction of writing), using the recognition of the elements belonging to those alignments to provide detection of possible columns or rows of the matrix and recognition of the elements themselves. The process of row detection is performed before column detection in order to provide matrix cell detection, and therefore overall detection of a matrix. In the below described examples, the alignments are detected as part of detecting the candidate matrix columns and rows, and then the candidate columns and rows are analyzed based on the recognized content to provide the recognized matrix output. Alternatively, the detection of either the vertical and/or horizontal alignments may be performed in a different manner separate from the matrix row and/or column hypothesizing, such as by the process of detecting multi-lines, e.g., multi-equations described in United States Patent Application Publication No. 2017/0011262 titled "System for Recognizing Multiple Object Input and Method and Product for Same" filed in the name of the present Applicant and Assignee, the entire content of which is incorporated by reference herein.

FIG. **21** shows an example arrangement **2100** of multiple input ink objects **2102** to **2116**. The ink objects represent handwritten input elements (e.g., one or more strokes forming one or more characters, symbols, etc.) input to the input surface **104** of the device **100**, for example, and rendered as digital ink thereon. The ink objects are generally illustrated as rectangles which correspond to the extent of the handwritten elements, and as such may represent bounding boxes of the elements as recognized by the HWR system **114**. At this point of the matrix detection process the content of the ink objects itself is not necessarily important rather it is the correspondence of the ink objects to recognized elements that is used by the process. It is noted that in the recognition process of the HWR system **114** the recognition result contains a hierarchical determination of recognition candidates for each stroke and groups of the strokes based on the language model employed. The ink objects shown relate to the most probable candidate of each recognized element, however the actual recognized content and therefore the recognized elements may change if different candidates are selected, for example, by users through the UI or the like, such that the arrangement itself changes. In such a case, the matrix detection process is at least partially used again to adapt to the changed arrangement, e.g., at least for that part of the arrangement which is changed.

As can be seen, in the arrangement **2100** the ink objects **2102**, **2104** and **2106** are generally horizontally aligned with one another, the ink objects **2108** and **2110** are generally horizontally aligned with one another, and the ink objects **2112**, **2114** and **2116** are generally horizontally aligned with one another. As such these horizontal alignments are detected by the present system and method in the manner described earlier, and in the illustrated example would most likely be determined as corresponding to potential matrix rows since no overlap between elements of the alignments is present. Further, the ink objects **2102**, **2108** and **2112** are generally vertically aligned with one another, the ink objects **2104**, **2110** and **2114** are generally vertically aligned with one another, and the ink objects **2106** and **2116** are generally vertically aligned with one another. As such these vertical alignments may be detected by the present system and method, and in the illustrated example may be determined as corresponding to potential matrix columns. Accordingly, the arrangement **2100** may be detected as corresponding to a 3×3 matrix, for example. The potential columns of such a matrix are detected as in the following examples of FIGS. **22** to **31**.

As shown in FIG. **22** it is known that the objects **2102**, **2104** and **2106** are substantially horizontally aligned, and from the recognition result of the HWR system **114**, it is known that the objects are (likely) separate content elements. Accordingly, the objects **2102**, **2104** and **2106** are allocated by the application **112** into separate potential first, second and third columns C_0, C_1 and C_2, respectively. The hypothesis of these column allocations is tested by considering whether any ink objects in the next horizontal alignment (in the present example in the downward direction) overlap vertically with the ink objects of the columns. That is, it is determined whether any ink objects overlap with the width of each of the ink objects **2102**, **2104** and **2106** by projecting the extents of these objects into the next horizontal line or row. This basically accords with checking whether the ink objects **2108** and **2110** overlap with the ink objects **2102**, **2104** and **2106** in the vertical direction. In FIG. **22** this is depicted by dashed line projections of the right hand and left hand boundaries of the bounding boxes of the ink objects **2102**, **2104** and **2106**. As can be seen, it is determined that the ink object **2108** overlaps with the ink object **2102**, but the ink object **2110** does not overlap with any of the ink objects **2102**, **2104** and **2106**. Accordingly, in FIG. **23** it is shown that the ink object **2108** is allocated into the first column C_0 with the ink object **2102**, the ink object **2104** remains allocated into the second column C_1, the ink object **2110** is allocated into the third column C_2 (which is adjusted from the allocation depicted in FIG. **22**), and the ink object **2106** is allocated into a fourth column C_3.

Next, the hypothesis of these column allocations is again tested by considering whether any ink objects in the next horizontal alignment overlap vertically with the ink objects of the columns. That is, it is determined whether any ink objects overlap with the width of each of the ink objects **2108**, **2104**, **2110** and **2106** by projecting the extents of these objects into the next horizontal line or row. This basically accords with checking whether the ink objects **2112**, **2114** and **2116** overlap with the ink objects **2102**, **2104** and **2106** in the vertical direction. In FIG. **23** this is depicted by dashed line projections of the right hand and left hand boundaries of the bounding boxes of the ink objects **2108**, **2104**, **2110** and **2106**. As can be seen in FIG. **23**, it is determined that the ink object **2112** overlaps with the ink object **2108**, the ink object **2114** overlaps with both of the ink objects **2104** and **2110**, and the ink object **2116** overlaps with the ink object **2106**. Accordingly, in FIG. **24** it is shown that the ink object **2112** is allocated into the first column C_0 with the ink objects **2102** and **2108**, the ink object **2114** is allocated into the second column C_1 with the ink object **2104**, the ink object **2110** is re-allocated into the second column C_1 as well, and the ink object **2116** is allocated into the third column C_2 with re-allocation of the ink object **2106** from the fourth column depicted in FIG. **23**. This recognition result of likely columns is depicted in FIG. **25** in which the respective boundaries of the three columns C_0, C_1 and C_2 are illustrated as dashed bounding boxes surrounding the ink objects allocated into those columns.

The re-allocation of columns depicted in FIG. **24** basically represents a type of fine filtering of the column detection of the present system and method, in which the detection of columns converges as closer and closer spaced content elements are considered on a row-by-row basis. This filtering may not be performed at this stage however, and all potential columns found in each step may be retained until a subsequent filtering step, which checks whether any columns should be merged. Such processing is described in detail later.

Once all possible hypotheses have been determined and tested, the application **112** may provide (meta)data to the ink objects, such as a tag indicting the row and column allocations, e.g., the cell allocations, recognized for the ink objects, where such data is stored with the ink objects in the memory **108** of the device **100**, for example. In this way, in any subsequent interactions with the ink objects, such as typesetting or editing, the positional relationships between the objects can be taken into account.

The above-described iterative approach of hypothesizing and testing column allocations row-by-row and through consideration of spatially adjacent content elements in those rows basically corresponds to iteratively considering 2×2 arrangements of objects across the entire arrangement in order to find potential columns. That is, the relative positions of two adjacent content elements in a first row are compared to two adjacent content elements in the next (second) row to determine (and adjust) column allocations. This can be performed in a number of ways. In one example, relative distances between features of the content elements are

compared to a threshold to determine if the elements are potentially separated into different columns. That is, a suitable column spacing is searched for between these elements.

For example, in the present case the first column hypothesis tested is that the ink object **2102** is in a first column and the next ink object of that row, i.e., the ink object **2104**, is in a second column, the second column hypothesis tested is that the ink object **2102** is in the first column and the leftmost ink object of the next row, i.e., the ink object **2108**, is in the second column. This can be done by considering whether the horizontal distance between the right hand boundary of the bounding box of the ink object **2102** and the left hand boundary of the bounding boxes of the ink objects **2104** and **2108**, say, is greater than a (first) threshold. The first threshold at its simplest is set to zero, or some non-zero (e.g., positive) value, such that any object which overlaps vertically with the object under consideration returns a negative distance. That is, measured distances like the distance between the objects **2102** and **2014** yield a positive value which is greater than the first threshold and measured distances like the distance between the ink objects **2102** and **2108** yield a negative value which is less than the first threshold. Accordingly, the application **112** correctly determines that it is likely that the ink objects **2102** and **2104** belong to different columns and the ink objects **2102** and **2108** do not belong to different columns.

The intra-row comparison is basically the horizontal line determination described earlier and the inter-row comparison is basically the projection of the higher leftmost object onto the next successive horizontal line described earlier, where this projection essentially forms the fourth element of the 2×2 arrangement being tested. It is understood that the ink objects may be defined by a different characteristic than the bounding box, such as the extent of the strokes themselves, a mean center or barycenter of the strokes making up each element, etc. Further, the distance may be measured in terms of pixels, as a function of some other parameter of the input surface **104**, like a guide for writing, such as an input line pattern or the like as described for example in United States Patent Application Publication No. 2017/0060819 titled "System and Method of Digital Note Taking" filed claiming a priority date of Aug. 25, 2015 in the name of the present Applicant and Assignee, the entire content of which is incorporated by reference herein, in terms of characteristics of the input itself, like average or mean dimensions of the handwritten objects, such as characters, the bounding boxes, etc., or a combination of such features or the like. Further still, it is understood that the iterative approach may be based on time order of entry of the strokes/elements rather than spatial order, as described above, or some combination thereof.

As such, the iterative 2×2 arrangement of objects is considered across the entire arrangement in order to find potential columns. In this way, localized variations in the handwritten elements of the arrangement are locally considered such that they do not influence the overall column detection across the entire arrangement. That is, the Applicant has found that users of digital applications for the handwritten input of two-dimensional structures, such a matrices, tend to skew the spacing of the structural elements as the dimensions of the structure increase, particularly in regular matrices, e.g., in which elements are of the same type. For example, for a simple 2×2 matrix, the relative spacing and sizing of the elements is reasonably consistent, e.g., like in FIG. **4**, whereas, as the matrix grows, certain local compression or extension of element spacing may result, e.g., like in FIG. **8** where the spacing of the elements toward the upper right hand corner is expanded and the spacing of the elements in the lower center is slightly compressed and misaligned horizontally compared to the elements at the upper left hand corner of the matrix, which is the starting position of writing.

The above-described spatial order approach is generally applicable to post-processing of the input, that is, once the user has handwritten all intended input of the matrix processing to recognize the matrix is performed through manual selection of a control element, like a menu button, or input of a gesture, like a multi-point touch on the interface surface such as a double tap, or through automatic processing, like the lapsing of a set amount of time after input, e.g., about half a second to about two seconds. Such a spatial approach is also applicable to processing during input, so-called incremental recognition, by using such triggers. The temporal approach is also applicable to either incremental or post-recognition processing. It is noted that the incremental approach using spatial and/or temporal considerations may decrease the processing time for final recognition of the input, for example, when the user indicates that conversion to typeset is desired or after a set time lapse since last input, since much of the processing has been performed during input.

It can be seen from the afore-described example that by projecting content elements of earlier rows onto later rows to determine likely columns, the presence of empty cells do not influence the detection of columns, and indeed the empty cells are correctly allocated within the columns, e.g., the empty cell between the ink objects **2106** and **2116** is allocated within the third column C_2. This is also the case however, in input situations where there is only a single content element within a column such as in the example matrix of FIG. **12**. For example, FIG. **26** shows an alternative example arrangement **2600**, which is the same as the arrangement **2100** but the ink object **2114** is not present. Accordingly, through the recognition process of the present system and method as described above, it may be determined that four columns C_0, C_1, C_2 and C_3 are present, as shown by dashed bounding boxes surrounding the ink objects allocated into those columns. This is because, unlike in the earlier example of the arrangement **2100**, the ink object **2114** does not overlap with both of the ink objects **2104** and **2110**, and as such these remain allocated to the second and third columns, as determined at the stage of FIG. **23** in the earlier example. Accordingly, in this alternative example each of the objects **2104** and **2110** are allocated as the only objects within their respective columns and the empty cells about these objects are also allocated into those columns.

The afore-described example 'coarse' column detection processes may cause the detection of excess vertical alignments, due to misaligned elements (from non-uniformity of handwriting), non-aligned elements (such as sign designators, e.g., plus and minus), and complex elements (such as sub-matrices, equations), as variously depicted in the examples of FIGS. **4** to **20**. As such, as described earlier the present system and method may also process the coarse detection result with a fine detection process in which it is determined whether some of the detected possible columns should be merged. Examples of such further processing are now described with respect to FIGS. **27** to **31**.

As described earlier, the first threshold may be set to a positive value. Such a setting basically causes a certain amount of padding to be provided about each ink object, thereby increasing the spacing required between columns for detection. Such padding can be used within the initial

'coarse' filtering or in subsequent 'fine' filtering to provide a mechanism to merge columns which are closely spaced. For example, FIG. **27** shows the process at the stage of the earlier example of FIG. **22**, but with a certain amount of padding p (designated by the circles in FIG. **27**) applied to the ink objects **2102**, **2104** and **2106** for the alternative arrangement **2600** of FIG. **26**. In this example, the value of the first threshold is set so that the padding p causes the ink object **2110** to vertically overlap with the ink object **2104**. Accordingly, the column detection process of the present system and method results in the column allocation as shown in FIG. **28**, in which the objects **2104** and **2110** are again allocated to the same column in a three column layout, like in the example of the arrangement **2100**. The value of the first threshold can be arbitrarily set (and resettable by users through the UI for example), or can be automatically defined by the system and method, for example, in relation to characteristics of the handwriting, like average or mean dimensions of the handwritten objects, such as characters.

An alternative or additional mechanism for fine filtering the column allocations is to take into account the relative spacing of elements across the recognized two-dimensional structure. This can be done in several ways. FIG. **29** shows the arrangement **2100** with the horizontal distances between each of the ink objects **2102** to **2116** illustrated. That is, the distance d_1 between the objects **2102** and **2104**, the distance d_2 between the objects **2108** and **2110**, the distance d_3 between the objects **2112** and **2114**, the distance d_4 between the objects **2104** and **2106** and the distance d_5 between the objects **2114** and **2116**. From these distances the mean, average, minimum and maximum distances between the content elements of each column can be determined. For example, the mean distance d_{m1} between the columns C_0 and C_1 is the mean of the distances d_1, d_2 and d_3, and the mean distance d_{m2} between the columns C_1 and C_2 is the mean of the distances d_4 and d_5. Such normalized distances of the coarsely allocated columns can be compared to other parameters of the input arrangement to provide a check of the allocations, which are provided as another (second) threshold for column detection.

An example of such a parameter as the second threshold is the spacing between the detected potential columns. This spacing is shown for example in FIGS. **25**, **26** and **28** variously as the distance d_{g1}, d_{g2} and d_{g3} which are measured between the right and left hand boundaries of the column bounding boxes, indicated in subscript as "$_{nL}$" and "$_{nR}$" for each of the columns C_0 to C_3, where "n" is the number of the column, e.g., 0 to 3. For example, the second threshold set on this parameter for merging columns may be the maximum positive spacing, such as d_{g2} in FIG. **25**, scaled down by a pre-determined constant value, such as about 0.1 to about 0.7, and typically about 0.3, for example, to provide for natural variations in handwriting, which is used to compare with the mean distance of each column, e.g., the mean distances d_{m1} and d_{m2}. Based on this comparison, if the mean distance is less than this second threshold value then the columns are merged. In this way, detected columns that are reasonably much less spaced than the largest spaced column, which is considered to have a high likelihood of being a column, are merged as they likely do not constitute columns.

Another example of such a parameter as the second threshold is the spacing within certain content elements. Such as the spacing between content features like alphabetic characters, words, numbers, superscripts, subscripts, brackets, symbols, and designators, such as minus and plus signs and divider lines. For example, FIG. **30** shows the ink object **2106** constituted by two elements **3000** and **3002** separated by a distance d_{c1}. For example, the element **3000** may be a designator, such as a minus sign, and the element **3002** may be a number. The present system and method may treat such adjacent elements as recognized by the HWR system **114** as single content elements based on rules set in the application **112** or the HWR system **114** itself may return ink objects in this way, for example, as described earlier with respect to FIGS. **8**, **9** and **10**. For example, the second threshold set on this parameter for merging columns may be the maximum spacing between such elements scaled up by a pre-determined constant value, such as about 1.1 to about 1.7, and typically about 1.4, for example, to provide for natural variations in handwriting, and compare this value with the mean distance of each column. Based on this comparison, if the mean distance is less than this second threshold value then the columns are merged, as shown in FIG. **31** in which the first and second columns of FIG. **25** are merged because the distance d_{g1} is less than the scaled distance d_{c1}, for example, so that two re-allocated columns C_0 and C_1 remain. In this way, detected columns that are reasonably spaced less than the largest spaced connected elements, like numbers with minus and plus signs, are merged as they likely do not constitute columns since it is expected that columns are spaced more widely than the inter-element spacing.

The use of the mean, or other common value, spacing of the ink objects in each coarsely detected column in the above-described merge operations based on global and maximal features of the recognized structure provide a relatively robust evaluation of any possible over-segmentation of the structure due to local variations within the structure.

As described earlier, the above-described examples of column detection can also apply to row detection, in which the vertical alignments are known from the HWR system **114** and used by the application **112** to determine the rows. Further, in the above-described examples of coarse and fine two-dimensional structure detection, the detection result can be provided as one or more probability scores for each detected column and row, which are calculated for example in terms of geometric cost for that probability result. That is, the described first and second thresholds are parameters against which the geometrical relationships of the recognized content of the handwritten input is compared to determine the geometrical cost of that recognized content being in a particular structure, such as rows and columns. This cost can be supplemented to the actual recognition results from the HWR system **114** and/or the structure detection result of the application **112** as weightings, for example, to the probability produced through the handwriting recognition results. In this way, the influence of certain recognized content on the subsequent allocation to structural elements can be easily made, such as allowing closer spacing for numerals as described earlier, merging of columns and/or rows, increasing the probability of a matrix based on the similarity of content, e.g., all numbers, characters, equations, increasing the probability score for a matrix if brackets or fence elements are detected in proximity to content elements.

