US010558342B2

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

(10) **Patent No.: US 10,558,342 B2**
(45) **Date of Patent: *Feb. 11, 2020**

(54) **AUTOMATED PIXEL SHIFTING WITHIN A DIGITAL IMAGE**

(71) Applicants: **Troy Plota**, Kihei, HI (US); **Sascha Connelly**, Kihei, HI (US)

(72) Inventors: **Troy Plota**, Kihei, HI (US); **Sascha Connelly**, Kihei, HI (US)

(*) 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.: **15/660,677**

(22) Filed: **Jul. 26, 2017**

(65) **Prior Publication Data**

US 2018/0032236 A1 Feb. 1, 2018

Related U.S. Application Data

(60) Provisional application No. 62/514,703, filed on Jun. 2, 2017, provisional application No. 62/368,017, filed on Jul. 28, 2016.

(51) **Int. Cl.**

G09G 5/00	(2006.01)
G06F 3/0484	(2013.01)
G06F 3/0481	(2013.01)
G06T 11/60	(2006.01)
G06T 13/80	(2011.01)

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

CPC ***G06F 3/04845*** (2013.01); ***G06F 3/0484*** (2013.01); ***G06F 3/04812*** (2013.01); ***G06T 11/60*** (2013.01); ***G06T 13/80*** (2013.01)

(58) **Field of Classification Search**

CPC G06T 11/60; G06T 19/00; G06T 11/00; G06F 9/4443; G06F 3/0481

USPC 345/619

See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

2007/0019007	A1	1/2007	Jung	
2009/0189912	A1	7/2009	Holtman	
2010/0134505	A1	6/2010	Andersen	
2011/0273583	A1	11/2011	Yoshino	
2015/0022518	A1	1/2015	Takeshita	
2015/0095882	A1	4/2015	Jaeger et al.	
2015/0262327	A1*	9/2015	Ohta	G06T 3/0093 345/646
2017/0140514	A1*	5/2017	Amirghodsi	G06T 5/50

OTHER PUBLICATIONS

SCICompanion, Palette Cycling, Mar. 6, 2016. [Online], Retrieved on Sep. 21, 2018, Retrieved from the Internet <URL: http://scicompanion.com/Documentation/palettecycling.html>.*

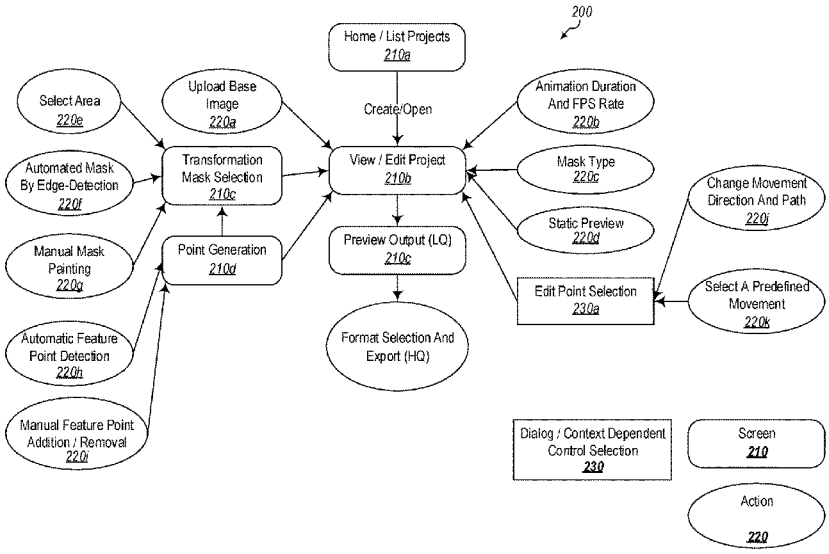
(Continued)

Primary Examiner — Gordon G Liu
(74) *Attorney, Agent, or Firm* — Workman Nydegger

(57) **ABSTRACT**

Automating the shifting of pixels within a digital image comprises a processor receiving an indication of a starting point through a user interface. The starting point is received through a user selection of a particular portion of the digital image. Additionally, the processor receives, through the user interface, a direction associated with the starting point. The processor also selects a set of pixels extending in the direction away from the starting point. Further, the processor shifts the set of pixels in the first direction. Shifting the set of pixels comprises rendering and re-rendering in a loop the set of pixels being shifted.

20 Claims, 9 Drawing Sheets



(56) **References Cited**

OTHER PUBLICATIONS

International Search Report and Written Opinion for application No. PCT/US2017/044256 dated Oct. 6, 2017.
U.S. Appl. No. 16/102,055, dated Oct. 2, 2018, Office Action.
U.S. Appl. No. 16/102,055, dated Jan. 18, 2019, Final Office Action.
U.S. Appl. No. 16/162,206, dated Oct. 16, 2018, Plota et al.
U.S. Appl. No. 16/162,206, dated Dec. 26, 2019, Office Action.
U.S. Appl. No. 16/102,055, dated Apr. 17, 2019, Notice of Allowance.
U.S. Appl. No. 16/162,206, dated May 3, 2019, Final Office Action.
U.S. Appl. No. 16/162,206, dated Nov. 27, 2019, Notice of Allowance.

* cited by examiner

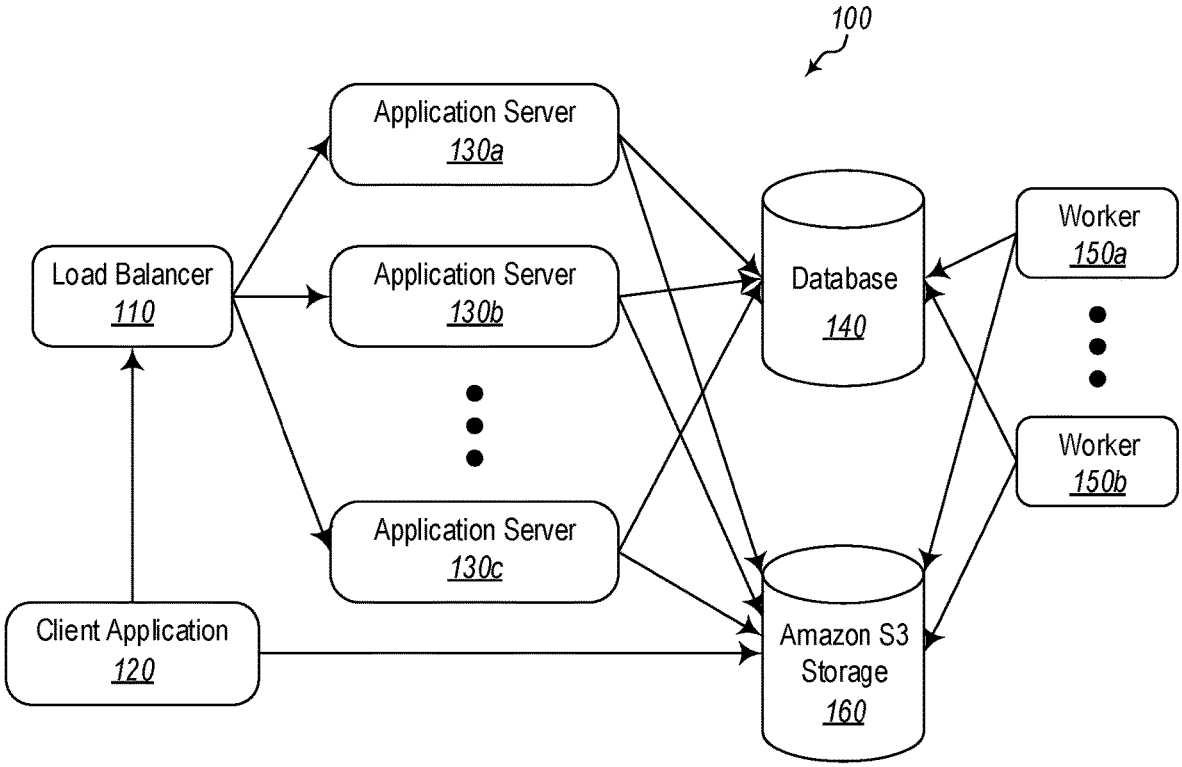

100
Application Server
130a
Load Balancer
110
Application Server
130b
Application Server
130c
Client Application
120
Database
140
Amazon S3
Storage
160
Worker
150a
Worker
150b

Fig. 1

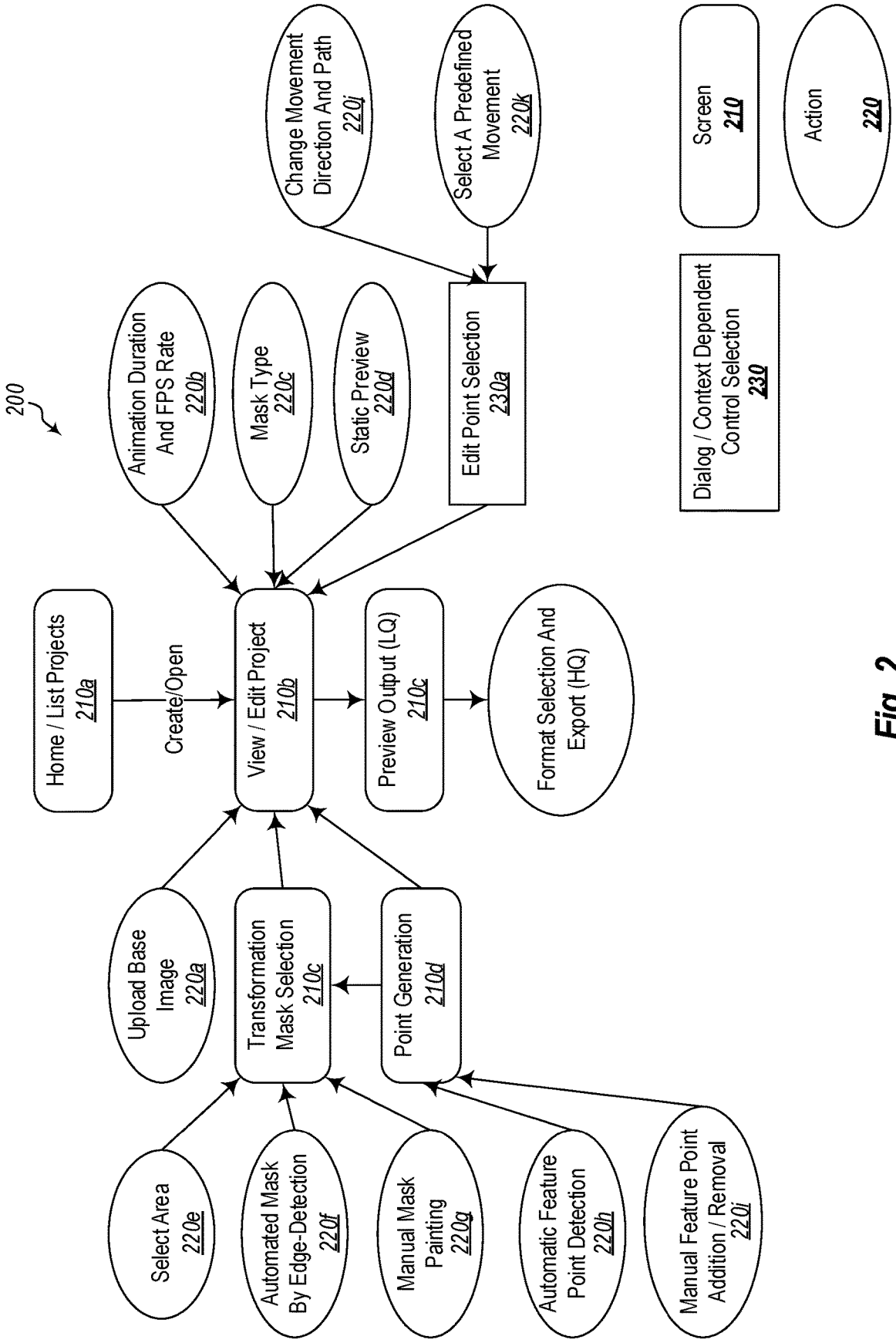

200
Home / List Projects 210a
Create/Open
View / Edit Project 210b
Preview Output (LQ) 210c
Format Selection And Export (HQ)
Animation Duration And FPS Rate 220b
Mask Type 220c
Static Preview 220d
Edit Point Selection 230a
Change Movement Direction And Path 220j
Select A Predefined Movement 220k
Upload Base Image 220a
Transformation Mask Selection 210c
Point Generation 210d
Select Area 220e
Automated Mask By Edge-Detection 220f
Manual Mask Painting 220g
Automatic Feature Point Detection 220h
Manual Feature Point Addition / Removal 220i
Screen 210
Action 220
Dialog / Context Dependent Control Selection 230