The various examples described herein can be applied to forms of input for recognition other than handwriting, such as offline recognition in which images rather than digital ink are recognized, for example, the elements may be input as an image captured as a photograph of writing on paper or a whiteboard, digitally captured on an interactive smartboard, etc.

The described methods and systems increase processing and recognition speed of multiple objects in two-dimensional structures, such as a matrix of numbers, characters, mathematical functions or equations, and combinations thereof, as structure recognition is performed in consideration of the recognition of the objects themselves. Furthermore, writing of structures, such as matrices, does not require specific user action for recognition, such as creating a new writing area, tapping a matrix button, drawing brackets, etc. Further, no learning or training of the algorithm is required, however this could be performed to improve results.

While the foregoing has described what is considered to be the best mode and/or other examples, it is understood that various modifications may be made therein and that the subject matter disclosed herein may be implemented in various forms and examples, and that they may be applied in numerous other applications, combinations, and environments, only some of which have been described herein. Those of ordinary skill in that art will recognize that the disclosed aspects may be altered or amended without departing from the true spirit and scope of the subject matter. Therefore, the subject matter is not limited to the specific details, exhibits, and illustrated examples in this description. It is intended to protect any and all modifications and variations that fall within the true scope of the advantageous concepts disclosed herein.

I claim:

1. A system for recognizing an arrangement of multiple objects on computing devices, each computing device comprising a processor, a memory and at least one non-transitory computer readable medium for recognizing handwriting input under control of the processor, the at least one non-transitory computer readable medium configured to:

determine, with the medium, at least one geometrical relationship between a plurality of recognized elements of the handwriting input; and

allocate, with the medium, the recognized elements having the at least one geometrical relationship therebetween to corresponding positions of an arrangement of the recognized elements at least partially determined by boundaries of the recognized elements.

2. A system as claimed in claim **1**, wherein the positions of the arrangement are cells of a two-dimensional structure of the recognized elements.

3. A system as claimed in claim **1**, wherein the at least one geometrical relationship is an overlap of the plurality of recognized elements in one or more directions of the arrangement.

4. A system as claimed in claim **3**, the at least one non-transitory computer readable medium configured to determine the overlap based on a comparison of a directional distance between one or more geometrical features of the plurality of recognized elements with at least one geometrical threshold.

5. A system as claimed in claim **4**, wherein the at least one geometrical threshold is based on at least one geometrical spacing of at least some of the recognized elements.

6. A system as claimed in claim **5**, wherein the at least one geometrical spacing is a maximum spacing between components of the at least some of the recognized elements.

7. A method for recognizing an arrangement of multiple objects on computing devices, each computing device comprising a processor and at least one non-transitory computer readable medium for recognizing handwriting input under control of the processor, the method comprising:

determining, with the medium, at least one geometrical relationship between a plurality of recognized elements of the handwriting input; and

allocating, with the medium, the recognized elements having the at least one geometrical relationship therebetween to corresponding positions of an arrangement of the recognized elements at least partially determined by boundaries of the recognized elements.

8. A method as claimed in claim **7**, wherein the positions of the arrangement are cells of a two-dimensional structure of the recognized elements.

9. A method as claimed in claim **7**, wherein the at least one geometrical relationship is an overlap of the plurality of recognized elements in one or more directions of the arrangement.

10. A method as claimed in claim **9**, comprising determining the overlap based on a comparison of a directional distance between one or more geometrical features of the plurality of recognized elements with at least one geometrical threshold.

11. A method as claimed in claim **10**, wherein the at least one geometrical threshold is based on at least one geometrical spacing of at least some of the recognized elements.

12. A method as claimed in claim **11**, wherein the at least one geometrical spacing is a maximum spacing between components of the at least some of the recognized elements.

13. A non-transitory computer readable medium having a computer readable program code embodied therein, said computer readable program code adapted to be executed to implement a method for recognizing an arrangement of multiple objects input to a computing device, the computing device comprising a processor and at least one system non-transitory computer readable medium for recognizing handwriting input under control of the processor, the method comprising:

determining, with the medium, at least one geometrical relationship between a plurality of recognized elements of the handwriting input; and

allocating, with the medium, the recognized elements having the at least one geometrical relationship therebetween to corresponding positions of an arrangement of the recognized elements at least partially determined by boundaries of the recognized elements.

14. A non-transitory computer readable medium as claimed in claim **13**, wherein the positions of the arrangement are cells of a two-dimensional structure of the recognized elements.

15. A non-transitory computer readable medium as claimed in claim **13**, wherein the at least one geometrical relationship is an overlap of the plurality of recognized elements in one or more directions of the arrangement.

16. A non-transitory computer readable medium as claimed in claim **15**, comprising determining the overlap based on a comparison of a directional distance between one or more geometrical features of the plurality of recognized elements with at least one geometrical threshold.

17. A non-transitory computer readable medium as claimed in claim **16**, wherein the at least one geometrical threshold is based on at least one geometrical spacing of at least some of the recognized elements.

18. A non-transitory computer readable medium as claimed in claim **17**, wherein the at least one geometrical spacing is a maximum spacing between components of the at least some of the recognized elements.

* * * * *

EXHIBIT N

US009489572B2

(12) **United States Patent**
Mélinand et al.

(10) **Patent No.: US 9,489,572 B2**
(45) **Date of Patent: Nov. 8, 2016**

(54) **SYSTEM AND METHOD FOR RECOGNIZING GEOMETRIC SHAPES**

(71) Applicant: **Vision Objects**, Nantes (FR)

(72) Inventors: **Robin Mélinand**, Carquefou (FR); **Florent Buisson**, Nantes (FR)

(73) Assignee: **MYSCRIPT**, Nantes (FR)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **14/558,181**

(22) Filed: **Dec. 2, 2014**

(65) **Prior Publication Data**

US 2016/0154998 A1 Jun. 2, 2016

(51) **Int. Cl.**

G06K 9/00	(2006.01)
G06F 3/0488	(2013.01)
G06T 11/60	(2006.01)
G06T 11/20	(2006.01)

(52) **U.S. Cl.**

CPC ***G06K 9/00422*** (2013.01); ***G06F 3/04883*** (2013.01); ***G06K 9/00402*** (2013.01); ***G06T 11/203*** (2013.01); ***G06T 11/60*** (2013.01)

(58) **Field of Classification Search**

CPC .. G06K 9/00422; G06K 9/222; G06T 11/60; G06T 11/203; G06F 3/04883; G06F 3/033; F02B 2075/025; G09B 5/065

USPC 382/189, 107, 120, 123, 188, 203, 214, 382/253, 290, 309, 321; 345/418, 419, 420, 345/440, 441; 700/90, 95, 117, 159, 180, 700/182; 716/50, 51, 52, 122; 359/556, 359/554, 555

See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,544,265	A	8/1996	Bozinovic et al.	
5,596,656	A	1/1997	Goldberg	
5,809,267	A	9/1998	Moran et al.	
5,880,743	A *	3/1999	Moran	G06T 13/80 345/473
6,233,351	B1	5/2001	Feeney et al.	
6,292,190	B1	9/2001	Corn	

(Continued)

FOREIGN PATENT DOCUMENTS

EP	1355275	10/2003
EP	1030263 B1	5/2004

(Continued)

OTHER PUBLICATIONS

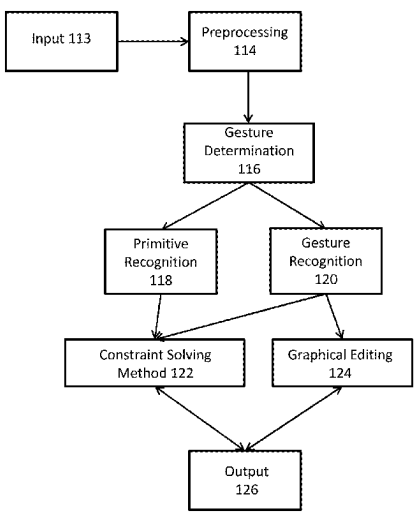
Nritten Opinion and International Search Report issued for PCT/EP2015/002400 dated Mar. 24, 2016.

Primary Examiner — Sheela C Chawan
(74) *Attorney, Agent, or Firm* — McDermott Will & Emery LLP

(57) **ABSTRACT**

A system and method that is able to recognize a user's natural drawing of geometric shapes. The system and method is able to process single-stroke and multi-stroke geometric shapes. It can also apply implicit and user defined constraints to the geometric shapes. The system and method applies these constraints at the vector component level rather than the primitive level. It does this by breaking down both the geometric shapes and constraints on a vector component level. This allows the system and method to handle a larger number of geometric shapes and constraints. After applying the constraints to the geometric shapes at the vector component level, the system and method outputs smooth geometric shapes that incorporated both the implicit and user defined constraints.

37 Claims, 12 Drawing Sheets



(56) **References Cited**

U.S. PATENT DOCUMENTS

6,411,732	B1	6/2002	Saund	
6,459,442	B1	10/2002	Edwards et al.	
6,476,834	B1	11/2002	Doval et al.	
7,256,773	B2 *	8/2007	Kolmykov-Zotov	G06F 3/04883 178/18.01
7,324,691	B2	1/2008	Li et al.	
7,352,902	B2	4/2008	Li et al.	
7,385,595	B2	6/2008	Bryborn et al.	
7,639,250	B2	12/2009	Xu et al.	
8,102,383	B2	1/2012	Cohen et al.	
8,159,501	B2	4/2012	Rao	
8,564,620	B2	10/2013	Bellamy et al.	
8,670,623	B2	3/2014	Oota	
8,718,375	B2	5/2014	Ouyang et al.	
2005/0275622	A1	12/2005	Patel et al.	
2008/0198146	A1	8/2008	Bryborn et al.	
2009/0278848	A1	11/2009	Robertson et al.	
2010/0100866	A1	4/2010	Kamper et al.	
2010/0149109	A1	6/2010	Elias	
2010/0251188	A1	9/2010	Fu et al.	
2011/0185272	A1	7/2011	Rajkumar	
2011/0279455	A1	11/2011	McDaniel	
2012/0176416	A1	7/2012	Dondurur et al.	
2012/0221938	A1	8/2012	Patterson et al.	

FOREIGN PATENT DOCUMENTS

EP	1519300	B1	7/2009
EP	1454225	B1	1/2013
WO	2012115853	A2	8/2012
WO	2014046302	A1	3/2014

* cited by examiner

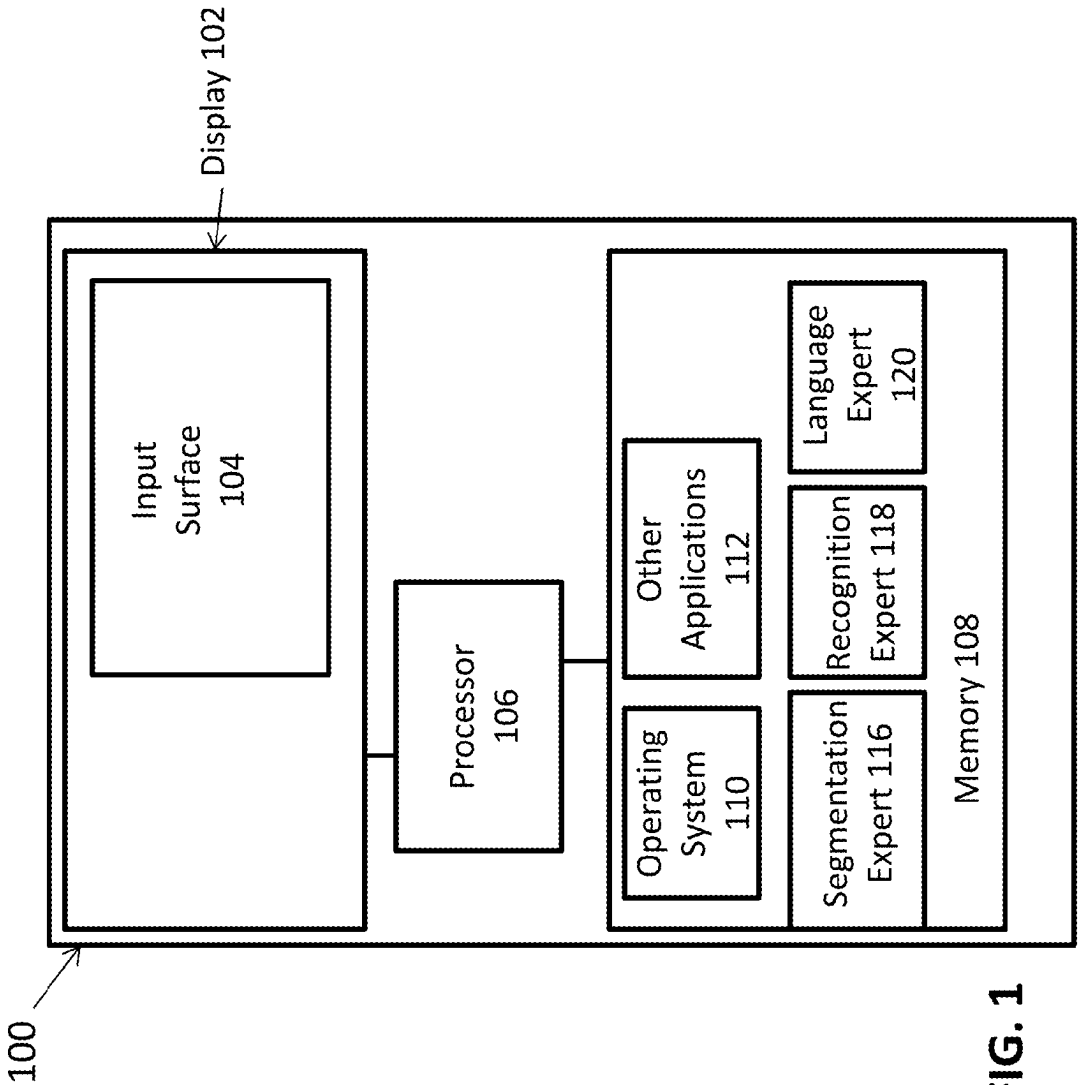

Display 102
Input Surface 104
Processor 106
Other Applications 112
Operating System 110
Language Expert 120
Recognition Expert 118
Segmentation Expert 116
Memory 108
100

FIG. 1

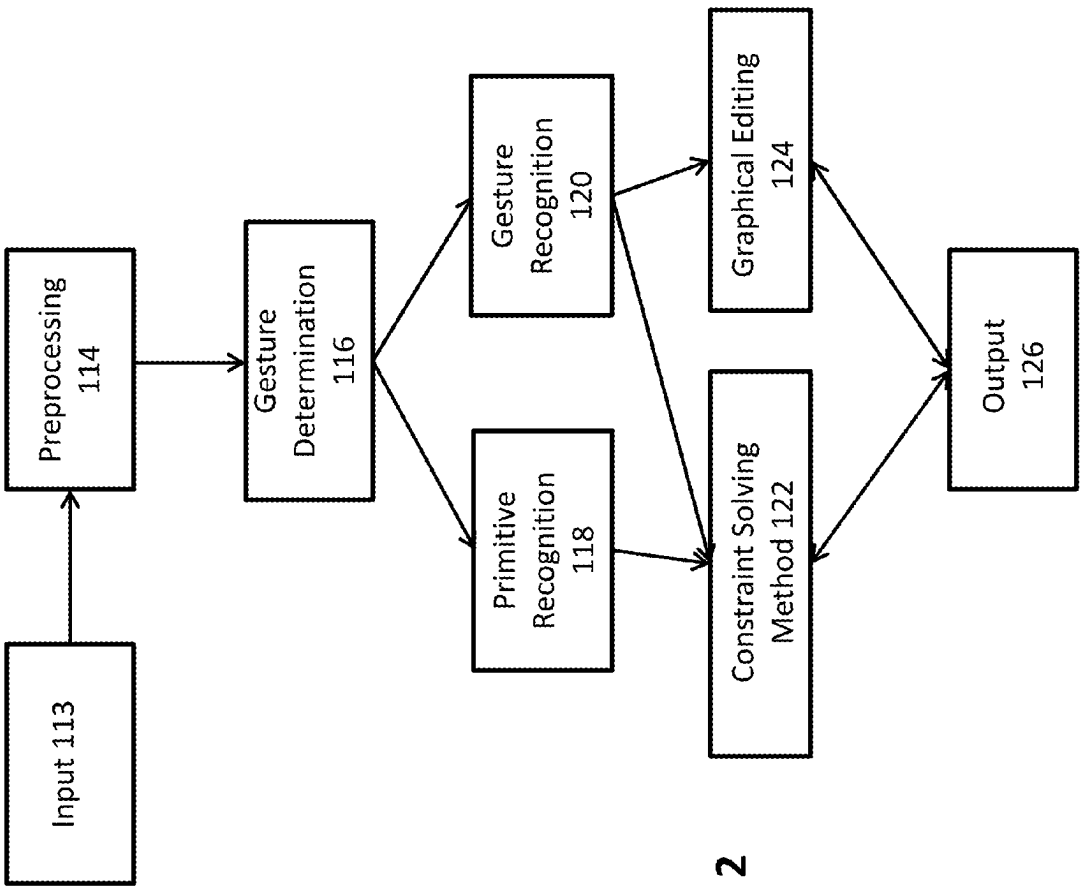

Input 113
Preprocessing 114
Gesture Determination 116
Gesture Recognition 120
Primitive Recognition 118
Graphical Editing 124
Constraint Solving Method 122
Output 126