Fig. 2

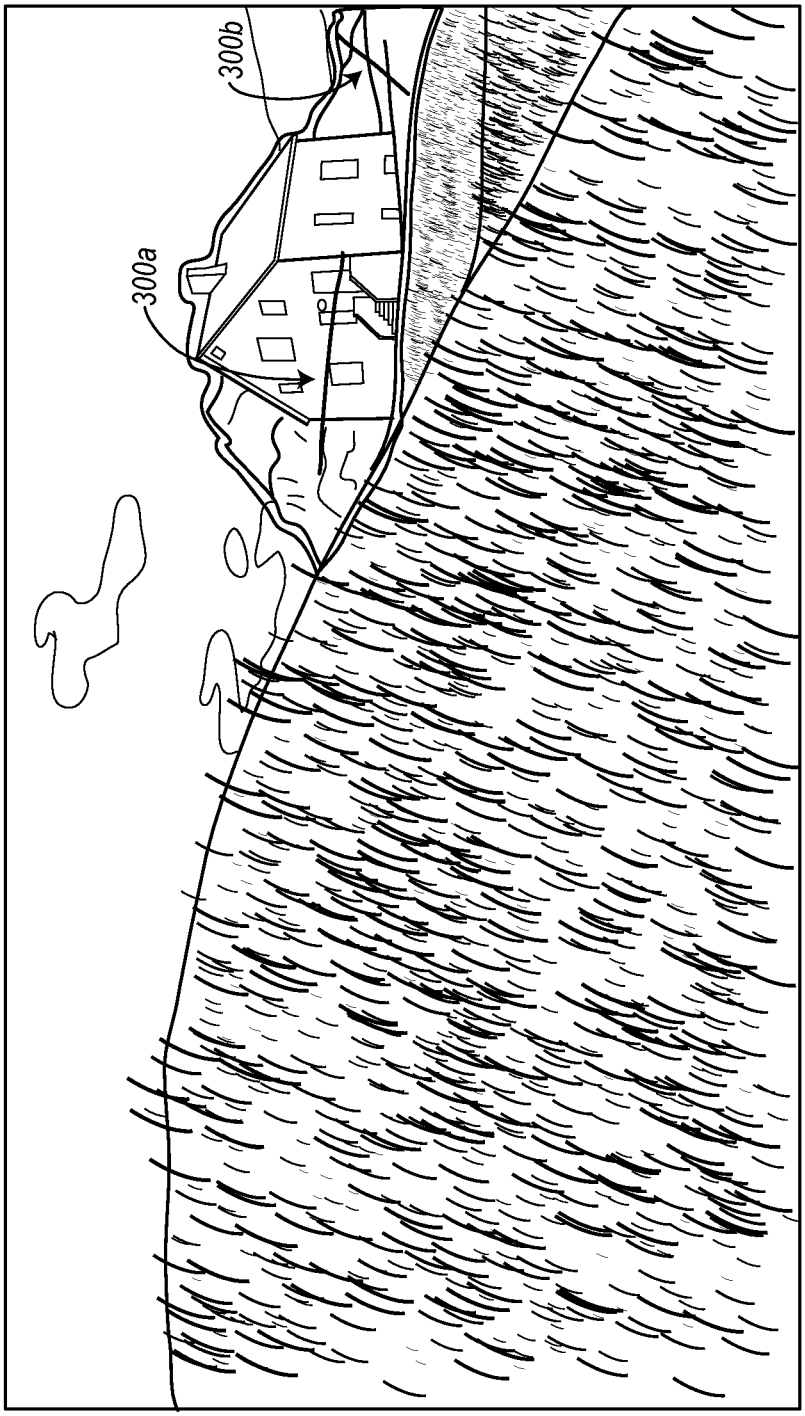


Fig. 3A

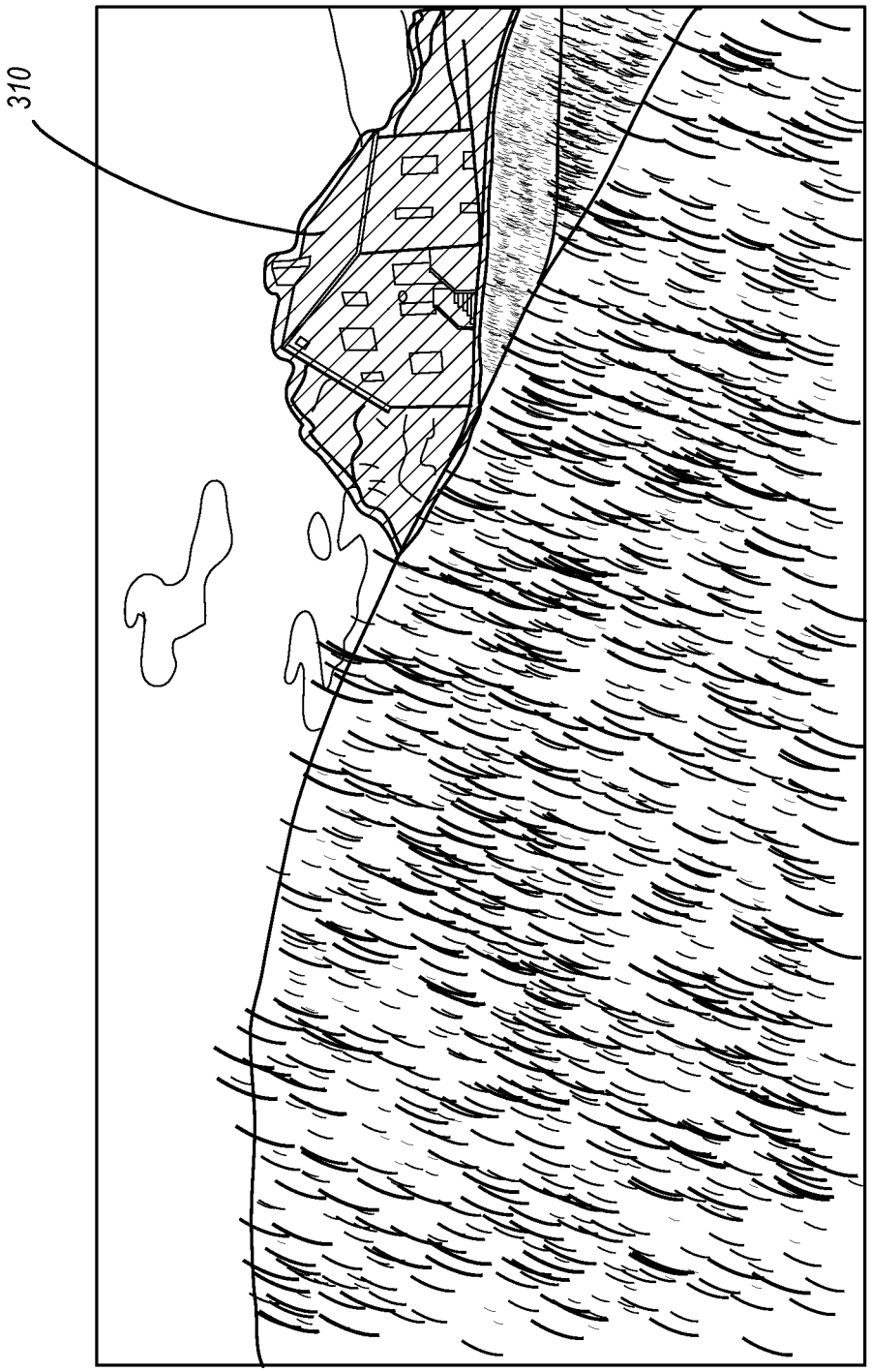


Fig. 3B

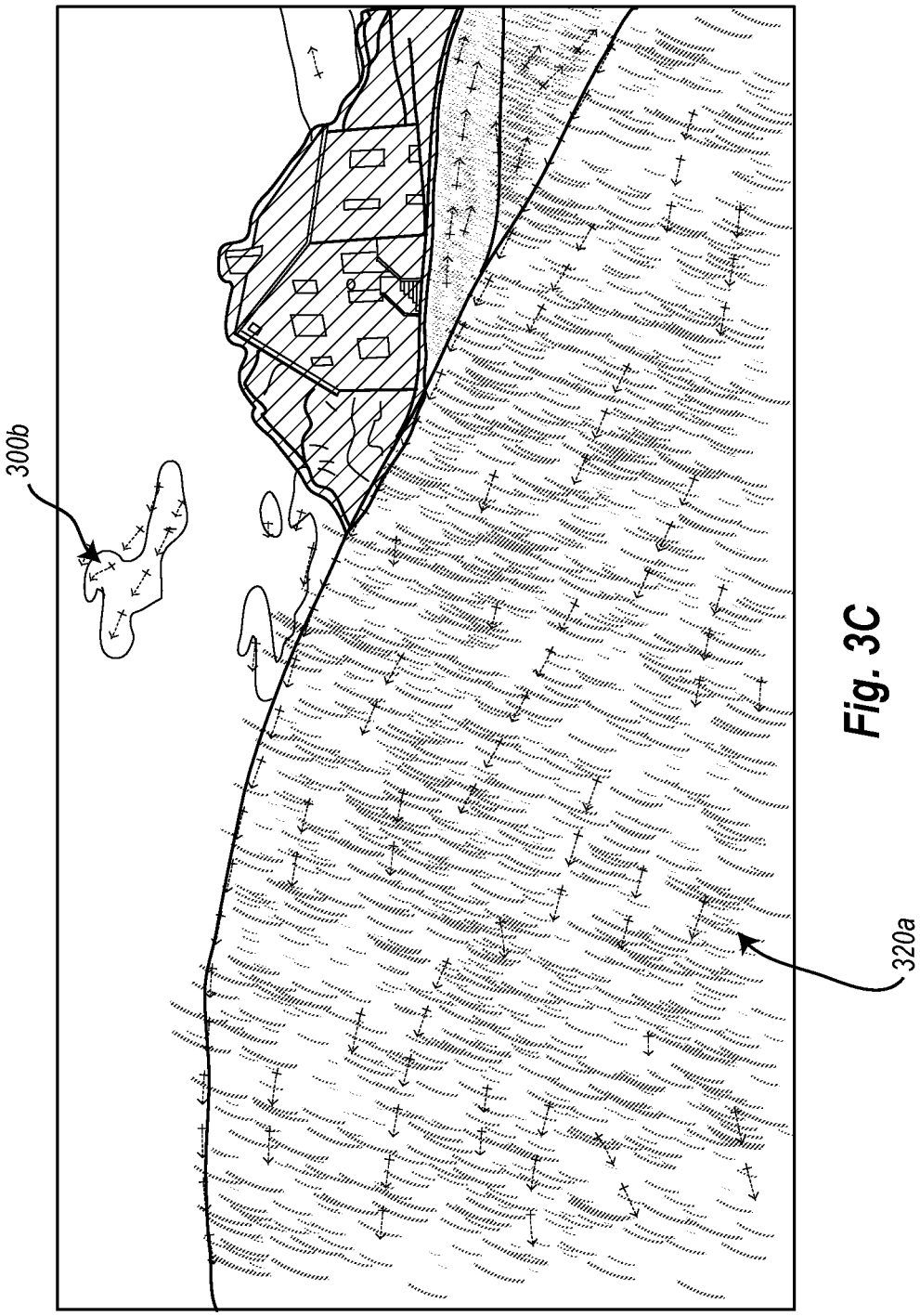


Fig. 3C

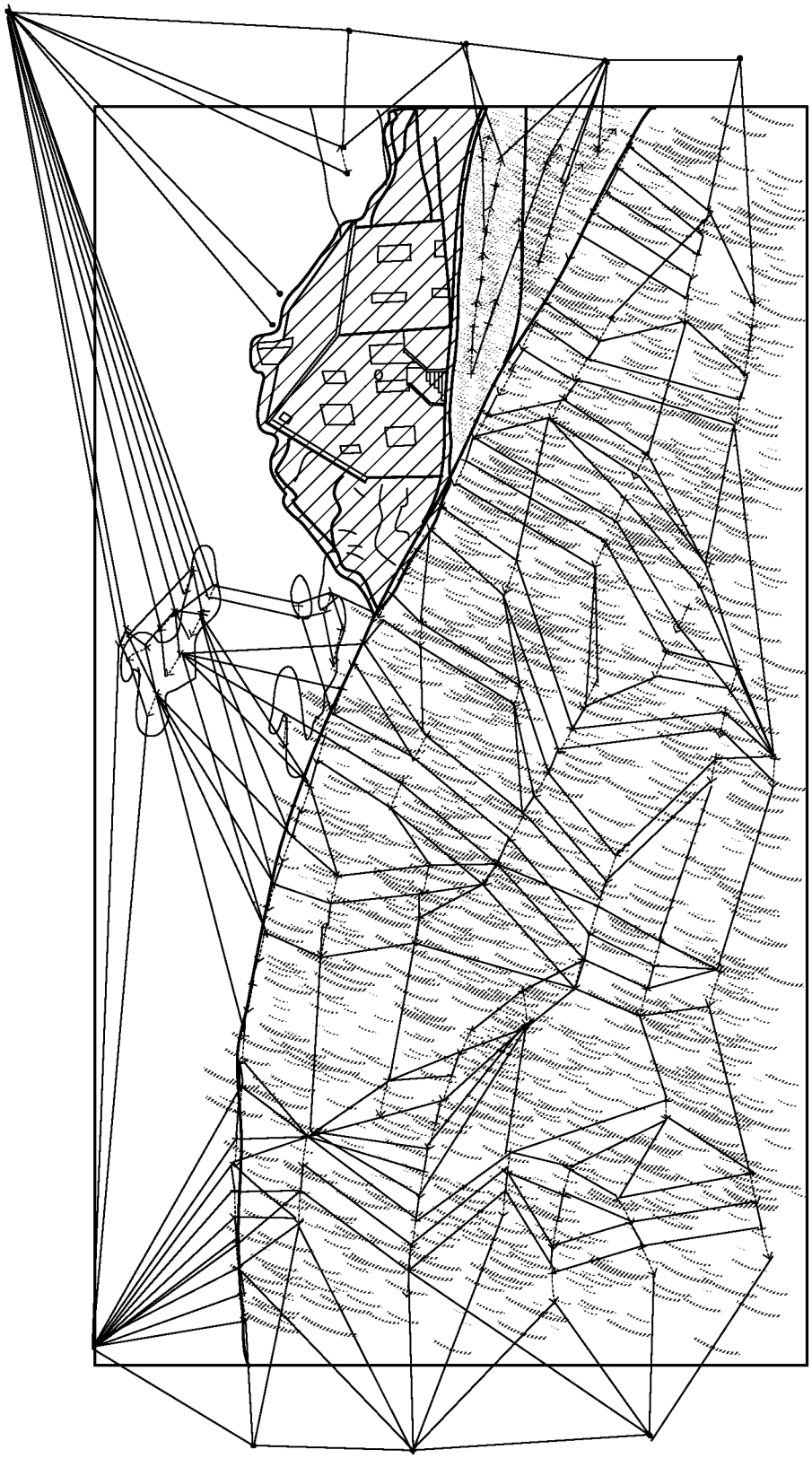


Fig. 4

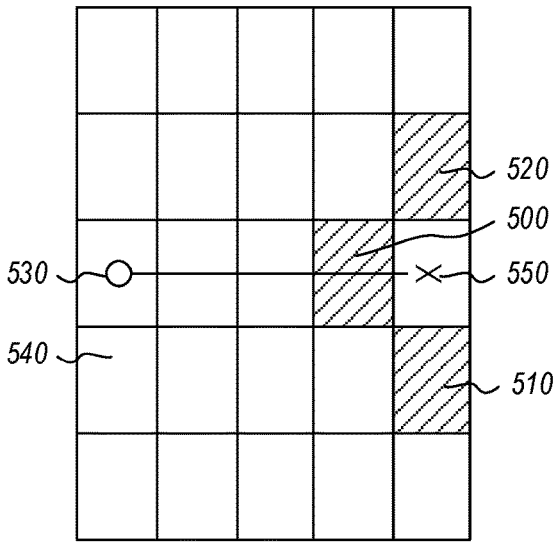


Fig. 5A

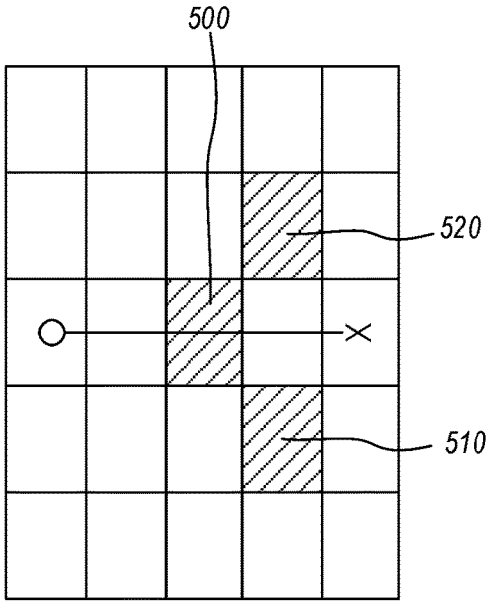


Fig. 5B

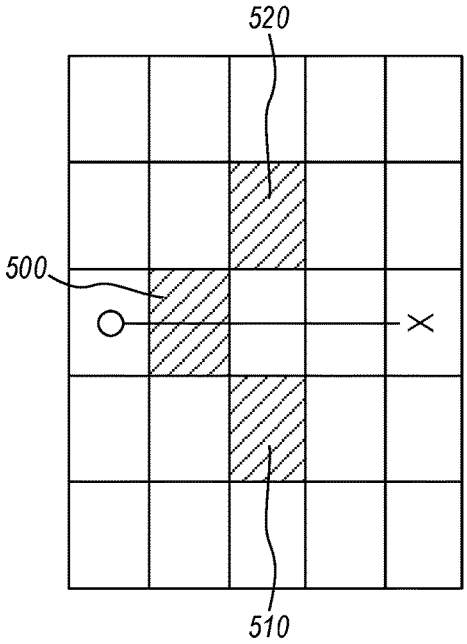


Fig. 5C

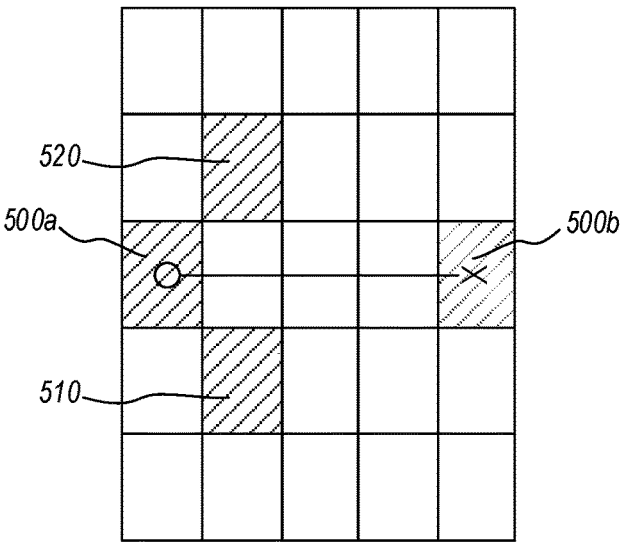


Fig. 5D

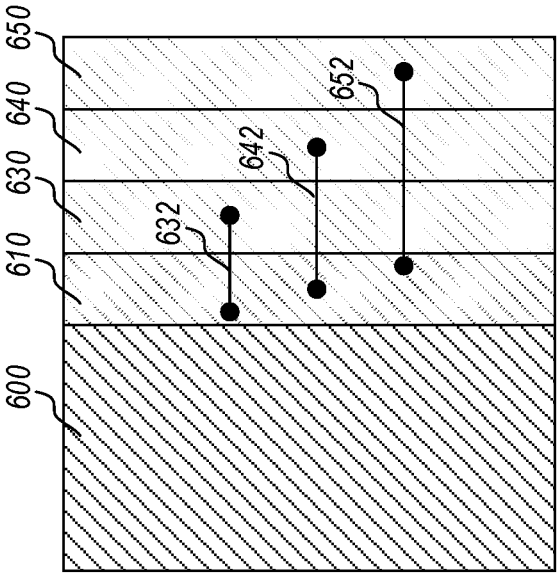


Fig. 6B