FIG. 2

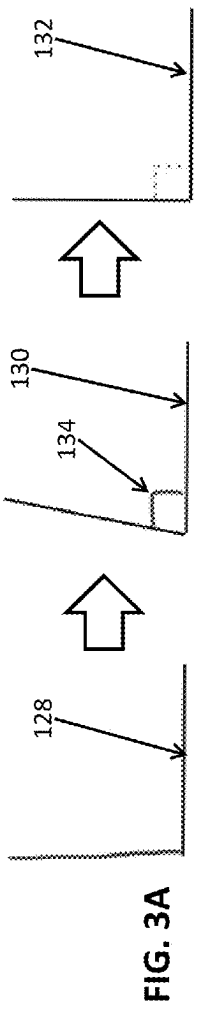


FIG. 3A

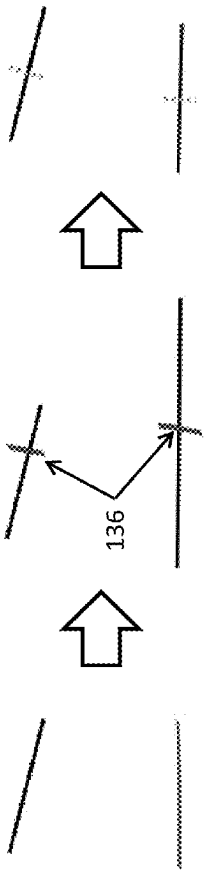


FIG. 3B

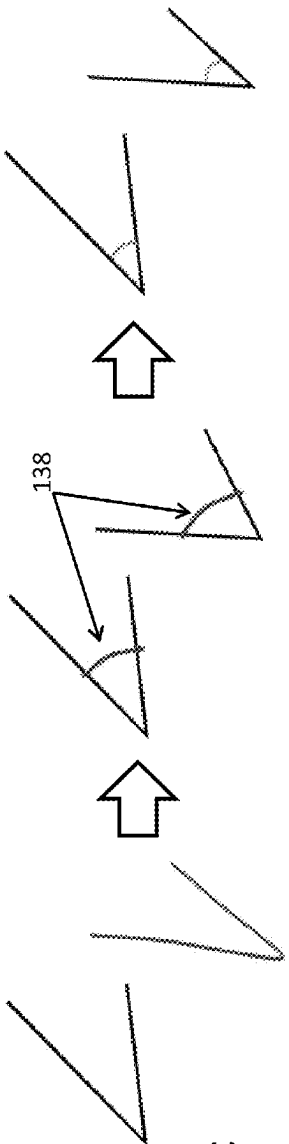


FIG. 3C

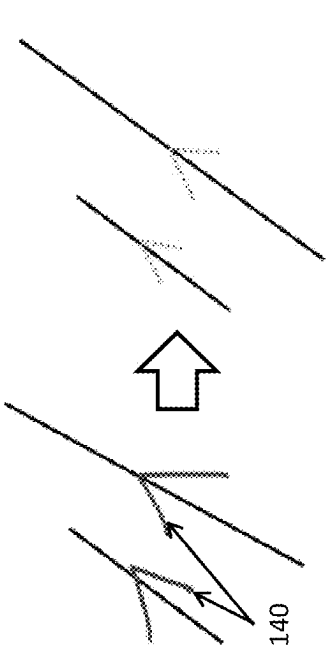


FIG. 3D

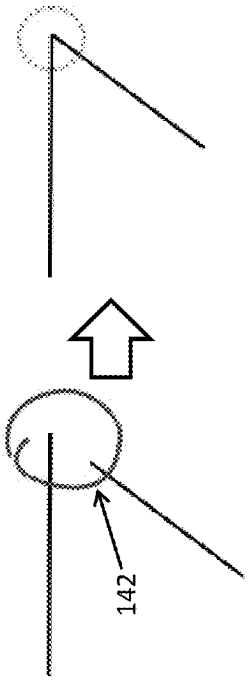


FIG. 3E



FIG. 3F

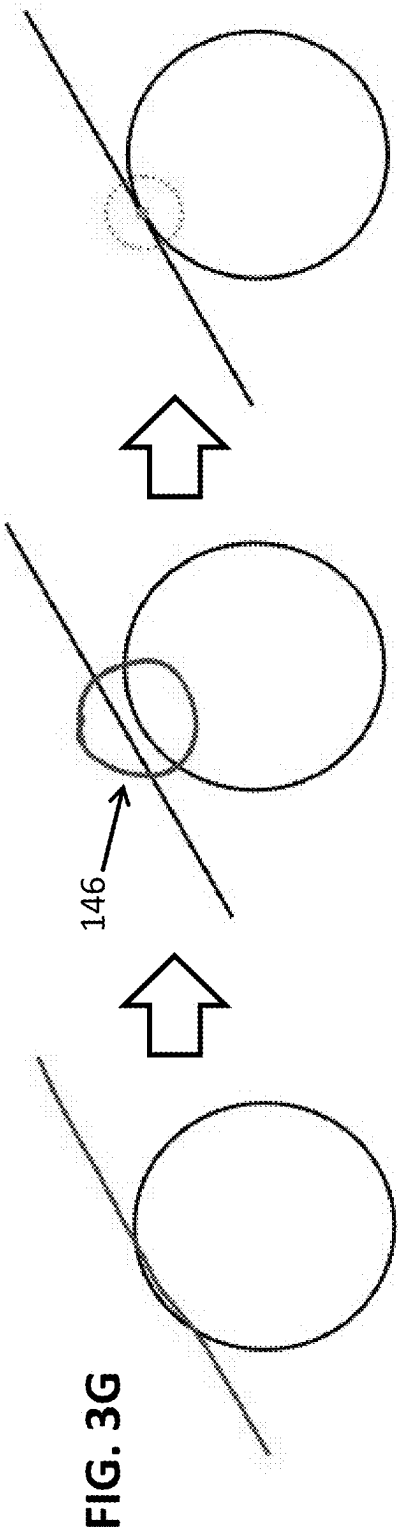


FIG. 3G

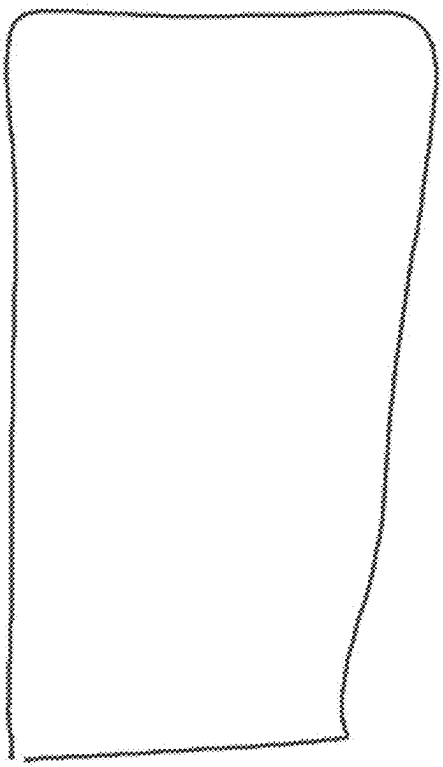


FIG. 4A

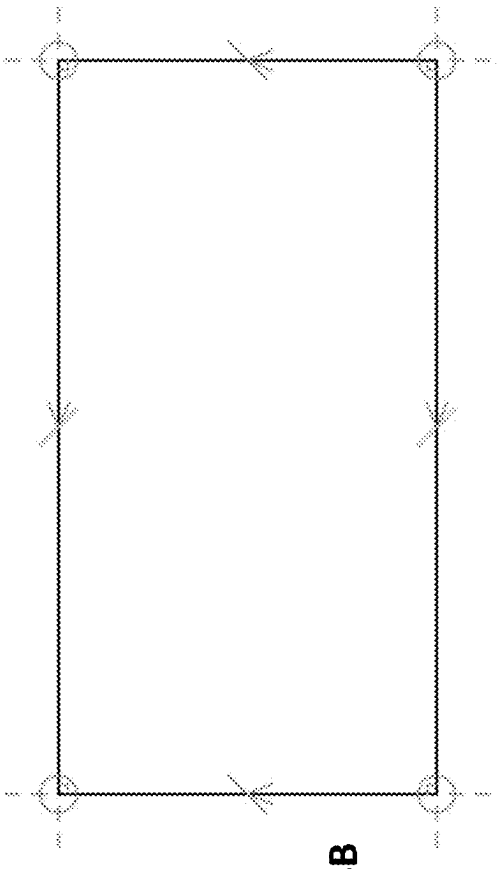


FIG. 4B

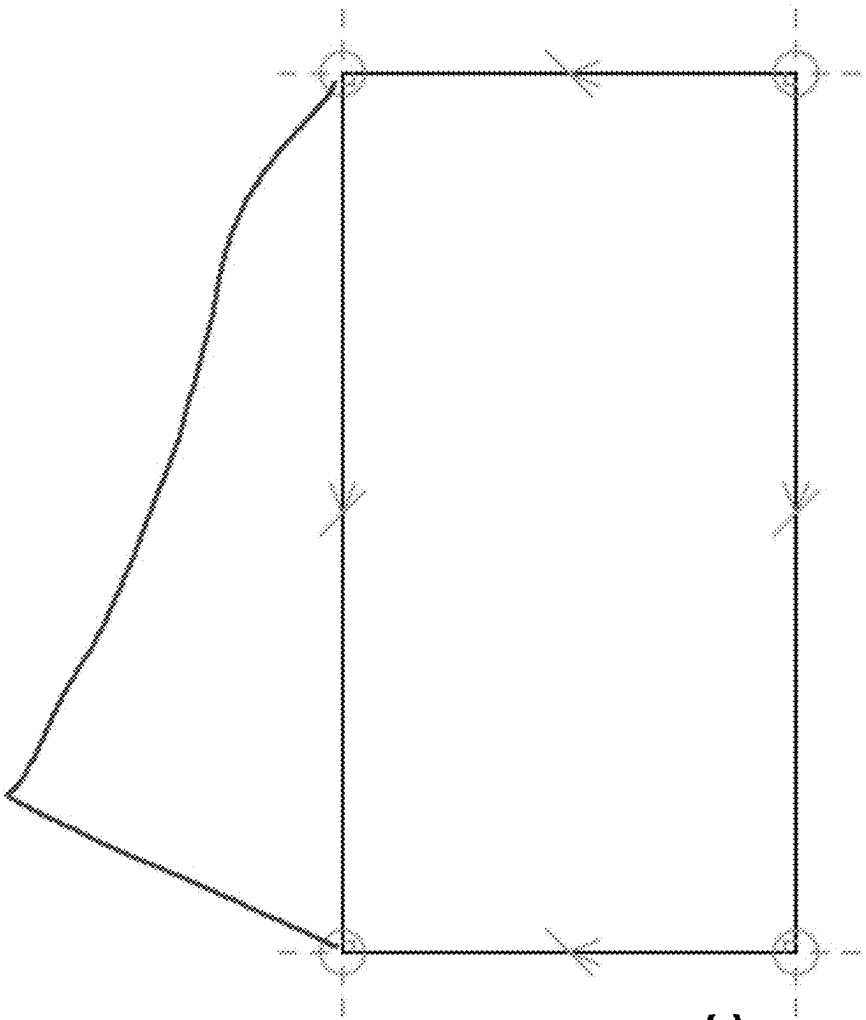


FIG. 4C

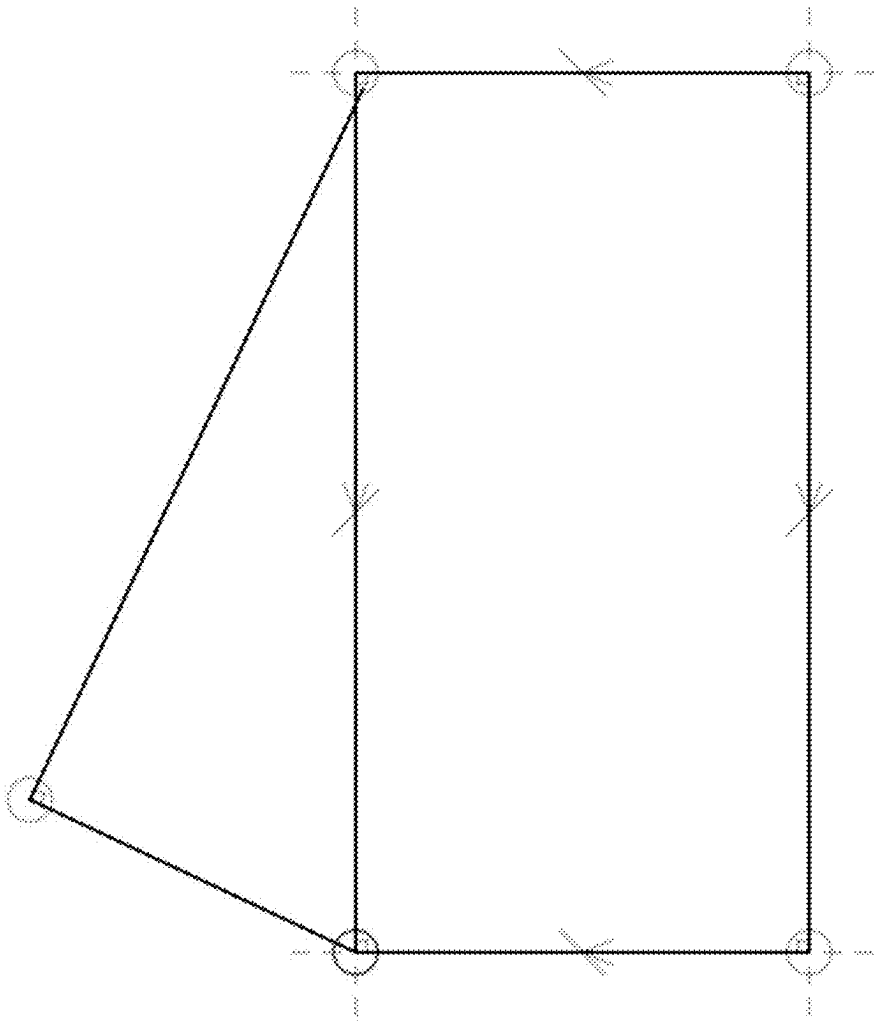


FIG. 4D

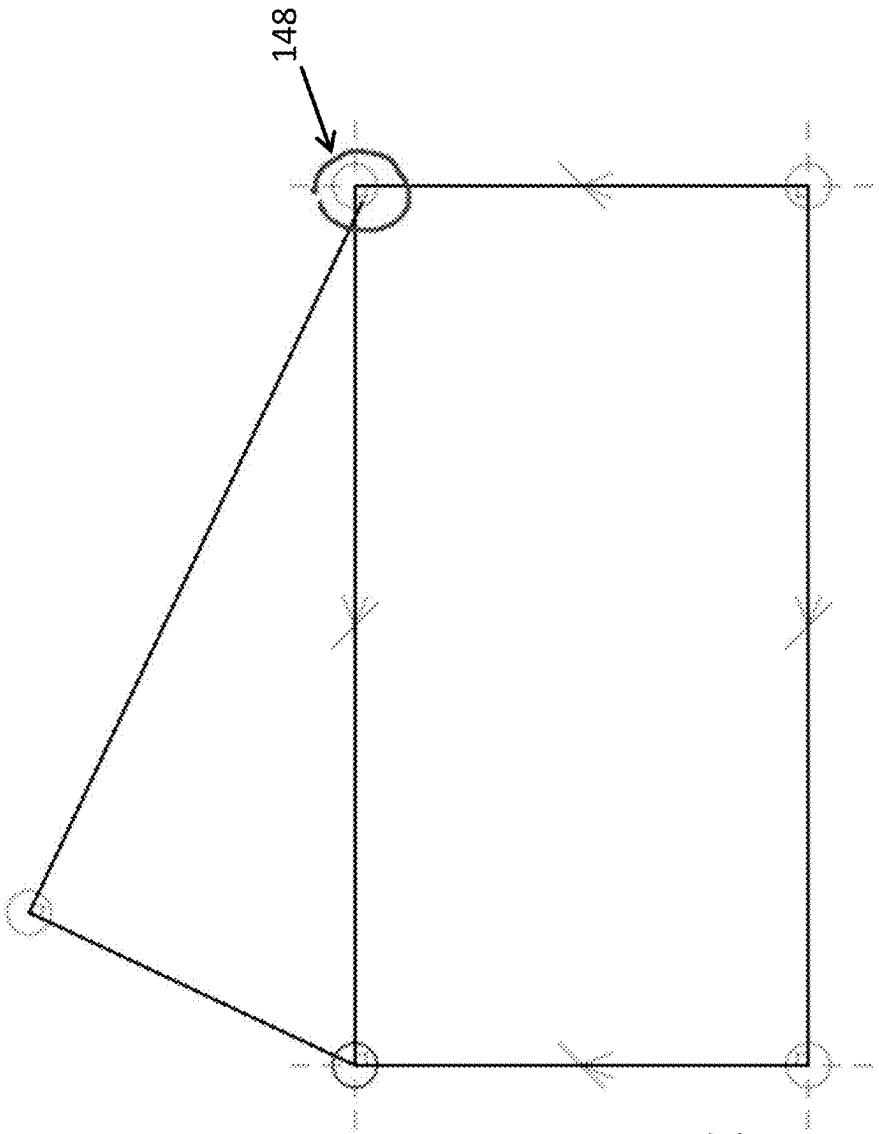


FIG. 4E

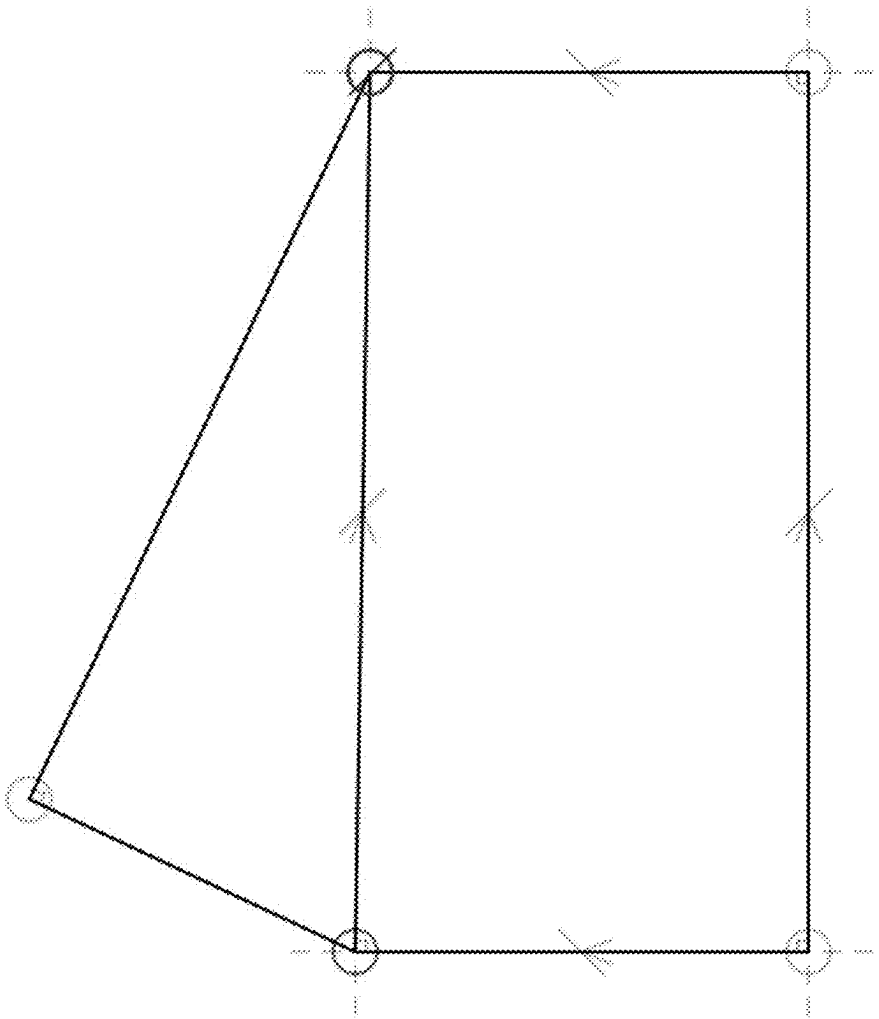


FIG. 4F

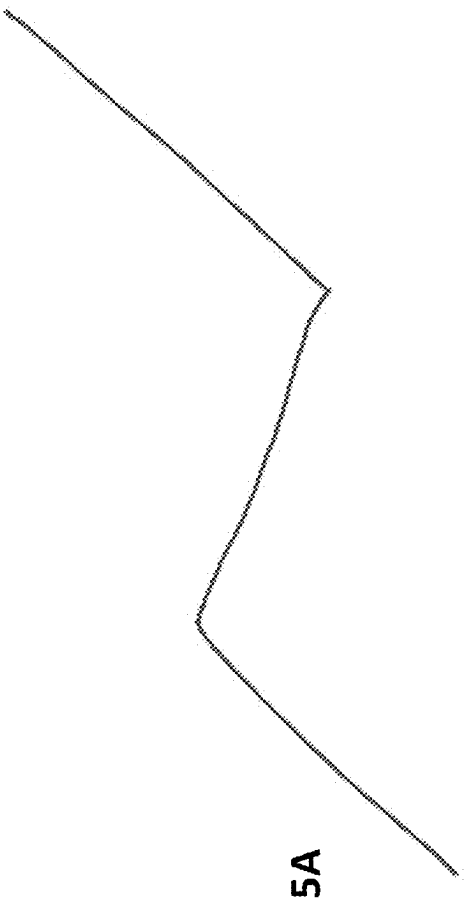


FIG. 5A

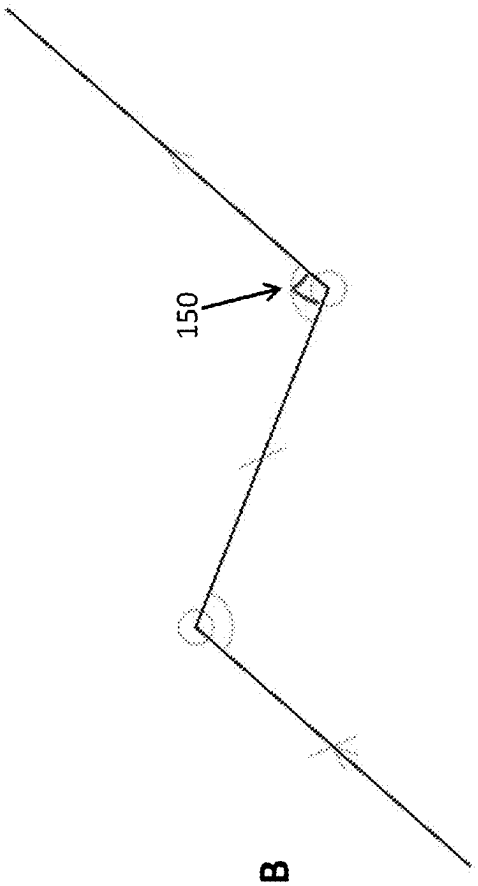


FIG. 5B

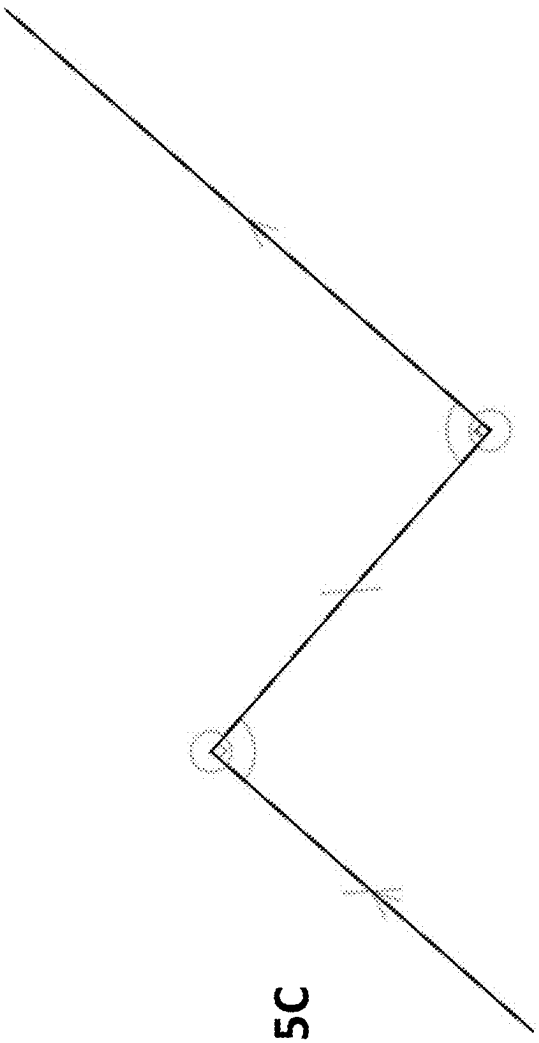


FIG. 5C

SYSTEM AND METHOD FOR RECOGNIZING GEOMETRIC SHAPES

TECHNICAL FIELD

The present invention relates generally to the field of computing device interfaces capable of recognizing user input of geometric shapes.

BACKGROUND

Computing devices continue to become more ubiquitous to daily life. Such devices take the form of computer desktops, laptops, tablet PCs, e-book readers, interactive whiteboards, mobile phones, smartphones, wearable computers, global positioning system (GPS) units, enterprise digital assistants (EDAs), personal digital assistants (PDAs), game consoles, and the like. Further, computing devices are being incorporated into cars, trucks, farm equipment, manufacturing equipment, building environment control (e.g., lighting, HVAC), and home and commercial appliances.

Computing devices generally consist of at least one processing element, such as a central processing unit (CPU), some form of memory, and input and output devices. The numerous varieties of computing devices as well as their subsequent uses necessitate a variety of input devices. One type of input device is a touch sensitive surface such as a touch screen or touch pad wherein user input is received through contact between the user's finger or an instrument such as a pen or stylus and the touch sensitive surface. Another type of input device is an input surface that senses motions made by a user above the input surface and without touching the input surface. Either of these methods of input can be used generally for drawing or inputting shapes. When the user input is a shape, the computing device must interpret the user's natural drawing technique using, for example, an on-line geometric shape recognition system or method.

Generally, on-line geometric shape recognition systems or methods monitor the initiation of a stroke, such as when the user contacts a touch sensitive surface (pen-down); the termination of a stroke, such as when the user stops contacting a touch sensitive surface (pen-up); and any movements (i.e. the stroke) the user makes with his or her finger or pen between the initiation and termination of the stroke.

On-line geometric shape recognition systems or methods may be classified as single-stroke or multi-stroke. Single-stroke recognition analyzes each single-stroke as a separate primitive. A primitive is a geometric shape. The system may only utilize single-stroke primitives or may utilize shorthand for primitives that would otherwise require multiple strokes. These single-stroke systems or methods typically have less input errors but may require users to memorize stroke patterns for multi-stroke primitives if shorthand is utilized. Multi-stroke recognition systems can recognize natural drawing and are more intuitive to users because one does not have to learn shorthand.

Whether one uses a single or multi-stroke system, on-line geometric shape recognition systems or methods usually consist of preprocessing, primitive recognition, and constraint solving. In practice, on-line geometric shape recognition systems or methods may include these steps along with additional steps. Further, on-line geometric shape recognition systems or methods may not clearly delineate each step or may perform the steps in an alternate order.

Preprocessing typically includes discarding irrelevant input data and normalizing, sampling, and removing noise from relevant data.

Primitive recognition specifies the different ways to break down the input data into individual lines, shapes, or other segments based on a set of basic models (e.g., lines, circles, ellipses, arcs, and points). Primitive recognition generally associates the segments with possible primitive candidates. Primitives can be any type of shape or curve, from basic (e.g., line segment, circle, ellipse, arc of circle, arc of ellipse, and point) to more complex (e.g., sinusoidal, spline, Bezier, and parametric function plots). Primitive candidates can comprise single-stroke, multi-stroke, and/or geometric variations of shapes. Primitive recognition may also be preceded by gesture determination that determines whether the input data is geometric shape or a gesture (e.g., an input stroke that provides additional information as to the relation of already input geometric shapes).

Constraint solving generally includes checking the primitives against a set of constraints (e.g., length equality, angle equality, parallelism, symmetry, alignment, tangency, connection, and perpendicularity). Typically, constraints are defined on a primitive level, meaning they are associated with specific primitives. Constraints can be implicit or explicit. Implicit constraints are such that the system infers a constraint based on user input of primitives. For example, the system or method may detect that two lines are close enough to being perpendicular and infer that a perpendicular restraint should apply, outputting two perpendicular lines. In this example, perpendicularity is an example of an implicit constraint. Explicit constraints are such that the user explicitly specifies additional constraints to apply. For instance, the system or method may not detect that two lines were intended to be perpendicular. A user may input an additional gesture to provide this explicit constraint. An on-line geometric shape recognition system or method may recognize both implicit and explicit constraints. Once the set of primitives and set of constraints are built, a constraint solving method enforces the maximum number of possible constraints. Following constraint solving, the on-line geometric shape recognition system or method outputs smooth geometric shapes.

Constraint solving at the primitive level is usually limited in the number of primitives and constraints that a system can analyze because constraints are generally defined at the primitive level and closely linked to primitive types. Each new primitive or constraint introduced into the system causes exponentially more relationships between primitives and constraints, creating exponentially more possibilities to analyze during constraint solving.