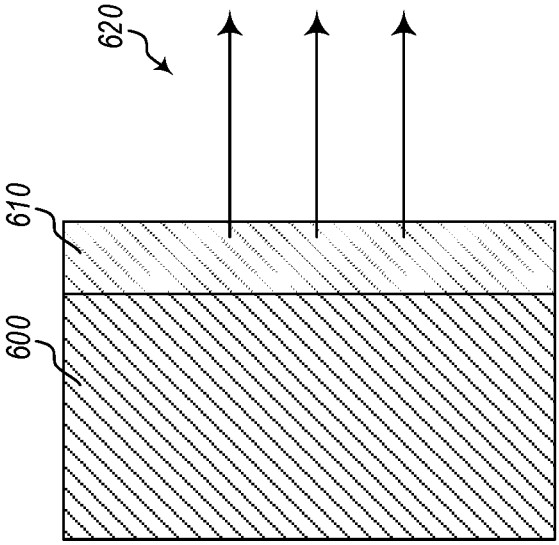


Fig. 6A

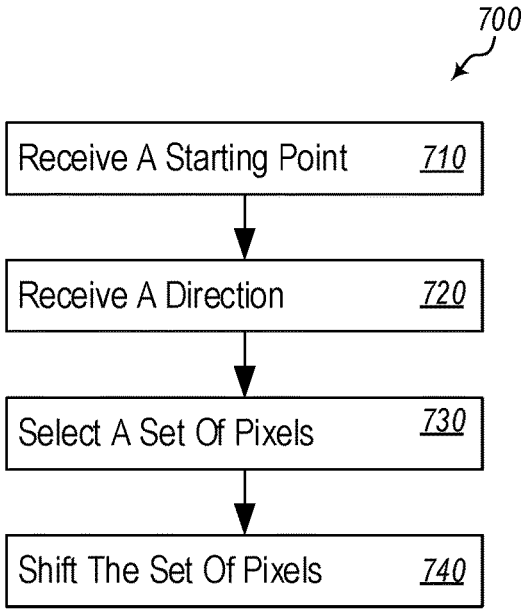

700
Receive A Starting Point 710
Receive A Direction 720
Select A Set Of Pixels 730
Shift The Set Of Pixels 740

Fig. 7

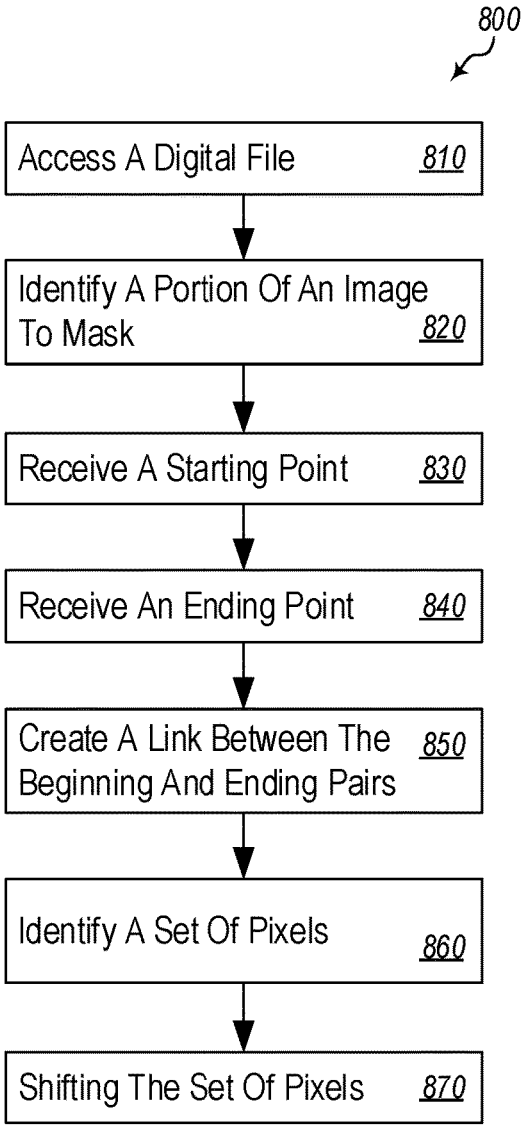

800
Access A Digital File 810
Identify A Portion Of An Image To Mask 820
Receive A Starting Point 830
Receive An Ending Point 840
Create A Link Between The Beginning And Ending Pairs 850
Identify A Set Of Pixels 860
Shifting The Set Of Pixels 870