The present system and method provides improved results for user input recognition by defining the constraints at a vector component level rather than primitive level. By defining constraints on the vector component level, the system and method is able to better handle increasingly large sets of primitives and constraints.

SUMMARY

The examples of the present invention that are described herein below provide methods, systems, and software for use in on-line geometric shape recognition. These permit a user to enter geometric shapes into a computing device as one would on a piece of paper. The present shape recognition system and method includes a computing device connected to an input device in the form of an input surface. A user is able to provide input by applying pressure to or gesturing

above the input surface using either his or her finger or an instrument such as a stylus or pen. The present system and method monitors the input strokes. After preprocessing the input strokes, the present system and method determines whether or not the input stroke is a gesture. If the input stroke is a gesture, the present system and method matches the gesture with an instruction (e.g., add an additional constraint or delete a drawing) to apply to a preexisting drawing. If the input is not a gesture, the present system and method determines which primitive closely matches the input stroke. Finally, the present system and method applies the set of constraints to the set of primitives to output smooth geometric shapes. After the output is displayed as a drawing on the output device, the user may proceed to adjust or otherwise enrich the output drawing with more shapes or constraints.

An object of the disclosed system and method is to provide an on-line geometric shape recognition system and method that can interpret a user's natural drawing style and output cleaned-up geometric shapes. This can be done by providing a system and method whereby the input strokes are associated with either primitives or gestures that the user can use to explicitly define constraints. Using the primitives and constraints associated with the user input, the system is able to output smooth geometric shapes.

Yet a further object of the disclosed system and method is to provide a simpler definition of constraints. This can be done by providing a system and method that defines constraints on a vector component level rather than on a primitive level.

Another object of the disclosed system and method is to relate primitives and constraints at the vector component level. This can be done by providing a system and method that breaks down primitives into sets of vector components and analyzes the relationship between the constraints and primitives at the vector component level.

An object of the disclosed system and method is also to provide an on-line geometric shape recognition system and method that allows further editing of the output. This can be done by providing a system and method that recognizes certain input editing gestures.

The present system and method will be more fully understood from the following detailed description of the examples thereof, taken together with the drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** shows a block diagram of a computing device in accordance with an example of the present system.

FIG. **2** shows a block diagram of a system for on-line geometric shape recognition in accordance with an example of the present system.

FIG. **3**A shows a pictorial illustration of a possible perpendicularity constraint in accordance with an example of the present system.

FIG. **3**B shows a pictorial illustration of a possible length equality constraint in accordance with an example of the present system.

FIG. **3**C shows a pictorial illustration of a possible angle equality constraint in accordance with an example of the present system.

FIG. **3**D shows a pictorial illustration of a possible parallelism constraint in accordance with an example of the present system.

FIG. **3**E shows a pictorial illustration of a possible junction constraint involving ends of line segments in accordance with an example of the present system.

FIG. **3**F shows a pictorial illustration of a possible junction constraint involving the end of a line segment and line segment in accordance with an example of the present system.

FIG. **3**G shows a pictorial illustration of a possible junction constraint involving a line segment and a portion of a circle in accordance with an example of the present system.

FIG. **4**A shows a pictorial illustration of a possible user input in accordance with an example of the present system.

FIG. **4**B shows a pictorial illustration of a possible output in accordance with an example of the present system.

FIG. **4**C shows a pictorial illustration of a possible output with possible additional user input in accordance with an example of the present system.

FIG. **4**D shows a pictorial illustration of a possible output in accordance with an example of the present system.

FIG. **4**E shows a pictorial illustration of a possible output with possible additional user input in accordance with an example of the present system.

FIG. **4**F shows a pictorial illustration of a possible output in accordance with an example of the present system.

FIG. **5**A shows a pictorial illustration of a possible user input in accordance with an example of the present system.

FIG. **5**B shows a pictorial illustration of a possible output with possible additional user input in accordance with an example of the present system.

FIG. **5**C shows a pictorial illustration of a possible output in accordance with an example of the present system.

DETAILED DESCRIPTION

In the following detailed description, numerous specific details are set forth by way of examples in order to provide a thorough understanding of the relevant teachings. However, it should be apparent to those skilled in the art that the present teachings may be practiced without such details. In other instances, well known methods, procedures, components, and/or circuitry have been described at a relatively high-level, without detail, in order to avoid unnecessarily obscuring aspects of the present teachings.

The various technologies described herein generally relate to on-line geometric shape recognition. The system and method described herein may be used to recognize a user's natural drawing style input through the processes of primitive recognition and constraint solving. The process of constraint solving can include the breaking down and comparison of primitives and constraints at the vector component level.