Fig. 8

AUTOMATED PIXEL SHIFTING WITHIN A DIGITAL IMAGE

CROSS-REFERENCE TO RELATED APPLICATIONS

This application claims priority to and the benefit of U.S. Provisional Application 62/514,703 entitled "Automated Pixel Shifting Within A Digital Image", filed on Jun. 2, 2017, and U.S. Provisional Application Ser. No. 62/368,017 entitled "Automated Pixel Shifting Within A Digital Image", filed on Jul. 28, 2016. The entire contents of each of the aforementioned applications and/or patents are incorporated by reference herein in their entirety.

BACKGROUND

Computers and computing systems have affected nearly every aspect of modern living. Computers are generally involved in work, recreation, healthcare, transportation, entertainment, household management, etc.

Computers, and in particular the fairly recent boom of digital photography, have changed the entire photography industry. For example, instead of taking pictures and then waiting for film to develop before viewing the images, with modern digital cameras, high quality photographs can be captured and immediately viewed. The increase in digital photography has been accompanied with a related increase in digital image software tools.

Digital image software tools include compression algorithms that dramatically reduce the size of digital images with little or no loss of image quality. Additional, digital image software tools also include tools to edit or otherwise manipulate digital images. For example, many digital image editing tools and digital image sharing tools include filters that can be applied to images to generate desired effects and appearances within an image.

While in some cases the use of digital image tools can appear seamless and quick to an end-user, many of these tools utilize extremely complex computer-based technological solutions. For example, properly applying a filter to an image can involve complex mathematical equations necessary to consistently adjust the color of individual pixels within a digital image.

Adjusting relatively simple aspects of a digital image, such as color, provides many users with highly desirable tools for editing their digital images. Additional, tools are desirable for improving digital images, such as the ability to incorporate movement within a digital image.

The subject matter claimed herein is not limited to embodiments that solve any disadvantages or that operate only in environments such as those described above. Rather, this background is only provided to illustrate one exemplary technology area where some embodiments described herein may be practiced.

BRIEF SUMMARY

Embodiments disclosed herein comprise a computer system for automating the shifting of pixels within a digital image. The system comprises one or more processors and one or more computer-readable media having stored thereon executable instructions that when executed by the one or more processors configure the computer system to perform various steps. For example, the system accesses, from memory, a digital image file. The digital image file comprises information that corresponds to individual pixels

within the digital image. The system also identifies a particular portion of the digital image to mask. The mask prevents pixels covered by the mask from being shifted. The system also received a first starting point through a user interface. The first starting point is received through a user selection of a first beginning portion of the digital image. Additionally, the system receives a first ending point through the user interface. The first starting point is received through a user selection of a first ending portion of the user interface.

The system then creates a first link between the first starting point and the first ending point. The first link comprises a first direction extending from the first starting point to the first ending point. The first link also comprises a first length between the first starting point and the first ending point. In addition, the system identifies a first set of pixels that lie between the first starting point and the first ending point. The system shifts the first set of pixels in the first direction. Shifting the first set of pixels comprises rendering and re-rendering in a loop the first set of pixels being shifted.

Disclosed embodiments include automating the shifting of pixels within a digital image by using a processor to receive an indication of a starting point through a user interface. The starting point is received through a user selection of a particular portion of the digital image. Additionally, the processor receives, through the user interface, a direction associated with the starting point. The processor also selects a set of pixels extending in the direction away from the starting point. Further, the processor shifts the set of pixels in the first direction. Shifting the set of pixels comprises rendering and re-rendering in a loop the set of pixels being shifted.

Additional disclosed embodiments comprise a method for automating the shifting of pixels within a digital image. The method comprises receiving a first indication of a first starting point through a user interface. The first starting point is received through a user selection of a portion of the digital image. The method also comprises receiving, through the user interface, a first direction associated with the first starting point. Additionally, the method comprises selecting a first set of pixels extending in the first direction away from the first starting point. The method also comprises shifting the first set of pixels in the first direction, wherein shifting the first set of pixels comprises rendering and re-rendering in a loop the first set of pixels being shifted.

This Summary is provided to introduce a selection of concepts in a simplified form that are further described below in the Detailed Description. This Summary is not intended to identify key features or essential features of the claimed subject matter, nor is it intended to be used as an aid in determining the scope of the claimed subject matter.

Additional features and advantages will be set forth in the description which follows, and in part will be obvious from the description, or may be learned by the practice of the teachings herein. Features and advantages of the invention may be realized and obtained by means of the instruments and combinations particularly pointed out in the appended claims. Features of the present invention will become more fully apparent from the following description and appended claims, or may be learned by the practice of the invention as set forth hereinafter.

BRIEF DESCRIPTION OF THE DRAWINGS

In order to describe the manner in which the above-recited and other advantages and features can be obtained, a more particular description of the subject matter briefly described

above will be rendered by reference to specific embodiments which are illustrated in the appended drawings. Understanding that these drawings depict only typical embodiments and are not therefore to be considered to be limiting in scope, embodiments will be described and explained with additional specificity and detail through the use of the accompanying drawings in which:

FIG. **1** illustrates a schematic of an embodiment of a computer system for automating the shifting of pixels within a digital image.

FIG. **2** illustrates a schematic of an embodiment of a software architecture for automating the shifting of pixels within a digital image.

FIGS. **3**A-**3**C illustrate embodiments of a photo that has received beginning and ending points.

FIG. **4** illustrates an embodiment of a photo that has formed a mesh from received beginning and ending points.

FIGS. **5**A-**5**D illustrates embodiments of individual pixels being shifted.

FIG. **6**A illustrates an embodiment of a mask.

FIG. **6**B illustrates another embodiment of a mask

FIG. **7** illustrates a flow chart of an embodiment of a method for automating the shifting of pixels within a digital image.

FIG. **8** illustrates a flow chart of another embodiment of a method for automating the shifting of pixels within a digital image.

DETAILED DESCRIPTION

Embodiments disclosed herein comprise systems, methods, and computer-readable media that automate the shifting of pixels within a digital image. Disclosed embodiments provide interfaces and tools for automating the shifting of pixels within a digital image. For example, disclosed embodiments automate the shifting of pixels within a digital photograph of water such that the water appears to be flowing.

In at least one embodiment, a user, through simple interface actions, selects portions of a digital image to automate the shifting of pixels. The user controls the speed of the pixel shifting, the magnitude of the pixel shifting, the direction of pixel shifting, and various other attributes of the pixel movement. Additionally, the user applies masks to the digital image to prevent pixels within specific portions of the digital image from moving. As such, disclosed embodiments provide technical solutions for automating the manipulation of pixels within a digital image.

For example, FIG. **1** illustrates a schematic of an embodiment of a computer system **100** for automating the shifting of pixels within a digital image. The depicted computer system **100** comprises a variety of different components, including one or more client applications **120**, load balancers **110**, application servers **130**(*a-c*), databases **140**, and workers **150**(*a-b*). In the depicted embodiment, the computer system **100** comprises a network-based or cloud-based system that allows a user to manipulate a digital photograph over a network connection. One will understand, however, that alternate or additional embodiments may be otherwise configured such that the system is otherwise distributed or localized. For instance, in at least one embodiment, the system **100** is executed by a single local computer.

In at least one embodiment, a client application **120** is executed within a web browser at a user's computer. The client application **120** comprises a user interface for uploading, or otherwise selecting, a digital image and issuing commands to cause one or more pixels within the digital image to shift. At least a portion of the processing and actual manipulation of the digital image occurs on an application server **130**(*a-c*) within the cloud. A load balancer **110** balances the processing load of one or more users between multiple application servers **130**(*a-c*). For example, the load balancer **110** may ensure that each application server **130** (*a-c*) is, on average, under the same processing load.