FIG. **1** shows a block diagram of an exemplary computing device **100**. The computing device may be a computer desktop, laptop, tablet PC, e-book reader, mobile phone, smartphone, wearable computer, digital watch, interactive whiteboard, global positioning system (GPS) unit, enterprise digital assistant (EDA), personal digital assistant (PDA), game console, or the like. Device **100** includes at least one processing element, some form of memory and input and/or output (I/O) devices. The components communicate with each other through inputs and outputs, such as connectors, lines, buses, cables, buffers, electromagnetic links, networks, modems, transducers, IR ports, antennas, or others known to those of ordinary skill in the art.

The device **100** includes at least one input surface **104**. The input surface **104** may employ technology such as resistive, surface acoustic wave, capacitive, infrared grid, infrared acrylic projection, optical imaging, dispersive signal technology, acoustic pulse recognition, or any other

appropriate technology as known to those of ordinary skill in the art. The input surface **104** may be bounded by a permanent or video-generated border that clearly identifies its boundaries.

In addition to the input surface **104**, the device **100** may include one or more additional I/O devices (or peripherals) that are communicatively coupled via a local interface. The local interface may have additional elements to enable communications, such as controllers, buffers (caches), drivers, repeaters, and receivers, which are omitted for simplicity but known to those of skill in the art. Further, the local interface may include address, control, and/or data connections to enable appropriate communications among the other computer components.

One such I/O device may be at least one display **102** for outputting data from the computing device such as images, text, and video. The display **102** may use LCD, plasma, CRT, or any other appropriate technology as known to those of ordinary skill in the art. At least some or all of display **102** may be co-located with the input surface **104**. Other additional I/O devices may include input devices such as a keyboard, mouse, scanner, microphone, touchpads, bar code readers, laser readers, radio-frequency device readers, or any other appropriate technology known to those of ordinary skill in the art. Furthermore, the I/O devices may also include output devices such as a printer, bar code printers, or any other appropriate technology known to those of ordinary skill in the art. Finally, the I/O devices may further include devices that communicate both inputs and outputs such as a modulator/demodulator (modem; for accessing another device, system, or network), a radio frequency (RF) or other transceiver, a telephonic interface, a bridge, a router, or any other appropriate technology known to those of ordinary skill in the art.

The device **100** also includes a processor **106**, which is a hardware device for executing software, particularly software stored in the memory **108**. The processor can be any custom made or commercially available general purpose processor, a central processing unit (CPU), a semiconductor based microprocessor (in the form of a microchip or chip-set), a macroprocessor, microcontroller, digital signal processor (DSP), application specific integrated circuit (ASIC), field programmable gate array (FPGA) or other programmable logic device, discrete gate or transistor logic, discrete hardware components, state machine, or any combination thereof designed for executing software instructions known to those of ordinary skill in the art. Examples of suitable commercially available microprocessors are as follows: a PA-RISC series microprocessor from Hewlett-Packard Company, an 80×86 or Pentium series microprocessor from Intel Corporation, a PowerPC microprocessor from IBM, a Sparc microprocessor from Sun Microsystems, Inc., a 68xxx series microprocessor from Motorola Corporation, DSP microprocessors, or ARM microprocessors.

The memory **108** can include any one or a combination of volatile memory elements (e.g., random access memory (RAM, such as DRAM, SRAM, SDRAM, etc.)) and non-volatile memory elements (e.g., ROM, EPROM, flash PROM, EEPROM, hard drive, magnetic or optical tape, memory registers, CD-ROM, WORM, DVD, redundant array of inexpensive disks (RAID), another direct access storage device (DASD), etc.). Moreover, memory **108** may incorporate electronic, magnetic, optical, and/or other types of storage media. The memory **108** can have a distributed architecture where various components are situated remote from one another but can also be accessed by the processor **106**. The memory **108** is coupled to a processor **106**, so the processor **106** can read information from and write information to the memory **108**. In the alternative, the memory **108** may be integral to the processor **106**. In another example, the processor **106** and the memory **108** may both reside in a single ASIC or other integrated circuit.

The software in memory **108** includes the on-line geometric shape recognition computer program, which may include one or more separate programs, each of which comprises an ordered listing of executable instructions for implementing logical functions. The operating system **110** controls the execution of the on-line geometric shape recognition computer program. The operating system **110** may be any proprietary operating system or a commercially available operating system, such as WEBOS, WINDOWS®, MAC and IPHONE OS®, LINUX, and ANDROID. It is understood that other operating systems may also be utilized without departing from the spirit of the system and method disclosed herein.

The memory **108** may include other application programs **112** related to geometric shape recognition as described herein, totally different functions, or both. The applications **112** include programs provided with the device **100** upon manufacture and may further include programs downloaded into the device **100** after manufacture. Some examples include a text editor, telephone dialer, contacts directory, instant messaging facility, computer-aided design (CAD) program, email program, word processing program, web browser, and camera.

The on-line geometric shape recognition computer program with support and compliance capabilities may be a source program, executable program (object code), script, or any other entity comprising a set of instructions to be performed. When a source program, the program needs to be translated via a compiler, assembler, interpreter, or the like, which may or may not be included within the memory, so as to operate properly in connection with the operating system. Furthermore, the on-line geometric shape recognition computer program with support and compliance capabilities can be written as (a) an object oriented programming language, which has classes of data and methods; (b) a procedure programming language, which has routines, subroutines, and/or functions, for example but not limited to C, C++, Pascal, Basic, Fortran, Cobol, Perl, Java, and Ada; or (c) functional programming languages for example but no limited to Hope, Rex, Common Lisp, Scheme, Clojure, Racket, Erlang, OCaml, Haskell, and F#.

The system may be initiated when processor **106** detects a user entered stroke via the input surface **104**. The user may enter a stroke with a finger or some instrument such as a pen or stylus. The user may also enter a stroke by making gesture above input surface **104** if technology that senses motions above input surface **104** is being used. A stroke is characterized by at least the stroke initiation location, the stroke termination location, and the path upon which the user connects the stroke initiation and termination locations.

FIG. **2** is a schematic pictorial of a system for on-line geometric shape recognition in accordance with an example of the present system. Once initiated with at least one input stroke **113**, the system preprocesses **114** the strokes. Preprocessing **114** can include normalizing the path connecting the stroke initiation and termination locations by applying size normalization and/or methods such as B-spline approximation to smooth the input.

After preprocessing, the system determines whether the stroke is a gesture or not in the context of an existing drawing. A gesture is an input stroke that provides additional information regarding the existing drawing. If the gesture

determination **116** recognizes a gesture, the system then passes the input to gesture recognition **120**. The system may then recognize the input as an annotation defining an explicit constraint the user may wish to impose on existing geometric shapes. In one embodiment, the gesture recognition **120** may use manually tuned heuristics based on polyline approximation of the gesture strokes and their position compared to the primitives to detect the gestures and distinguish between them in context of the existing drawing. However, any gesture recognition **120** method known to those of ordinary skill in the art may be used. The recognized gesture is passed on to the constraint solving method **122**. The constraint solving method **122** will be discussed below. The gesture may also be associated with graphical editing instructions. In this case, the gesture is passed onto the graphical editing **124**.

The graphical editing **124** may be any method known to those of ordinary skill in the art to edit existing output **126**. Such graphical editing **124** typically uses information from the gesture recognition **120** and the output **126** to adjust the current drawing by erasing it, filling in shapes with patterns or colors, changing the thickness or highlighting certain lines, or otherwise editing the current output **126** in ways known to those of ordinary skill in the art.

If gesture determination **116** does not detect a gesture, the input is passed to the primitive recognition **118**. Primitives may include the basic geometric shapes (e.g., line segment, circle, ellipse, arc of circle, arc of ellipse, and point) or additionally more complex shapes (e.g., sinusoidal, spline, Bezier, and parametric function plots). The system may also define a "junk" primitive, which cause the system to discard the input as an unrecognized primitive. Depending on the number of primitives defined in the system, primitive recognition **118** may include single-stroke primitive recognition that treats each stroke as a separate primitive to process. Primitive recognition **118** may also include multi-stroke recognition that treats multiple consecutive input strokes as single primitive.

The system may employ any combination of single and multi-stroke recognition methods known to those of ordinary skill in the art to determine which primitive most closely matches the input **113**. One example of primitive recognition **118** is to fit the input **113** with all primitives, giving a score for each fitting attempt. Dynamic programming may then be used to select the best set of primitives to encompass all of the input strokes based on the best overall fitting scores. Once the best set of primitives has been chosen, the set of primitives is sent to the constraint solving method **122**.

The constraint solving method **122** may receive input from the primitive recognition **118**, gesture recognition **120**, and/or output **126**. Constraints link primitives together (e.g., length equality, angle equality, parallelism, symmetry, alignment, tangency, connection, and perpendicularity). FIGS. **3**A-**3**G are pictorial illustrations of possible constraints. In each case, the far left illustration is the input **113** and the far right illustration is the output **126**. The middle illustration is the application of constraints by the constraint solving method **122**. Constraints can be implicit, deduced by the constraint solving method **122** depending on the primitives provided by the primitive recognition **118**, or can be explicit, defined by the user through gestures. For instance, the constraint solving method **122** may deduce the two line segments input **128** by the user in FIG. **3**A were intended to be perpendicular and automatically apply the perpendicularity constraint, outputting **132** the far right illustration of FIG. **3**A. In this example, the middle illustration **130** represents the constraint solving method **122** adding the perpendicularity constraint internally.

In another instance, the constraint solving method **122** may not detect that the user intended two perpendicular line segments in FIG. **3**A and output the non-perpendicular line segments in the middle illustration **130**. The user may then annotate the junction of the two line segments with a square **134**, which the system would recognize as a gesture. The gesture recognition **120** would pass this gesture onto the constraint solving method **122**, which would adjust the output to be two perpendicular line segments, as seen in the output **132** of far right illustration in FIG. **3**A.

In FIGS. **3**A-**3**G, the constraints can be deduced by the constraint solving method **122** or explicitly annotated by the user. The annotations of these figures (i.e., a square **134** for perpendicularity in FIG. **3**A, two small line segments **136** for length equality in FIG. **3**B, two arcs **138** for angle equality in FIG. **3**C, two arrows **140** for parallelism in FIG. **3**D, a circle **142** around the ends of line segments to form an junction in FIG. **3**E, a circle **144** around the end of a line segment and line segment to form a junction in FIG. **3**F, and a circle **146** around a line segment and portion of a circle for a junction or tangency in FIG. **3**G) are one embodiment, and the present system and method may use any gestures for these constraints as well as others. Further, constraints naturally associated with primitives may be deduced during primitive recognition **118** prior to the constraint solving method **122**. For instance, the primitive recognition **118** may recognize a square from the input and also apply the constraints of four equal sides, four internal right angles, and parallel opposite sides. If primitive recognition **118** recognizes a polyline, it may also apply the junction constraint where the lines intersect. However, for clarity this specification will only refer to the constraint solving method **122** as applying all constraints even though it is envisioned that the primitive recognition **118** may handle this function in some situations.

The constraint solving method **122** includes defining constraints and primitives at the vector component level rather than at a primitive level. When defining constraints at the primitive level, constraints are associated with specific primitives. However, by defining constraints at the vector component level a user may add additional constraints and primitives without adding exponentially more work. For example, when the junction constraint is defined and applied at the primitive level with two line segments, it must first be associated with the beginning or ending of a line segment. Once a circle primitive is added to the present system and method, the junction constraint must also be associated with the circle center together with other line segment externalities. This constraint becomes even more complicated once ellipses and other more complex primitives are added to the present system and method, even though the constraint is still defining just a junction between two points.

To avoid this exponential increase in complexity, the constraint solving method **122** breaks to down both primitives and constraints at the vector component level, such that the method focuses on the relationships between points. In one embodiment, the constraint solving method **122** may define at least three items: points, lengths, and slope angles. To define primitives as items, the constraint solving method **122** may break down the primitives into their items. For example, a line segment primitive may be defined by the two extremity points, a slope, and a length. Another example is a circle primitive defined by a central point and radius length. With the primitives broken down to their items, the constraint solving method **122** applies vector component

constraints. Even though the constraint solving method **122** may include breaking down the primitives into their vector component items, in other embodiments, the primitive recognition **118** may break down the primitives into the vector components prior to passing the information to the constraint solving method **122**. In this alternative embodiment, the primitive items from the primitive recognitions **118** would still have constraints applied to them by the constraint solving method **122**.

Unlike constraints defined on the primitive level, constraints defined at the vector component level link items together. Vector component constraints may include:

- Slope Angle Constraint defined between two points and one slope angle;
- Length Constraint defined between two points (A, B) and one length (L) so that ||AB||=L;
- Projection Constraint defined between two points (A, B), a length (L) and a slope(s) so that B is the projection of A following slope(s) at distance (L) and vice-versa;
- Junction Constraint defined between two points (A, B) to ensure A=B;
- Slope Angle Difference Constraint defined between two slope angles (s1, s2) and a given value (v) so that s2−s1=v;
- Length Ratio Constraint defined between two lengths (L1, L2) and a given value (r) so that L1/L2=r;
- Length Sum Constraint defined between three points (A, B, C) and one length (L) so that ||AC||+||BC||=L; and
- Existence Constraint defined for any point, and useful if it is isolated and not linked by any other constraint

Other constraints may be defined in a similar manner as would be known to those of skill in the art. Further, any constraint may be defined as a combination of vector component constraints. One benefit of using vector component constraints is that the constraint solving method **122** may deduce the value of an item if some of the other items are known. As an example, the Length Constraint links two point items (A, B) to a length item L. So, if A and B are known, the value of L can be determined. Conversely, if A and L are known, B can be defined as on a circle of radius L centered on A, which information is added to the intersection parameter of B. The same applies if we switch A and B. By breaking down both primitives and constraints on a vector component level the present system and method will be able to solve more situations.

Once the constraint solving method **122** has broken down the primitives into items, the constraint solving method **122** may apply all possible constraints to the primitives or may build a list of specific constraints based on priority. By placing constraints in a hierarchy, the constraint solving method **122** avoids applying contradictory constraints. For instance, constraints that maintain primitive integrity may be high priority. This ensures that primitives do not change their inherent shape (e.g., circles will remain a point and a radius). A second priority may constraints that the constraint solving method **122** implicit in the primitives such as a junction of a polyline. Other lower priority constraints may be those defined explicitly by the user (e.g., user defined parallelism) or those for pairs of primitives (e.g., a line segment tangentially intersecting to a circle). However, this hierarchy is only one example, and the present system and method is able to prioritize constraints in any manner one skilled in the art would necessitate. After the constraint solving method **122** applies the proper constraints to the primitives, it rebuilds the primitives from their vector component levels to produce output **126** in the form of smooth geometric shapes.

FIGS. **4**A-**4**F show a schematic pictorial illustration of possible inputs and outputs relating to a rectangle and polyline in accordance with an example of the present system. FIG. **4**A shows a schematic illustration of an input of a rectangle. After preprocessing **114**, the gesture determination **116** would not define this input as a gesture and pass the input on for primitive recognition **118**. After primitive recognition **118** determined the input was a rectangle, the information would be passed on to the constraint solving method **122**. The constraint solving method **122** would apply the constraints known for a rectangle (e.g., four line segments that intersect at four interior right angles, and opposite sides are parallel and of equal length). This would produce the rectangle shown in FIG. **4**B as output.

A user may further add to the rectangle with input shown FIG. **4**C. The primitive recognition **118** would recognize this as a polyline primitive. The constraint solving mechanism **122** would also detect a junction in the middle of the new polyline that is a right angle as well as junction with corner of the previous rectangle. This would produce new output shown in FIG. **4**D. A user may want another junction that was not detected and add annotation **148** to the output as shown in FIG. **4**E. This annotation **148** would be processed as a gesture, recognized as a gesture to indicate a junction, and passed on to the constraint solving method **122**. The constraint solving method would take output along with this gesture and all other relevant constraints to produce new output shown in FIG. **4**F.

FIGS. **5**A-**5**C show a schematic pictorial illustration of possible input and output relating to a polyline in accordance with an example of the present system. FIG. **5**A shows a schematic illustration of an input of a polyline. After preprocessing **114**, the gesture determination **116** would not define this input as a gesture and pass the input on for primitive recognition **118**. After primitive recognition **118** determined the input was a polyline, the information would be passed on to the constraint solving method **122**. The constraint solving method **122** would apply the constraints detected by in the primitive polyline: two junctions between three line segments, parallelism between the first and third line segments, the corresponding angle quality between the two angles, and length equality between the first and second line segments. This would produce the polyline output shown in FIG. **5**B as output. A user may further indicate perpendicularity as shown with the additional small box annotation **150** on the second angle shown FIG. **5**B. This annotation **150** would be processed as a gesture, recognized as a gesture to indicate perpendicularity, and passed on to the constraint solving method **122**. The constraint solving method would take output along with this gesture and all other relevant constraints to produce new output shown in FIG. **5**C.

Through the present on-line geometric shape recognition system and method, the best results for user input drawing recognition are provided by breaking down primitives and constraints on a vector component level. By applying constraints to primitives on the vector component level and rebuilding the altered primitives for output, the present on-line geometric shape recognition system and method is able to handle more primitives and constraints without an exponential increase in difficulty.

While the foregoing has described what is considered to be the best mode and/or other examples, it is understood that various modifications may be made therein and that the subject matter disclosed herein may be implemented in various forms and examples, and that they may be applied in numerous other applications, combinations, and environ-

ments, only some of which have been described herein. Those of ordinary skill in that art will recognize that the disclosed aspects may be altered or amended without departing from the true spirit and scope of the subject matter. Therefore, the subject matter is not limited to the specific details, exhibits, and illustrated examples in this description. It is intended to protect any and all modifications and variations that fall within the true scope of the advantageous concepts disclosed herein.

We claim:

1. A non-transitory computer readable medium having a computer readable program code embodied therein, said computer readable program code adapted to be executed to implement a method for providing geometric shape recognition for at least one input stroke, said method comprising:

recognizing at least one primitive that corresponds to at least one input stroke;

applying at least one constraint to the at least one primitive, wherein the at least one constraint and at least one primitive are applied on a vector component level; and

providing an output based on the application of the at least one constraint to the at least one primitive,

wherein if the at least one constraint comprises multiple constraints, the multiple constraints are applied to the at least one primitive based on a priority order, wherein the priority order comprises constraints maintaining the at least one primitive's integrity, constraints implicit in the at least one primitive, and constraints defined by a gesture.

2. The non-transitory computer readable medium according to claim **1**, wherein the at least one primitive comprises a plurality of input strokes.

3. The non-transitory computer readable medium according to claim **1**, wherein the at least one input stroke is at least one user gesture made above an input surface.

4. The non-transitory computer readable medium according to claim **1**, wherein the at least one input stroke is preprocessed, wherein the preprocessing includes at least the normalization and smoothing of the at least one input stroke.

5. The non-transitory computer readable medium according to claim **4**, wherein the at least one primitive is defined on the vector component level as being constituted of at least one point, length, or slope angle.

6. The non-transitory computer readable medium according to claim **5**, wherein the at least one constraint is defined on the vector component level as applying to at least one point, length, or slope angle.

7. The non-transitory computer readable medium according to claim **6**, wherein the at least one constraint to apply to the at least one primitive is associated with the at least one primitive.

8. The non-transitory computer readable medium according to claim **4**, wherein the preprocessed at least one input stroke is analyzed to determine whether the at least one input stroke is a gesture, wherein if the at least one input stroke is a gesture, the gesture can provide at least one further constraint to the output based on the application of the at least one constraint to the at least one primitive.

9. The non-transitory computer readable medium according to claim **8**, wherein the gesture provides graphical editing instructions to apply to the output based on the application of the at least one constraint to the at least one primitive.

10. The non-transitory computer readable medium according to claim **9**, wherein the graphical editing instructions comprise at least a deletion instruction.

11. The non-transitory computer readable medium according to claim **1**, wherein the possible at least one constraint comprises at least one of length equality, angle equality, parallelism, symmetry, alignment, tangency, connection, and perpendicularity.

12. The computer program product according to claim **1**, wherein the possible at least one constraint comprises at least one of Slope Angle Constraint, Length Constraint, Projection Constraint, Junction Constraint, Slope Angle Difference Constraint, Length Ratio Constraint, Length Sum Constraint, or Existence Constraint.

13. The non-transitory computer readable medium according to claim **1**, wherein the at least one primitive comprises at least one of a line segment, circle, ellipse, arc of circle, arc of ellipse, and point.

14. A method for providing geometric shape recognition for at least one input stroke on a computing device, the computing device comprising a processor and at least one non-transitory computer readable medium under control of the processor, said method comprising:

recognizing, by the at least one non-transitory computer readable medium, at least one primitive that corresponds to at least one input stroke;

applying, by the at least one non-transitory computer readable medium, at least one constraint to the at least one primitive, wherein the at least one constraint and at least one primitive are applied on a vector component level; and

providing, by the at least one non-transitory computer readable medium, an output based on the application of the at least one constraint to the at least one primitive,

wherein if the at least one constraint comprises multiple constraints, the multiple constraints are applied to the at least one primitive based on a priority order, wherein the priority order comprises constraints maintaining the at least one primitive's integrity, constraints implicit in the at least one primitive, and constraints defined by a gesture.

15. The method according to claim **14**, wherein the at least one primitive comprises a plurality of input strokes.

16. The method according to claim **14**, wherein the at least one input stroke is at least one user gesture made above an input surface.

17. The method according to claim **14**, wherein the at least one input stroke is preprocessed by the at least one non-transitory computer readable medium, wherein the preprocessing includes at least the normalization and smoothing of the at least one input stroke.

18. The method according to claim **17**, wherein the at least one primitive is defined on the vector component level as being constituted of at least one point, length, or slope angle.

19. The method according to claim **18**, wherein the at least one constraint is defined on the vector component level as applying to at least one point, length, or slope angle.

20. The method according to claim **19**, wherein the at least one constraint to apply to the at least one primitive is associated with the at least one primitive.

21. The method according to claim **17**, wherein the preprocessed at least one input stroke is analyzed to determine whether the at least one input stroke is a gesture, wherein if the at least one input stroke is a gesture, the gesture can provide at least one further constraint to the output based on the application of the at least one constraint to the at least one primitive.

22. The method according to claim **21**, wherein the gesture provides graphical editing instructions to apply to

the output based on the application of the at least one constraint to the at least one primitive.

23. The method according to claim **22**, wherein the graphical editing instructions comprise at least a deletion instruction.

24. The method according to claim **14**, wherein the possible at least one constraint comprises at least one of length equality, angle equality, parallelism, symmetry, alignment, tangency, connection, and perpendicularity.

25. The method according to claim **14**, wherein the possible at least one constraint comprises at least one of Slope Angle Constraint, Length Constraint, Projection Constraint, Junction Constraint, Slope Angle Difference Constraint, Length Ratio Constraint, Length Sum Constraint, or Existence Constraint.

26. The method according to claim **14**, wherein the at least one primitive comprises at least one of a line segment, circle, ellipse, arc of circle, arc of ellipse, and point.

27. A non-transitory computer readable medium having a computer readable program code embodied therein, said computer readable program code adapted to be executed to implement a method for providing geometric shape recognition for at least one input stroke, said method comprising:

preprocessing at least one input stroke;

analyzing the preprocessed the at least one input stroke to determine whether the at least one input stroke is a gesture:

wherein if the at least one input stroke is not a gesture:

recognizing at least one primitive that corresponds to the at least one input stroke;

applying at least one constraint to the at least one primitive, wherein the at least one constraint and the at least one primitive are applied on the vector component level;

wherein if the at least one input stroke is a gesture, the gesture provides at least one further constraint to an output based on the application of the at least one constraint to the at least one primitive, wherein the at least one constraint and the at least one primitive are applied on the vector component level; and

providing an output based on the application of the at least one constraint to the at least one primitive,

wherein if the at least one constraint comprises multiple constraints, the multiple constraints are applied to the at least one primitive based on a priority order, wherein the priority order comprises constraints maintaining the at least one primitive's integrity, constraints implicit in the at least one primitive, and constraints defined by a gesture.

28. The non-transitory computer readable medium according to claim **27**, wherein the at least one primitive comprises a plurality of input strokes.

29. The non-transitory computer readable medium according to claim **28**, wherein the at least one primitive is defined on the vector component level as being constituted of at least one point, length, or slope angle.

30. The non-transitory computer readable medium according to claim **29**, wherein the at least one constraint is defined on the vector component level as applying to at least one point, length, or slope angle.

31. The non-transitory computer readable medium according to claim **30**, wherein the at least one constraint to apply to the at least one primitive is associated with the at least one primitive.

32. The non-transitory computer readable medium according to claim **30**, wherein the gesture provides graphical editing instructions to apply to the output based on the application of the at least one constraint to the at least one primitive.

33. The non-transitory computer readable medium according to claim **32**, wherein the graphical editing instructions comprise at least a deletion instruction.

34. The non-transitory computer readable medium according to claim **27**, wherein the at least one input stroke is at least one user gesture made above an input surface.

35. The non-transitory computer readable medium according to claim **27**, wherein the possible at least one constraint comprises at least one of length equality, angle equality, parallelism, symmetry, alignment, tangency, connection, and perpendicularity.

36. The non-transitory computer readable medium according to claim **27**, wherein the possible at least one constraint comprises at least one of Slope Angle Constraint, Length Constraint, Projection Constraint, Junction Constraint, Slope Angle Difference Constraint, Length Ratio Constraint, Length Sum Constraint, or Existence Constraint.

37. The non-transitory computer readable medium according to claim **27**, wherein the at least one primitive comprises at least one of a line segment, circle, ellipse, arc of circle, arc of ellipse, and point.

* * * * *

UNITED STATES PATENT AND TRADEMARK OFFICE

CERTIFICATE OF CORRECTION

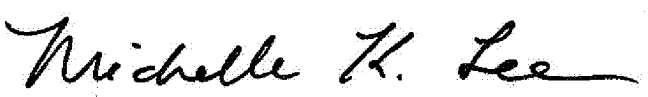
PATENT NO.	: 9,489,572 B2
APPLICATION NO.	: 14/558181
DATED	: November 8, 2016
INVENTOR(S)	: Robin Mélinand et al.

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

On the Title Page

Item (71) Applicant should read -- MyScript -- rather than "(Vision Objects)"

Signed and Sealed this
Fourteenth Day of March, 2017



Michelle K. Lee
Director of the United States Patent and Trademark Office

EXHIBIT O

U.S. Patent No. 9,965,175

Claim Language	Infringement Read
1. A system for digital note taking on computing devices, each computing device comprising a processor and at least one non-transitory computer readable medium for recognizing handwriting note taking input under control of the processor, the at least one non-transitory computer readable medium configured to:	Goodnotes 6 is a digital note taking app that can be used on computing device with iOS, Windows, and Android operating systems. The computing devices on which Goodnotes 6 is used include a processor and a computer readable medium (Goodnotes app) for recognizing handwriting note taking input under control of the processor. *See e.g.*: https://www.goodnotes.com/pricing; https://www.goodnotes.com/features; https://support.goodnotes.com/hc/en-us/articles/10779382123919-Reflow-your-handwriting-with-Lasso-Resize-Mode#h_01J8584JHDY4ME8YX1K4BXC866 (reflow feature)
cause display of, on a display interface of a computing device, digital ink in a block layout in accordance with a layout of geometric blocks of the handwriting input and a configuration of the computing device display interface, said configuration of the computing device being defined by horizontal and vertical dimensions of the display interface, and wherein the at least one non-transitory computer readable medium is configured to:	Goodnotes 6 causes display of, on a display interface of a computing device, digital ink in a block layout in accordance with a layout of geometric blocks of the handwriting input and a configuration of the computing device display interface.

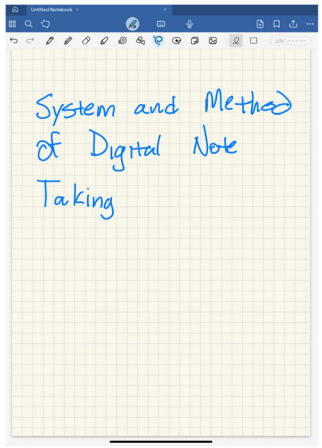


Figure 1

As denoted by the red boxes in **Figure 2**, each of the words is a geometric block of handwriting input.

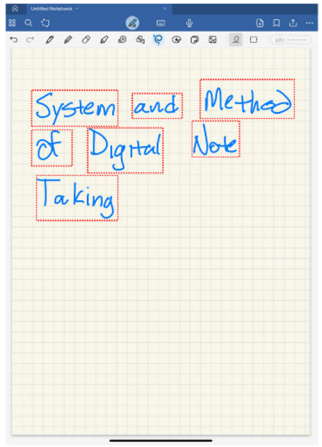
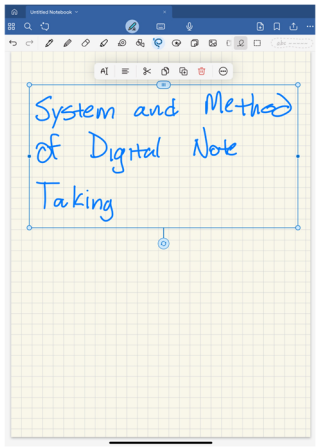


Figure 2

As can also be seen in the above image, the configuration of the computing device is defined by horizontal and vertical dimensions of the display interface (*i.e.*, the edges of the virtual paper).

when the computing device display interface changes from a configuration whereby the geometric blocks of digital ink fit in horizontal adjacency within the horizontal dimension of the display interface to another configuration of reduced horizontal dimension, the reduced horizontal dimension being less than a cumulative horizontal geometric block dimension of the geometric blocks, cause moving of said one or more geometric blocks relatively to said others of geometric blocks while otherwise maintaining the layout of the said others of geometric blocks.	When the computing device display interface changes from a configuration whereby the geometric blocks of digital ink fit in horizontal adjacency within the horizontal dimension of the display interface to another configuration of reduced horizontal dimension, the reduced horizontal dimension being less than a cumulative horizontal geometric block dimension of the geometric blocks, one or more of the geometric blocks is moved relative to said others of geometric blocks while otherwise maintaining the layout of the said others of geometric blocks. As depicted in **Figure 3** and **Figure 4** below, when the horizonal dimension is reduced to a configuration that is less than the cumulative horizontal geometric block dimension of the geometric blocks (blue box), the geometric blocks containing certain words are moved relative to other blocks while otherwise maintaining the layout.  **Figure 3**

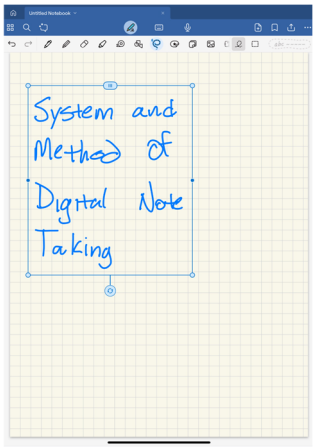


Figure 4

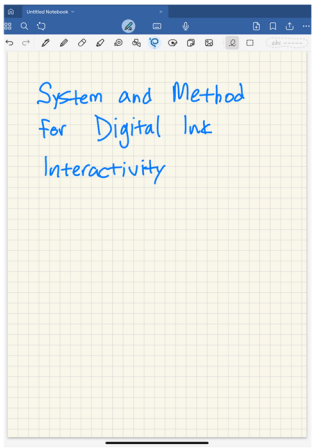
The implementation of this claimed feature is shown in Goodnotes's own video[1] and written materials[2] promoting new features in Goodnotes 6. In the video, the reflow feature is referred to as a "massive upgrade" and demonstrates that when the horizontal dimension is reduced to less than the cumulative horizontal block dimension of the text, portions of the text are moved relative to other portions of the text. *See* Video at 8:00-8:20. Goodnotes also separately touted this feature on Reddit, explaining that one of Goodnotes 6 "new handwriting editing capabilities" is that "[y]our handwriting now reflows as seamlessly as typed text when you lasso and resize width, copy and paste, add space in the middle of sentences, or align left/right/center."

[1] https://www.youtube.com/watch?v=hNWQKI8FzJs, "Goodnotes Event: 3 new features to supercharge your productivity, creativity & learning" (Oct. 3, 2024).
[2] https://www.reddit.com/r/GoodNotes/comments/1fnfft0/new_in_v650_smart_ink_sticky_notes_math_assist/, "New in v6.5.0 - Smart Ink, Sticky Notes, Math Assist, Hover fix, and more (September 23rd, 2024)."

EXHIBIT P

U.S. Patent No. 10,082,951

Claim Language	Infringement Read
1. A system for providing interactive ink on a computing device, the computing device comprising a processor, a memory and at least one non-transitory computer readable medium for recognizing handwriting input under control of the processor, the at least one non-transitory computer readable medium configured to:	Goodnotes 6 is a digital note taking app that can be used on computing device with iOS, Windows, and Android operating systems. The computing devices on which Goodnotes 6 is used include a processor, a memory, and a computer readable medium (Goodnotes app) for recognizing handwriting input under control of the processor. *See e.g.*: https://www.goodnotes.com/pricing; https://www.goodnotes.com/features; https://support.goodnotes.com/hc/en-us/articles/10779382123919-Reflow-your-handwriting-with-Lasso-Resize-Mode#h_01J8584JHDY4ME8YX1K4BXC866 (reflow feature); https://support.goodnotes.com/hc/en-us/articles/7353695997199-Convert-handwriting-to-text-in-Goodnotes-6 (convert-to-text).
cause display of, on a display interface of the computing device, first digital ink in accordance with first handwriting input;	Goodnotes 6 causes display on a display interface of the computing device, first digital ink in accordance with first handwriting input.

	 **Figure 1**
allocate spatial or temporal references to ink elements of the first digital ink;	Goodnotes 6 allocates spatial and or temporal references to ink elements of the first digital ink. Specifically, each of the letters/words is allocated a spatial or temporal reference. This is confirmed by Goodnotes 6 convert-to-text feature and reflow feature. The convert-to-text allows a user to confer handwritten notes created in Goodnotes to typed text by (i) selecting the Lasso Tool from the tool bar; (ii) circling the handwritten notes to convert; (iii) selecting "*Convert*" menu option; (iv) tapping the *Text* menu option to display the text; and (v) selecting the Convert button.