The application servers **130**(*a-c*) and the client application **120** are in communication with one or more web storage services **160**, such as AMAZON™ S3 storage. Additionally, the application servers **1430**(*a-c*) may be in communication with various backend databases **140**. The various storage services **160** and databases **140** may store the digital images and software components used within the computer system **100**.

FIG. **2** illustrates a schematic of an embodiment of a software architecture **200** for automating the shifting of pixels within a digital image. In particular, FIG. **2** depicts various screen interfaces **210**, actions **220**, and dialog/context dependent controls **230** that are present within the software architecture **200**. In at least one embodiment, a user is presented with a Home/List Projects screen interface. **210***a* The Home/List Projects **210***a* provide the user with a selection of previously saved projects and/or the option to select a digital image and initiate a new project.

Once a user selects a project of choice, the software architecture **200** accesses, from memory, a digital image file of interest. As used herein, a digital image file comprises information that corresponds to individual pixels within a digital image. A view/edit project screen interface **210***b* provides the user with various editing options, both manual and automated. For example, the editing screen provides the user with actions for uploading a base image **220***a*, selecting a transformation mask **210***c*, generating points **210***d*, adjusting animation duration and FPS rate **220***b*, selecting a mask type **220***c*, generating a static preview **220***d*, and editing a point selection **230***a*.

In at least one embodiment, a user is presented with options to select between various different types of transformation mask operations. For example, the user can choose a select area option **220***e* that allows a user to draw boundaries around an area. A mask is then automatically generated within that area. Additionally, a user may select an automated mask by edge-detection option **220***f* that scans a digital image and automatically identifies distinct areas within the image based upon edge detection. A user may also select a manual mask painting option **220***g* that allows a user to manually draw a mask on the digital image.

A mask defines a static portion of the image with respect to shifting. In at least one embodiment, the mask is created by using a combination of two tools—automatic and manual. The automatic tool lets the user define masked and shifted areas by drawing lines in them. Any changes along those lines trigger an automatic edge detection using a watershed transformation on the image gradient. This generates a binary mask image that the user can later modify via a simple brush tool. The final mask image is then once again passed through edge detection so it can be saved in vector format.

In at least one embodiment, a mask is later applied in one of two ways. In the first application method, the mask is fixed and is applied by adding static (unmovable) point on all mask edges. In the second application method, the mask is an overlay that cuts-and-pastes the masked portions of the image on top of the animation.

For example, FIG. **3**A depicts a user utilizing the automated mask by edge-detection option **220***f*. In particular, the

user has drawn two lines **300***a*, **300***b* across the house and mountain behind the house. The software architecture **200** interprets the lines as indicating the area that the user wishes to mask. In at least one embodiment, a user may only select a single pixel within the area that should be masked. Various different embodiments may accept a variety of different input interfaces. For example, a user may draw the lines using a computer mouse, a touch interface, a stylus interface, a keyboard interface, or any number of other user interfaces.

Upon receiving the selection, the software architecture **200** performs an edge detections method to identify edges that form boundaries around the selected area. After identifying a boundary, the software architecture **200** generates a mask to cover the area within the boundary. For instance, FIG. **3**B illustrates a mask **310** covering the house and the mountain behind the house, but excluding the sky and the grass in front of the house. As such, a user is provided with tools for identifying a particular portion of the digital image to mask. The mask prevents pixels covered by the mask from being shifted.

In at least one embodiments, methods other than generating a mask are used to prevent portions of a digital from moving. For example, in at least one embodiment, a user selects the manual mask painting option **220***g* and manually places stabilizer points around a portion of the digital image that should not move. The stabilizer points indicate to the architecture software that pixels associated with the stationary points should be not be shifted or otherwise moved. In at least one embodiment, the stabilizer points are useable for creating a fence around an area that should not be shifted. For example, placing the stabilizer points causes the architecture software to stop shifting pixels at a boundary that is made up of the stabilizer points.

The software architecture **200** also provides a user with a point generation display **210** that comprises options **220***h*, **220***i* for generating points. For example, the user may be provided with an automatic feature point detection option **220***h* that generate points through automatic feature point detection or a manual feature point addition/removal option **220***i*. When generating points through automatic feature point detection the software architecture **200** identifies areas outside the mask that are associated with directional texture, directional patterns, that otherwise comprise a visually detectable pattern of motion. For example, FIG. **3**C depicts the picture of the house and landscape with points drawn throughout the foreground grass **320***a* and points drawn throughout the clouds in the sky **320***b*.

In at least one embodiment, the software architecture **200** places the points within the grass based upon identified lines and edges within the grass that extend outward from the ground or from the user generated mask **310**. Based upon the lines and textures associated with the grass extending upwards and away from the ground, the software architecture **200** determines that the grass is associated with movement away from the ground. As such, the software architecture **200** automatically places points through the grass.

In at least one embodiment, a user can alternatively or additionally manually add or remove points. For example, a user can manually add all of the points depicted in FIG. **3**C or a user can direct the software architecture **200** to automatically add a portion of the points and then the user can add additional points.

In at least one embodiment, points are added to a digital image in two different stages. During the first stage, the software architecture **200** receives a first starting point through a user interface. A starting point is received through a user selection of a beginning portion of the digital image. The beginning portion of the digital image comprises a starting pixel, or area, from which the user wishes pixels to shift. For example, in FIG. **3**C, various beginning points (also referred to herein as "starting points") comprise the points nearest to the base of the grass and represented by crosses comprise starting points. In at least one embodiment, a beginning point comprise a particular pixel that is selected by a user or automatically selected by the software architecture **200**.

During the second stage of the point generation, the software architecture **200** receives an ending point through the user interface. The ending point is received through a user selection of an ending portion of the user interface. The ending portion of the digital image comprises an ending pixel, or area, to which the user wishes pixels to shift. For example, in FIG. **3**C various ending points are selected on the outer periphery of the grass and are represented by arrows. In at least one embodiment, each starting point is paired with a specific ending point and vice versa. Additionally, the software architecture creates a link between the starting point and first ending point. In at least one embodiment, a link is a vector extending from the starting point to the ending point. A link comprises a direction extending from the first starting point to the first ending point and a length between the first starting point and the first ending point.

A user is capable of choosing the direction of a link and length of a link at will. For example, upon placing a starting point, the user can place an associated ending point at any other location within the user interface—even outside the edges of the digital image. As such, in at least one embodiment, a single digital image can be associated with a multitude of different starting points, ending points, and associated links going in different directions and comprising different lengths. In the case the points are generated automatically by the software architecture **200**, a user is free to move and adjust the points in any way that is desirable.

In at least one embodiment, the software architecture **200** generates a starting mesh from the collective starting points. Similarly, the software architecture **200** generates an ending mesh from the collective ending points. For example, FIG. **4** depicts a starting mesh and ending mesh overlaid on the digital image of the house and landscape. In at least one embodiment, the software architecture provides the user with the option of viewing one or both of the meshes. Visualizing the meshes may assist a user in viewing the inputs that cause the pixels to shift.

Once a user has established one or more starting and ending point pairs, the software architecture identifies sets of pixels that lie between the respective starting points and the respective ending points. In at least one embodiment, a first set of pixel may comprise a group of pixels that intersect with a link between a starting point and an ending point and that are not covered by a mask. The size of the group of pixels may be user selectable or automatically determined. For example, a user may increase the threshold distances that the set of pixels extend beyond the link. As such at one extreme, the set of pixels may comprise a line of individual pixels extending from the starting point to the ending point, or at another extreme the set of pixels may comprise a relatively wide swatch of pixels that are parallel to the link that extends between the starting point and the ending point.

After identifying the appropriate set of pixels associated with each starting point and ending point pair, the software architecture **200** shifts the respective sets of pixels in the direction determined by their relative links. In at least one embodiment, shifting the sets of pixels comprises rendering

and re-rendering in a loop the sets of pixels being shifted. Additionally, as described above, different sets of pixels may be associated with different directions. As such, sets of pixels may travel in different directions within the digital image.

In at least one embodiment once pixels begin to shift, the user may notice that a particular portion of the digital image that the user desired to not move is in fact being shifted. To correct this error, the user can edit the mask such that it covers the portion of interest. Additionally, in at least one embodiment, to correct the error, the user executes a feather tool that blurs the portion of interest such that the shifting pixels are no longer noticeable. In a further embodiment, the user places stabilizer points along the portion of interest. The stabilizer points may be placed with a simple click along the particular portion of the digital image. The stabilizer point cause pixels within the particular portion of the digital image to not be shifted.

By way of example, FIGS. **5**A-**5**D depict embodiments of individual pixels being shifted. In particular, FIG. **5**A depicts a starting point **550**, an ending point **530**, and an associated link **540**. Additionally, FIG. **5**A depicts exemplary pixels **500**, **510**, and **520**. One will understand that the pixels **500**, **510**, **520** are provided only for the sake of clarity and explanation and that in various embodiments the pixels may comprise a variety of different colors.

FIG. **5**A depicts the pixels **500**, **510**, **520** in their original state within the digital image. Once the software architecture **200** begins shifting the set of pixels in the direction of the link (extending from the starting point to the ending point), the pixels **500**, **510**, **520** are moved. For example, FIG. **5**B depicts the pixels **500**, **510**, **520** after the initial movement. As compared to FIG. **5**A the set of pixels are shifted left, along the direction of the link, one space.

In various embodiments, a user is provided with an animation duration and FPS rate option **220***b* for determining the step size within the shift and/or the speed at which the shift occurs. For example, while the embodiment depicted in FIGS. **5**A-**5**D have a step size of a single pixel space, in various embodiments, the pixels may shift multiple pixels at a time. Additionally, in at least one embodiment the step size is time constrained. For example, a user may indicate that each respective loop in the pixel shifting only take 2 seconds. In such a case, the step size may be determined by the length of the link and the frame rate of the shifting. In other words, one set of pixels may need to travel a link length of 20 pixels within the 2 seconds, while another set of pixels may only need to travel a link length of 6 pixels within two seconds. As such, based upon the frame rate at which the steps occur, the two sets of pixels would operate at different step sizes. In at least one embodiment, a user can set a particular desired framerate to be associated with a digital image. Additionally, in at least one embodiment, the magnitude of the shifting of a set of pixels is proportionally related to the length of an associated link.

In contrast in at least one embodiment, the step size of each set of pixels is consistent such that sets of pixels along longer links take more time to complete a loop than sets of pixels along shorter links. Accordingly, a user can create a particularly long link to indicate a long, or more extended, motion within a digital image. In contrast, the user can create a particular short link to indicate a short, or faster, motion within the digital image.

FIG. **5**C depicts pixels **500**, **510**, **520** another step along the shift. In FIG. **5**D the pixels **500**, **510**, **520** are shown at yet another step along the shift, but in this step the pixels **500**, **510**, **520** are beginning to loop around. For example, pixel **500***a* is at the left of the shift, while corresponding pixel **500***b* is reappearing at the right of the shift to begin the shifting loop. In various additional or alternative embodiments, the set of pixels only begins to loop as pixels **500**, **510**, **520** leave the screen. For example, once pixel **500***a* reaches the end of the link (i.e., ending point **530**), pixel **500***a* disappears on the next step. Upon disappearing, pixel **500***b* then reappears at the beginning of the link (i.e., starting point **550**) to start the loop. Additionally, in at least one additional or alternative embodiment, the pixels **500**, **510**, **520** fade in and out at the beginning and ending of each loop. For example, upon reaching the end of the link, pixel **500***a* fades out of the image by slowing growing transparent, such that the disappearance of pixel **500***a* is not abrupt or jarring. Similarly, pixel **500***b* can fade in such that its appearance is not abrupt of jarring.

In various alternative or additional embodiments, different methods of pixel shifting and blending can be used to similar effect. For example, in at least one embodiment, the pixels immediately begin cycling as soon as the shifting begins. For example, instead of what is shown in FIGS. **5**A-**5**D, as soon as pixel **500** shifted to its new position depicted in FIG. **5**B, pixel **500**B would appear in the position depicted in FIG. **5**D. As such, the pixels begin looping and blending into each other immediately.

In at least one embodiment, the software architecture **200** uses alpha blending when shifting the pixels from the starting point to the ending point. For example, the software application **200** may utilize a sin, cosine, or linear alpha blend to shift the set of pixels. Additionally, in at least one embodiment, the software architecture uses a warping function, such as a Shepard's distortion. When using a mesh algorithm, the software architecture **200** triangulates the mesh using defined points and then calculates an affine transformation for every triangle. Further, the software architecture **200** supports animation blending in order to achieve the seaming effect of a continuous motion. This is achieved by blending animated frames two by two using one of a few predefined blend functions.

Similarly, a user is provided with a mask type option **220***c* to select between different types of masks. Each different mask type may use different mask configurations. In at least one embodiments, mask types also vary by the color that is used to depict the mask within the image.

For example, FIGS. **6**A and **6**B illustrate an embodiment of a mask **600**. In the depicted embodiment, a strip of points **610** on the edge of the mask ("edge strip") is used to generate the flow of points simulating an animation. The dimensions and shape of the edge strip depend on properties of the mask and the user selected flow. In at least one embodiment, the width of the edge strip **610** is determined as a linear function of the dimensions of the target image. Further, the shape of the edge strip **610** may be a result of sliding the shape of the mask **600** in the direction of the vector flow. As such, a user is able to customize the mask.

When in use, the mask **600** assists in generating pixel flow. For example, as depicted in FIG. **6**B the points **632**, **642**, **652** on the edge strip **610** are being stretched in order to generate a flow of points in the desired direction. For example, the points **632**, **642**, **652** are shown as being stretched to strips **630**, **640**, **650** that are stretched from the edge **610** of the mask **600**. The stretching motion is performed continuously such that the points **632**, **642**, **652** are continuously reset and stretched such that the pixel shifting is continuous and results in the impression of motion.

In various additional or alternative embodiments, a user has the ability to further manipulate a digital image. For

example, instead of the straight links depicted in FIGS. **5**A-**5**D, the use can view an edit point selection dialog **230***a* as depicted in FIG. **2**. The edit point selection provides a user with the options of changing a movement direction and path **220***j* or selecting a predefined movement **220***k*. For example, instead of straight links extending between a starting point and ending point, the user can create a custom path between any two starting and ending points. In such a case, the associated set of pixels would travel along the path of the link. Similarly, the user may select a predefined path (i.e., movement) that is defined within the software architecture **200**. For example, the predefined paths may include a curved path. Using the curved path option, the user can cause a set of pixels to travel along a curved path (i.e., link) between a starting and ending point.

Once a user is satisfied with their work on an image, a preview output screen **210***e* allows a user to view the image while the pixels are being shifted. Such a view may give the impression that at least a portion of the static image is animated. In contrast, a user is also give a static preview option **220***d* that allows the user to view the un-animated image. If the user is satisfied with the final product, the user is provided with a format selection and export action **220**. This action generates a variety of different formats that are configured for viewing. For example, an animated GIF may be produced.

Additionally, in at least one embodiment, a user is able to save all of the masking work, point generation work, and animation duration and FPS rate work separate from the underlying image. For example, the user may perform the above discussed steps on a particular photo. The user may then decide to manipulate the photo within a photo editor application. In such a case, instead of having to restart the animation process with the modified photo, the user applies the previously saved animation work to the modified photo.

The following discussion now refers to a number of methods and method acts that may be performed. Although the method acts may be discussed in a certain order or illustrated in a flow chart as occurring in a particular order, no particular ordering is required unless specifically stated, or required because an act is dependent on another act being completed prior to the act being performed.

One will appreciate that embodiments disclosed herein can also be described in terms of flowcharts comprising one or more acts for accomplishing a particular result. For example, FIG. **7** and the corresponding text describe acts in various methods and systems for automating the shifting of pixels within a digital image. The acts of FIG. **7** are described below.

For example, FIG. **7** illustrates that a flow chart of an exemplary method **700** for automating the shifting of pixels within a digital image includes an act **710** of receiving a starting point. Act **710** comprises receiving an indication of a starting point through a user interface, wherein the starting point is received through a user selection of a particular portion of the digital image. For example, as depicted and explained with respect to FIG. **5**A, a user selects a starting point **550**. Similarly, FIG. **3**C depicts various user-selected starting points surrounding the grass in the digital image.

Additionally, FIG. **7** illustrates that the method includes an act **720** of receiving a direction. Act **720** comprises receiving, through the user interface, a direction associated with the starting point. For example, as depicted and explained with respect to FIG. **5**A, the use can select an ending point **530**. A link **540** is then formed between the starting point **550** and the ending point **530**. The link **540** is associated with a direction extending from the starting point **550** to the ending point **530**. As such, the software architecture **200** receives a direction from the link **540**.

In contrast, in additional or alternative embodiments, the software architecture **200** can receive a direction without the use of an ending point **530** or a link **540**. For example, the software architecture **200** may receive the direction from a default setting stored within memory or explicitly from the user. Further, in at least one embodiment, the software architecture **200** identifies a direction from line and edge detection. For example, the software architecture **200** may identify a direction that follow the blades of grass depicted in FIG. **3**A. In any case, in at least one embodiment, a user may only need to enter a starting point and the direction is identified either automatically or through user interaction other than the creation of an ending point.

FIG. **7** also illustrates that the method includes an act **730** of selecting a set of pixels. Act **730** comprises selecting a set of pixels extending in the direction away from the starting point. For example, as depicted and explained with respect to FIG. **5**A, the software architecture **200** selects a set of pixels **500**, **510**, **520** that extend in the direction of the link **540** away from the starting point **550**. The computer architecture **200** may define the set of pixels **500**, **510**, **520** as being every pixel that extends in the direction away from the starting point and that is within a user or computer-defined threshold distance from the link **540**.

Further, FIG. **7** illustrates that the method includes an act **740** of shifting the set of pixels. Act **740** comprises shifting the set of pixels in the first direction. Shifting the set of pixels comprises rendering and re-rendering in a loop the set of pixels being shifted. For example, as depicted and explained with respect to FIGS. **5**A-**5**D, a shifting loop is described where pixels **500**, **510**, **520** are shifted upward and then re-looped.