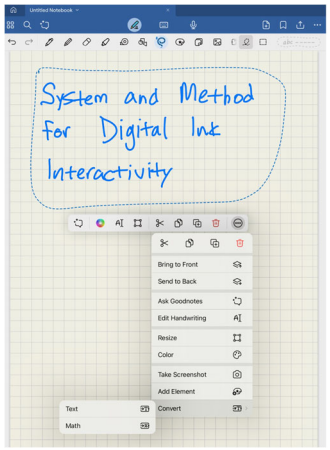


Figure 2

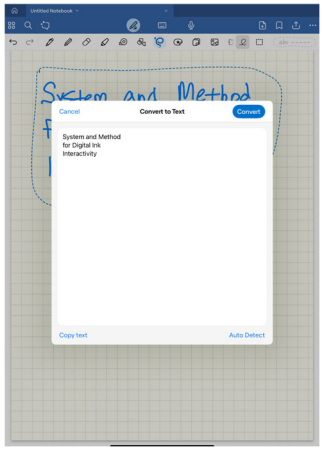


Figure 3

After the "Convert" button is pressed, Goodnotes 6 displays a screen with typed text replacing the corresponding handwritten text.

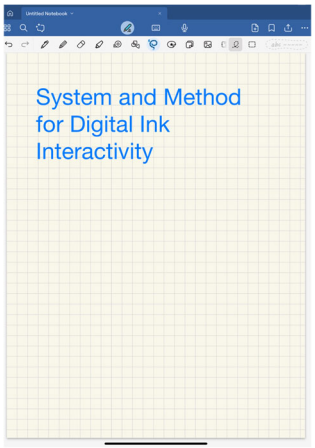
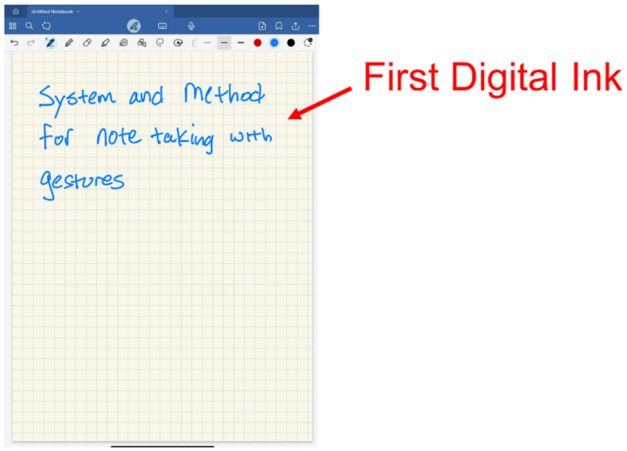
	 **Figure 4** Goodnotes 6 reflow feature also meets this limitation. As discussed with respect to claim 1 of the '175 patent, when the horizontal dimension is reduced to less than the cumulative horizontal block dimension of the text, portions of the text are moved relative to other portions of the text. In order for entire words to be moved relative to one another (as opposed to individual strokes), Goodnotes 6 allocates spatial or temporal references to ink elements of the first digital ink.
map the spatial or temporal references to corresponding recognized elements of the first handwriting input; and	As noted with respect to the previous element, Goodnotes 6 maps the spatial or temporal references to corresponding recognized elements of the first handwriting input. This is evidenced by the convert-to-text feature which successfully converts handwriting input to typed text. Is it also evidenced by the reflow feature which successfully moved words while maintaining an appropriately structured layout.
determine and store, in the memory of the computing device, ink objects including the spatial or temporal references and mapped recognized elements	As noted above with respect to the previous elements, Goodnotes permits a user to convert-to-text or reflow after the digital ink is entered. In order for these features to operate successfully, Goodnotes 6 determines and stores in memory the ink objects along with spatial or temporal references that are mapped to recognized elements.

EXHIBIT Q

U.S. Patent No. 11,402,991

Claim Language	Infringement Read
1. A system for editing digital documents on computing devices, each computing device comprising a processor and at least one non-transitory computer readable medium for recognizing handwriting input under control of the processor, the processor configured to:	Goodnotes 6 is a digital note taking app that can be used on computing device with iOS, Windows, and Android operating systems. The computing devices on which Goodnotes 6 is used include a processor and a non-transitory computer readable medium (Goodnotes app) for recognizing handwriting input under control of the processor. The processor is configured to execute the instructions below based on Goodnotes 6. *See e.g.*: https://www.goodnotes.com/pricing; https://www.goodnotes.com/features; https://support.goodnotes.com/hc/en-us/articles/7443501398671-Scribble-to-Erase-Handwriting (scribble to erase)
process a handwriting input, wherein the handwriting input can represent handwriting content to be added to displayed text of previously processed handwriting content or an editing gesture for enacting control on the displayed text of the previously processed handwriting content;	Goodnotes 6 causes the processor to process a handwriting input. Among other things, the handwriting input to be an editing gesture that enacts control of the displayed text of previously processed handwriting content. Goodnotes refers to this as “Scribble to Erase.”
cause display of, on a display interface of a computing device, first digital ink in accordance with document content;	Goodnotes 6 causes the processor to display on a display interface of a computing device first digital ink in accordance with document content.

	 **Figure 1**
determine whether one or more strokes of the handwriting input correspond to i) the handwriting content to be added to the displayed text of the previously processed handwriting content or ii) one or more editing gestures for enacting control on the displayed text of the previously processed handwriting content by 1) determining characteristics of the handwriting input, 2) determining context of the handwriting input by evaluating a location of the one or more strokes of the handwriting input relative to the displayed text of the previously processed handwriting content, 3) comparing geometric features of the one or more strokes of the handwriting input with respect to the displayed text of the previously processed handwriting content, and 4) considering a language of the displayed text of the previously processed handwriting content; and	When a scribble gesture is made over a word, Goodnotes 6 determines whether the gesture editing gestures for enacting control on the displayed text of the previously processed handwriting content. It does so by (1) determining characteristics of the handwriting input (2) determining context of the handwriting input by evaluating a location of the one or more strokes of the handwriting input relative to the displayed text of the previously processed handwriting content, (3) comparing geometric features of the one or more strokes of the handwriting input with respect to the displayed text of the previously processed handwriting content, and (4) considering a language of the displayed text of the previously processed handwriting content. Specifically, Goodnotes 6 determines that the handwriting input is a scribble in a location over previously processed handwriting content and further by comparing the scribble with respect to the displayed text and by considering the language of the displayed text of the previously processed handwriting input.

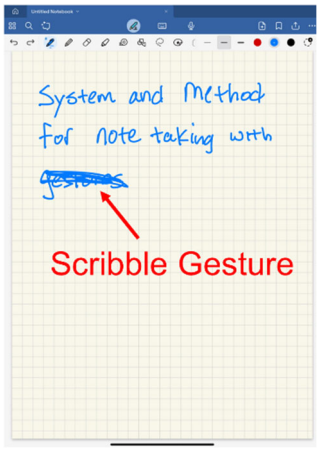
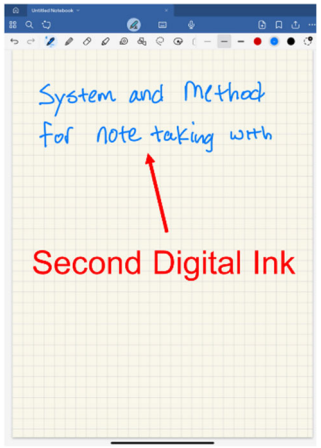
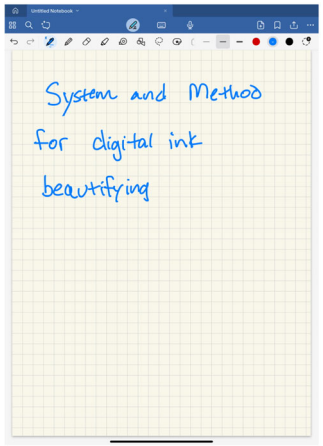
	 **Figure 2**
cause display of, on the display interface, second digital ink in accordance with the handwriting input.	When Goodnotes 6 determines that the input is a scribble, it deletes the underlying word and causes display of, on the display interface, second digital ink in accordance with the remaining handwriting input.  **Figure 3**

EXHIBIT R

U.S. Patent No. 10,410,316

Claim Language	Infringement Read
1. A system for providing editing of digital ink on computing devices, each computing device comprising a processor and at least one non-transitory computer readable medium for recognizing handwriting input under control of the processor, the at least one non-transitory computer readable medium configured to:	Goodnotes 6 is a digital note taking app that can be used on computing device with iOS, Windows, and Android operating systems. The computing devices on which Goodnotes 6 is used include a processor and a non-transitory computer readable medium (Goodnotes app) for recognizing handwriting input under control of the processor. The computer readable medium is configured to perform the actions below. *See e.g.*: https://www.goodnotes.com/pricing; https://www.goodnotes.com/features; https://www.youtube.com/watch?v=hNWQKI8FzJs (at 9:10)
cause display of, on a display interface of a computing device, digital ink in accordance with the handwriting input;	Goodnotes 6 causes display of, on a display interface of a computing device, digital ink in accordance with the handwriting input.

	 **Figure 1**
determine, from segmentation information of the handwriting input determined through the recognition of the handwriting input, ranges of elements of the digital ink;	Goodnotes 6 causes display of, on a display interface of a computing device, digital ink in accordance with the handwriting input. This is demonstrated by the fact that the app allows handwriting to be pasted in the middle of a sentence or paragraph. As shown in the following elements, for this to happen, Goodnotes 6 determines from segmentation information of the handwriting input determined through the recognition of the handwriting input, ranges of elements of the digital ink (*i.e.*, the beginning and end of handwritten text words).
cause display of, on the display interface, an input cursor in relation to the determined digital ink element ranges; and	Goodnotes 6 causes display of, on the display interface, an input cursor in relation to the determined digital ink element ranges. As shown below, the input cursor is placed in relation to the displayed digital ink elements (in between “for” and “digital”).

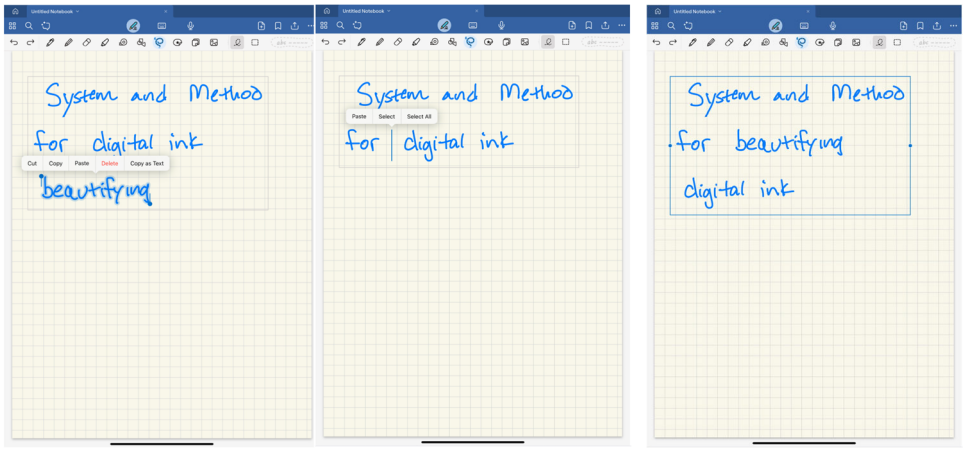
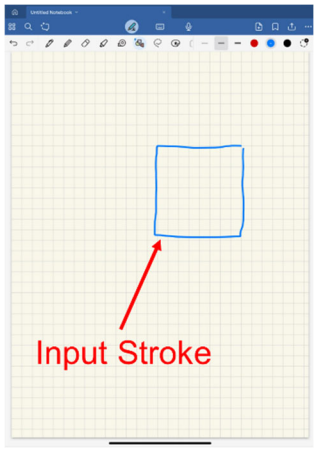
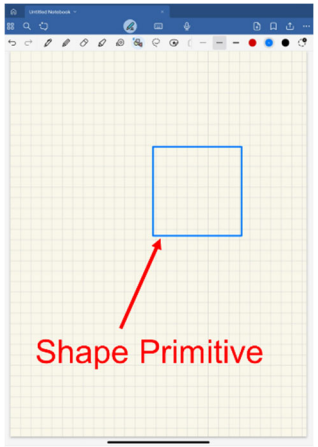
	 **Figure 2**
cause said display of the input cursor within a region of overlap of ranges of elements of the digital ink determined in the step of determining ranges of elements of the digital ink.	Goodnotes 6 causes the display of the input cursor within a region of overlap of ranges of elements of the digital ink determined when determining ranges of elements of the digital ink. As shown above, the input cursor is placed in between words of handwritten text.

EXHIBIT S

U.S. Patent No. 10,181,076

Claim Language	Infringement Read
1. A non-transitory computer readable medium having a computer readable program code embodied therein, said computer readable program code adapted to be executed to implement a method for providing geometric shape recognition for at least one input stroke, said method comprising:	Goodnotes 6 is a digital note taking app that can be used on computing device with iOS, Windows, and Android operating systems. The Goodnotes app is a computer readable program code adapted to be executed to implement a method for providing geometric shape recognition for at least one input stroke. *See e.g.*: https://www.goodnotes.com/pricing; https://www.goodnotes.com/features; https://support.goodnotes.com/hc/en-us/articles/360000113596-Draw-perfect-lines-curves-and-shapes (draw perfect shapes)
recognizing at least one shape primitive that corresponds to at least one input stroke;	Goodnotes 6 recognizes at least one shape primitive that corresponds to at least one input stroke. As shown in the images below, Goodnotes 6 recognized the shape primitive as a square that corresponds to at least one input stroke.

	 **Figure 1**  **Figure 2**
defining the at least one shape primitive as a plurality of vector components;	Goodnotes 6 defines the shape as a plurality of vector components. As shown in the image below, the vector components are the line segments between the dots.

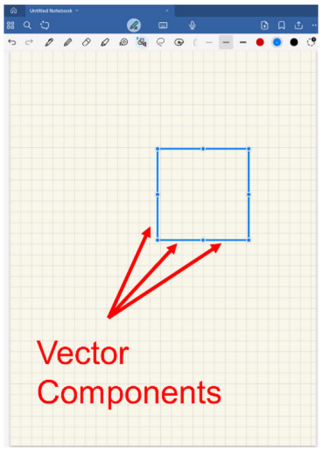
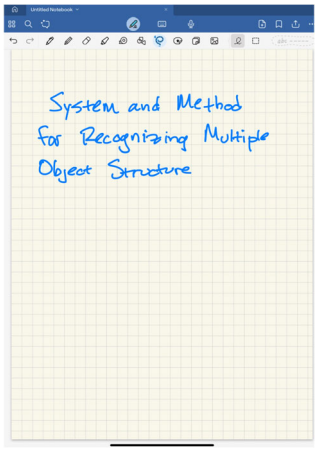
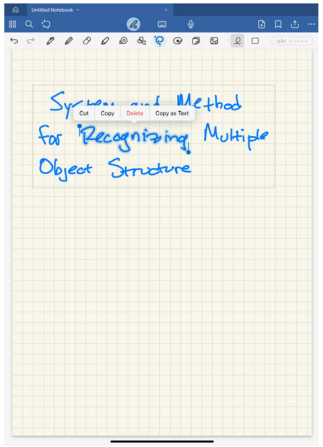
	 **Figure 1**
applying at least one vector component level constraint to the vector components, the at least one constraint geometrically linking at least two of the vector components; and	Goodnotes 6 applies at least one vector component level constraint to the vector components, the at least one constraint geometrically linking at least two of the vector components. As can be seen above the, the constraint geometrically linking at least two of the vector components (such as at a corner). This occurs even in situations where the handwriting input may not connect two lines (as seen in **Figure 1**), but the shape still includes a constraint that connects the lines (as seen in **Figure 2**).
providing an output based on the application of the at least one constraint to the vector components.	Goodnotes 6 provides an output based on the application of the at least one constraint to the vector components. This output is depicted above in **Figure 2)** which depicts the complete shape.

EXHIBIT T

U.S. Patent No. 10,013,603

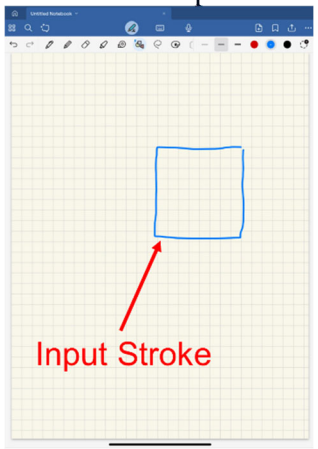
Claim Language	Infringement Read
1. A system for recognizing an arrangement of multiple objects on computing devices, each computing device comprising a processor, a memory and at least one non-transitory computer readable medium for recognizing handwriting input under control of the processor, the at least one non-transitory computer readable medium configured to:	Goodnotes 6 is a digital note taking app that can be used on computing device with iOS, Windows, and Android operating systems. The app can recognize an arrangement of multiple objects. The computing devices on which Goodnotes 6 is used include a processor, a memory, and a computer readable medium (Goodnotes app) for recognizing handwriting notetaking input under control of the processor. *See e.g.*: https://www.goodnotes.com/pricing; https://www.goodnotes.com/features; https://support.goodnotes.com/hc/en-us/articles/10779390143247-Select-content-with-the-Lasso-Tool#h_01J857YQNCQ1NJ44896ZEFZ0KA (edit mode with lasso tool)
determine, with the medium, at least one geometrical relationship between a plurality of recognized elements of the handwriting input; and	Goodnotes 6 determines at least one geometrical relationship between a plurality of recognized elements of the handwriting input. As shown below, Goodnotes 6 includes an "Edit Mode" that allows the user to select an entire section of handwriting input (using the lasso tool) and then further press on a word to select only that word for editing (i.e., the word "Recognizing"). For the app to perform this functionality, the app determines a geometrical relationship between the plurality of recognized elements of the handwriting input (i.e., the several words) and allows selection of only one of those elements (i.e., the word "Recognizing") by pressing on a portion of the word.

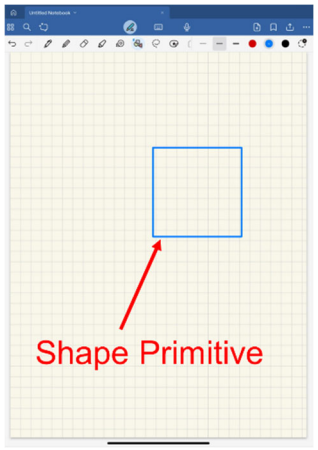
	 **Figure 1**  **Figure 2**
allocate, with the medium, the recognized elements having the at least one geometrical relationship therebetween to corresponding	As illustrated above, Goodnotes 6 allocates the recognized element having the at least one geometrical relationship therebetween to corresponding positions of an arrangement of the recognized elements (i.e., the word “Recognizing”) by at least

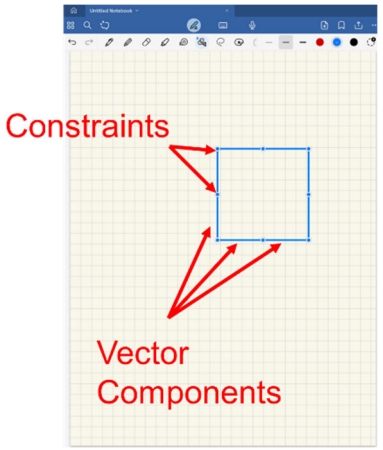
positions of an arrangement of the recognized elements at least partially determined by boundaries of the recognized elements.	partially determined by boundaries of the recognized elements (i.e., the light blue colors surrounding each letter in the word “Recognizing”).

EXHIBIT U

U.S. Patent No. 9,489,572

Claim Language	Infringement Read
1. A non-transitory computer readable medium having a computer readable program code embodied therein, said computer readable program code adapted to be executed to implement a method for providing geometric shape recognition for at least one input stroke, said method comprising:	Goodnotes 6 is a digital note taking app that can be used on computing device with iOS, Windows, and Android operating systems. The Goodnotes app is computer readable program code adapted to be executed to implement a method for providing geometric shape recognition for at least one input stroke. *See e.g.*: https://www.goodnotes.com/pricing; https://www.goodnotes.com/features; https://support.goodnotes.com/hc/en-us/articles/360000113596-Draw-perfect-lines-curves-and-shapes (draw perfect shapes)
recognizing at least one primitive that corresponds to at least one input stroke;	Goodnotes 6 recognizes at least one shape primitive that corresponds to at least one input stroke. As shown in the images below, Goodnotes 6 recognized the primitive as a square that corresponds to at least one input stroke.  **Figure 1**

	 **Figure 2**
applying at least one constraint to the at least one primitive, wherein the at least one constraint and at least one primitive are applied on a vector component level; and	Goodnotes 6 applies at least one constraint to the at least one primitive. The constraint and one primitive are further applied on a vector component level. As seen in the image below, Goodnotes 6 applies at least one constraint (i.e., depicted by "dots") to the primitive (i.e., square shape) on a vector component level (i.e., segments connected by the "dots").

	 **Figure 3**
providing an output based on the application of the at least one constraint to the at least one primitive,	As shown in the images above, Goodnotes 6 provides an output on the screen based on the application of the constraints to the primitive.
wherein if the at least one constraint comprises multiple constraints, the multiple constraints are applied to the at least one primitive based on a priority order, wherein the priority order comprises constraints maintaining the at least one primitive's integrity, constraints implicit in the at least one primitive, and constraints defined by a gesture.	As shown above, if the constraint comprises multiple constraints, the multiple constraints are applied to the at least one primitive based on a priority order, wherein the priority order comprises constraints maintaining the at least one primitive's integrity, constraints implicit in the at least one primitive, and constraints defined by a gesture. For instance, as shown above, when the handwritten input stroke is close to but does not connect the ends of the primitive (*see* **Figure 1**, Goodnotes 6 prioritizes maintaining the primitive's integrity by connecting the line segments to create the square shape (*see* **Figure 2**), but also includes constraints implicit in the primitive (e.g., the constraints in the middle of the side of the square), and constraints defined by a gesture (e.g., moving constraints via the user's finger to move portions of the primitive to create a new shape).

JS 44 (Rev. 03/24)

CIVIL COVER SHEET

The JS 44 civil cover sheet and the information contained herein neither replace nor supplement the filing and service of pleadings or other papers as required by law, except as provided by local rules of court. This form, approved by the Judicial Conference of the United States in September 1974, is required for the use of the Clerk of Court for the purpose of initiating the civil docket sheet. *(SEE INSTRUCTIONS ON NEXT PAGE OF THIS FORM.)*

I. (a) PLAINTIFFS

MYSCRIPT SAS

(b) County of Residence of First Listed Plaintiff _____
(EXCEPT IN U.S. PLAINTIFF CASES)

(c) Attorneys *(Firm Name, Address, and Telephone Number)*

Deron R. Dacus; THE DACUS FIRM, P.C.; 821 ESE Loop 323, Suite 430, Tyler, TX 75701; (903) 705-7233

DEFENDANTS

GOODNOTES LIMITED

County of Residence of First Listed Defendant _____
(IN U.S. PLAINTIFF CASES ONLY)

NOTE: IN LAND CONDEMNATION CASES, USE THE LOCATION OF THE TRACT OF LAND INVOLVED.

Attorneys *(If Known)*

II. BASIS OF JURISDICTION *(Place an "X" in One Box Only)*

- ☐ 1 U.S. Government Plaintiff
- ☒ 3 Federal Question *(U.S. Government Not a Party)*
- ☐ 2 U.S. Government Defendant
- ☐ 4 Diversity *(Indicate Citizenship of Parties in Item III)*

III. CITIZENSHIP OF PRINCIPAL PARTIES *(Place an "X" in One Box for Plaintiff and One Box for Defendant)*

(For Diversity Cases Only)

	PTF	DEF		PTF	DEF
Citizen of This State	☐ 1	☐ 1	Incorporated *or* Principal Place of Business In This State	☐ 4	☐ 4
Citizen of Another State	☐ 2	☐ 2	Incorporated *and* Principal Place of Business In Another State	☐ 5	☐ 5
Citizen or Subject of a Foreign Country	☐ 3	☐ 3	Foreign Nation	☐ 6	☐ 6

IV. NATURE OF SUIT *(Place an "X" in One Box Only)*

Click here for: Nature of Suit Code Descriptions.

CONTRACT
- ☐ 110 Insurance
- ☐ 120 Marine
- ☐ 130 Miller Act
- ☐ 140 Negotiable Instrument
- ☐ 150 Recovery of Overpayment & Enforcement of Judgment
- ☐ 151 Medicare Act
- ☐ 152 Recovery of Defaulted Student Loans (Excludes Veterans)
- ☐ 153 Recovery of Overpayment of Veteran's Benefits
- ☐ 160 Stockholders' Suits
- ☐ 190 Other Contract
- ☐ 195 Contract Product Liability
- ☐ 196 Franchise

REAL PROPERTY
- ☐ 210 Land Condemnation
- ☐ 220 Foreclosure
- ☐ 230 Rent Lease & Ejectment
- ☐ 240 Torts to Land
- ☐ 245 Tort Product Liability
- ☐ 290 All Other Real Property

TORTS

PERSONAL INJURY
- ☐ 310 Airplane
- ☐ 315 Airplane Product Liability
- ☐ 320 Assault, Libel & Slander
- ☐ 330 Federal Employers' Liability
- ☐ 340 Marine
- ☐ 345 Marine Product Liability
- ☐ 350 Motor Vehicle
- ☐ 355 Motor Vehicle Product Liability
- ☐ 360 Other Personal Injury
- ☐ 362 Personal Injury - Medical Malpractice

PERSONAL INJURY
- ☐ 365 Personal Injury - Product Liability
- ☐ 367 Health Care/ Pharmaceutical Personal Injury Product Liability
- ☐ 368 Asbestos Personal Injury Product Liability

PERSONAL PROPERTY
- ☐ 370 Other Fraud
- ☐ 371 Truth in Lending
- ☐ 380 Other Personal Property Damage
- ☐ 385 Property Damage Product Liability

CIVIL RIGHTS
- ☐ 440 Other Civil Rights
- ☐ 441 Voting
- ☐ 442 Employment
- ☐ 443 Housing/ Accommodations
- ☐ 445 Amer. w/Disabilities - Employment
- ☐ 446 Amer. w/Disabilities - Other
- ☐ 448 Education

PRISONER PETITIONS

Habeas Corpus:
- ☐ 463 Alien Detainee
- ☐ 510 Motions to Vacate Sentence
- ☐ 530 General
- ☐ 535 Death Penalty

Other:
- ☐ 540 Mandamus & Other
- ☐ 550 Civil Rights
- ☐ 555 Prison Condition
- ☐ 560 Civil Detainee - Conditions of Confinement

FORFEITURE/PENALTY
- ☐ 625 Drug Related Seizure of Property 21 USC 881
- ☐ 690 Other

LABOR
- ☐ 710 Fair Labor Standards Act
- ☐ 720 Labor/Management Relations
- ☐ 740 Railway Labor Act
- ☐ 751 Family and Medical Leave Act
- ☐ 790 Other Labor Litigation
- ☐ 791 Employee Retirement Income Security Act

IMMIGRATION
- ☐ 462 Naturalization Application
- ☐ 465 Other Immigration Actions

BANKRUPTCY
- ☐ 422 Appeal 28 USC 158
- ☐ 423 Withdrawal 28 USC 157

INTELLECTUAL PROPERTY RIGHTS
- ☐ 820 Copyrights
- ☒ 830 Patent
- ☐ 835 Patent - Abbreviated New Drug Application
- ☐ 840 Trademark
- ☐ 880 Defend Trade Secrets Act of 2016

SOCIAL SECURITY
- ☐ 861 HIA (1395ff)
- ☐ 862 Black Lung (923)
- ☐ 863 DIWC/DIWW (405(g))
- ☐ 864 SSID Title XVI
- ☐ 865 RSI (405(g))

FEDERAL TAX SUITS
- ☐ 870 Taxes (U.S. Plaintiff or Defendant)
- ☐ 871 IRS—Third Party 26 USC 7609

OTHER STATUTES
- ☐ 375 False Claims Act
- ☐ 376 Qui Tam (31 USC 3729(a))
- ☐ 400 State Reapportionment
- ☐ 410 Antitrust
- ☐ 430 Banks and Banking
- ☐ 450 Commerce
- ☐ 460 Deportation
- ☐ 470 Racketeer Influenced and Corrupt Organizations
- ☐ 480 Consumer Credit (15 USC 1681 or 1692)
- ☐ 485 Telephone Consumer Protection Act
- ☐ 490 Cable/Sat TV
- ☐ 850 Securities/Commodities/ Exchange
- ☐ 890 Other Statutory Actions
- ☐ 891 Agricultural Acts
- ☐ 893 Environmental Matters
- ☐ 895 Freedom of Information Act
- ☐ 896 Arbitration
- ☐ 899 Administrative Procedure Act/Review or Appeal of Agency Decision
- ☐ 950 Constitutionality of State Statutes

V. ORIGIN *(Place an "X" in One Box Only)*

☒ 1 Original Proceeding ☐ 2 Removed from State Court ☐ 3 Remanded from Appellate Court ☐ 4 Reinstated or Reopened ☐ 5 Transferred from Another District *(specify)* ☐ 6 Multidistrict Litigation - Transfer ☐ 8 Multidistrict Litigation - Direct File

VI. CAUSE OF ACTION

Cite the U.S. Civil Statute under which you are filing ***(Do not cite jurisdictional statutes unless diversity)***:
35 U.S.C. § 271

Brief description of cause:
Patent Infringement

VII. REQUESTED IN COMPLAINT:

☐ CHECK IF THIS IS A **CLASS ACTION** UNDER RULE 23, F.R.Cv.P.

DEMAND $

CHECK YES only if demanded in complaint:

JURY DEMAND: ☒ Yes ☐ No

VIII. RELATED CASE(S) IF ANY

(See instructions): JUDGE _____ DOCKET NUMBER _____

DATE
May 15, 2025

SIGNATURE OF ATTORNEY OF RECORD
Deron R. Dacus

FOR OFFICE USE ONLY

RECEIPT # _____ AMOUNT _____ APPLYING IFP _____ JUDGE _____ MAG. JUDGE _____

INSTRUCTIONS FOR ATTORNEYS COMPLETING CIVIL COVER SHEET FORM JS 44

Authority For Civil Cover Sheet

The JS 44 civil cover sheet and the information contained herein neither replaces nor supplements the filings and service of pleading or other papers as required by law, except as provided by local rules of court. This form, approved by the Judicial Conference of the United States in September 1974, is required for the use of the Clerk of Court for the purpose of initiating the civil docket sheet. Consequently, a civil cover sheet is submitted to the Clerk of Court for each civil complaint filed. The attorney filing a case should complete the form as follows:

I.(a) **Plaintiffs-Defendants.** Enter names (last, first, middle initial) of plaintiff and defendant. If the plaintiff or defendant is a government agency, use only the full name or standard abbreviations. If the plaintiff or defendant is an official within a government agency, identify first the agency and then the official, giving both name and title.

(b) **County of Residence.** For each civil case filed, except U.S. plaintiff cases, enter the name of the county where the first listed plaintiff resides at the time of filing. In U.S. plaintiff cases, enter the name of the county in which the first listed defendant resides at the time of filing. (NOTE: In land condemnation cases, the county of residence of the "defendant" is the location of the tract of land involved.)

(c) **Attorneys.** Enter the firm name, address, telephone number, and attorney of record. If there are several attorneys, list them on an attachment, noting in this section "(see attachment)".

II. **Jurisdiction.** The basis of jurisdiction is set forth under Rule 8(a), F.R.Cv.P., which requires that jurisdictions be shown in pleadings. Place an "X" in one of the boxes. If there is more than one basis of jurisdiction, precedence is given in the order shown below.
United States plaintiff. (1) Jurisdiction based on 28 U.S.C. 1345 and 1348. Suits by agencies and officers of the United States are included here.
United States defendant. (2) When the plaintiff is suing the United States, its officers or agencies, place an "X" in this box.
Federal question. (3) This refers to suits under 28 U.S.C. 1331, where jurisdiction arises under the Constitution of the United States, an amendment to the Constitution, an act of Congress or a treaty of the United States. In cases where the U.S. is a party, the U.S. plaintiff or defendant code takes precedence, and box 1 or 2 should be marked.
Diversity of citizenship. (4) This refers to suits under 28 U.S.C. 1332, where parties are citizens of different states. When Box 4 is checked, the citizenship of the different parties must be checked. (See Section III below**; NOTE: federal question actions take precedence over diversity cases.**)

III. **Residence (citizenship) of Principal Parties.** This section of the JS 44 is to be completed if diversity of citizenship was indicated above. Mark this section for each principal party.

IV. **Nature of Suit.** Place an "X" in the appropriate box. If there are multiple nature of suit codes associated with the case, pick the nature of suit code that is most applicable. Click here for: Nature of Suit Code Descriptions.

V. **Origin.** Place an "X" in one of the seven boxes.
Original Proceedings. (1) Cases which originate in the United States district courts.
Removed from State Court. (2) Proceedings initiated in state courts may be removed to the district courts under Title 28 U.S.C., Section 1441.
Remanded from Appellate Court. (3) Check this box for cases remanded to the district court for further action. Use the date of remand as the filing date.
Reinstated or Reopened. (4) Check this box for cases reinstated or reopened in the district court. Use the reopening date as the filing date.
Transferred from Another District. (5) For cases transferred under Title 28 U.S.C. Section 1404(a). Do not use this for within district transfers or multidistrict litigation transfers.
Multidistrict Litigation – Transfer. (6) Check this box when a multidistrict case is transferred into the district under authority of Title 28 U.S.C. Section 1407.
Multidistrict Litigation – Direct File. (8) Check this box when a multidistrict case is filed in the same district as the Master MDL docket.
PLEASE NOTE THAT THERE IS NOT AN ORIGIN CODE 7. Origin Code 7 was used for historical records and is no longer relevant due to changes in statute.

VI. **Cause of Action.** Report the civil statute directly related to the cause of action and give a brief description of the cause. **Do not cite jurisdictional statutes unless diversity.** Example: U.S. Civil Statute: 47 USC 553 Brief Description: Unauthorized reception of cable service.

VII. **Requested in Complaint.** Class Action. Place an "X" in this box if you are filing a class action under Rule 23, F.R.Cv.P.
Demand. In this space enter the actual dollar amount being demanded or indicate other demand, such as a preliminary injunction.
Jury Demand. Check the appropriate box to indicate whether or not a jury is being demanded.

VIII. **Related Cases.** This section of the JS 44 is used to reference related cases, if any. If there are related cases, insert the docket numbers and the corresponding judge names for such cases.

Date and Attorney Signature. Date and sign the civil cover sheet.