Additionally, FIG. **8** depicts an embodiment of a method **800** for shifting pixels comprises a step **810** of accessing a digital file. Act **810** includes accessing, from memory, a digital image file, wherein the digital image file comprises information that corresponds to individual pixels within the digital image. For example, as depicted and described with respect to FIG. **1**, a user can upload a digital image through a client application **120**. The computer system **100** then receives the digital image.

FIG. **8** also illustrates that the method **800** includes an act **820** of identifying a portion of an image to mask **820**. Act **820** comprises identifying a particular portion of the digital image to mask, wherein the mask prevents pixels covered by the mask from being shifted. For example, as depicted and described with respect to FIGS. **3**A-**3**C, a user is able to mask a particular portion of the digital image. The masked portion of the image is held static, such that the pixels do not shift.

Additionally, FIG. **8** illustrates that the method **800** includes an act **830** of receiving a starting point. Act **830** comprises receiving a first starting point through a user interface, wherein the first starting point is received through a user selection of a first beginning portion of the digital image. For example, as depicted and explained with respect to FIG. **5**A, a user selects a starting point **550**. Similarly, FIG. **3**C depicts various user-selected starting points surrounding the grass in the digital image.

FIG. **8** also illustrates that the method **800** includes an act **840** of receiving an ending point. Act **840** comprises receive a first ending point through the user interface, wherein the first ending point is received through a user selection of a first ending portion of the user interface. For example, as depicted and explained with respect to FIG. **5**A, a user

selects an ending point **530**. Similarly, FIG. **3**C depicts various user-selected starting ending surrounding the grass in the digital image.

In addition, FIG. **8** illustrates that the method **800** includes an act **850** of creating a link between the beginning and the ending points. Act **850** comprises create a first link between the first starting point and the first ending point, wherein the first link comprises a first direction extending from the first starting point to the first ending point and a first length between the first starting point and the first ending point. For example, as depicted and explained with respect to FIG. **5**A, a link **540** is created between beginning points **550** and ending points **530**. Similarly, FIG. **3**C depicts various links connecting beginning points and ending points.

Further, FIG. **8** illustrates that the method **800** includes an act **860** of identifying a set of pixels. Act **860** comprises identifying a first set of pixels that lie between the first starting point and the first ending point. For example, as depicted and explained with respect to FIG. **5**A, the software architecture **200** identifies a set of pixels **500**, **510**, **520** that extend in the direction of the link **540** away from the starting point **550**. The computer architecture **200** may define the set of pixels **500**, **510**, **520** as being every pixel that extends in the direction away from the starting point and that is within a user or computer-defined threshold distance from the link **540**.

Further still, FIG. **8** illustrates that the method **800** includes an act **870** of shifting the set of pixels **870**. Act **870** comprises shifting the second set of pixels in the second direction, wherein shifting the second set of pixels comprises rendering and re-rendering in a loop the second set of pixels being shifted. For example, as depicted and explained with respect to FIGS. **5**A-**5**D, a shifting loop is described where pixels **500**, **510**, **520** are shifted upward and then re-looped.

Accordingly, disclosed embodiments provide novel and innovative technical methods for automatically shifting pixels within a digital image. The shifted pixels may give a digital image the perception of movement. At least one disclosed embodiment requires only a single digital image to create a perception of movement within the digital image. Disclosed embodiments also provide novel and innovative technical solutions for receiving user input through a computer interface to simply and efficiently manipulate a digital image by shifting pixels.

Further, the methods may be practiced by a computer system including one or more processors and computer-readable media such as computer memory. In particular, the computer memory may store computer-executable instructions that when executed by one or more processors cause various functions to be performed, such as the acts recited in the embodiments. In various embodiments, disclosed methods and systems may comprises software executed within the cloud. For example, a user may access the software through a web browser. In additional or alternative embodiments, the software is executed locally at a device. For example, the software may be executed on a mobile computing device such as a smart phone or tablet.

Embodiments of the present invention may comprise or utilize a special purpose or general-purpose computer including computer hardware, as discussed in greater detail below. Embodiments within the scope of the present invention also include physical and other computer-readable media for carrying or storing computer-executable instructions and/or data structures. Such computer-readable media can be any available media that can be accessed by a general purpose or special purpose computer system. Computer-readable media that store computer-executable instructions are physical storage media. Computer-readable media that carry computer-executable instructions are transmission media. Thus, by way of example, and not limitation, embodiments of the invention can comprise at least two distinctly different kinds of computer-readable media: physical computer-readable storage media and transmission computer-readable media.

Further, computing system functionality can be enhanced by a computing system's ability to be interconnected to other computing systems via network connections. Network connections may include, but are not limited to, connections via wired or wireless Ethernet, cellular connections, or even computer to computer connections through serial, parallel, USB, or other connections. The connections allow a computing system to access services at other computing systems and to quickly and efficiently receive application data from other computing systems.

Interconnection of computing systems has facilitated distributed computing systems, such as so-called "cloud" computing systems. In this description, "cloud computing" may be systems or resources for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, services, etc.) that can be provisioned and released with reduced management effort or service provider interaction. A cloud model can be composed of various characteristics (e.g., on-demand self-service, broad network access, resource pooling, rapid elasticity, measured service, etc.), service models (e.g., Software as a Service ("SaaS"), Platform as a Service ("PaaS"), Infrastructure as a Service ("IaaS"), and deployment models (e.g., private cloud, community cloud, public cloud, hybrid cloud, etc.).

Cloud and remote based service applications are prevalent. Such applications are hosted on public and private remote systems such as clouds and usually offer a set of web based services for communicating back and forth with clients.

Many computers are intended to be used by direct user interaction with the computer. As such, computers have input hardware and software user interfaces to facilitate user interaction. For example, a modern general-purpose computer may include a keyboard, mouse, touchpad, camera, etc. for allowing a user to input data into the computer. In addition, various software user interfaces may be available.

Examples of software user interfaces include graphical user interfaces, text command line based user interface, function key or hot key user interfaces, and the like.

Physical computer-readable storage media includes RAM, ROM, EEPROM, CD-ROM or other optical disk storage (such as CDs, DVDs, etc.), magnetic disk storage or other magnetic storage devices, or any other medium which can be used to store desired program code means in the form of computer-executable instructions or data structures and which can be accessed by a general purpose or special purpose computer.

A "network" is defined as one or more data links that enable the transport of electronic data between computer systems and/or modules and/or other electronic devices. When information is transferred or provided over a network or another communications connection (either hardwired, wireless, or a combination of hardwired or wireless) to a computer, the computer properly views the connection as a transmission medium. Transmissions media can include a network and/or data links which can be used to carry or desired program code means in the form of computer-executable instructions or data structures and which can be

accessed by a general purpose or special purpose computer. Combinations of the above are also included within the scope of computer-readable media.

Further, upon reaching various computer system components, program code means in the form of computer-executable instructions or data structures can be transferred automatically from transmission computer-readable media to physical computer-readable storage media (or vice versa). For example, computer-executable instructions or data structures received over a network or data link can be buffered in RAM within a network interface module (e.g., a "NIC"), and then eventually transferred to computer system RAM and/or to less volatile computer-readable physical storage media at a computer system. Thus, computer-readable physical storage media can be included in computer system components that also (or even primarily) utilize transmission media.

Computer-executable instructions comprise, for example, instructions and data which cause a general-purpose computer, special purpose computer, or special purpose processing device to perform a certain function or group of functions. The computer-executable instructions may be, for example, binaries, intermediate format instructions such as assembly language, or even source code. Although the subject matter has been described in language specific to structural features and/or methodological acts, it is to be understood that the subject matter defined in the appended claims is not necessarily limited to the described features or acts described above. Rather, the described features and acts are disclosed as example forms of implementing the claims.

Those skilled in the art will appreciate that the invention may be practiced in network computing environments with many types of computer system configurations, including, personal computers, desktop computers, laptop computers, message processors, hand-held devices, multi-processor systems, microprocessor-based or programmable consumer electronics, network PCs, minicomputers, mainframe computers, mobile telephones, PDAs, pagers, routers, switches, and the like. The invention may also be practiced in distributed system environments where local and remote computer systems, which are linked (either by hardwired data links, wireless data links, or by a combination of hardwired and wireless data links) through a network, both perform tasks. In a distributed system environment, program modules may be located in both local and remote memory storage devices.

Alternatively, or in addition, the functionality described herein can be performed, at least in part, by one or more hardware logic components. For example, and without limitation, illustrative types of hardware logic components that can be used include Field-programmable Gate Arrays (FPGAs), Program-specific Integrated Circuits (ASICs), Program-specific Standard Products (ASSPs), System-on-a-chip systems (SOCs), Complex Programmable Logic Devices (CPLDs), etc.

The present invention may be embodied in other specific forms without departing from its spirit or characteristics. The described embodiments are to be considered in all respects only as illustrative and not restrictive. The scope of the invention is, therefore, indicated by the appended claims rather than by the foregoing description. All changes which come within the meaning and range of equivalency of the claims are to be embraced within their scope.

What is claimed is:

1. A computer system for automating a shifting of pixels within a digital image, comprising:

one or more processors; and

one or more computer-readable media having stored thereon executable instructions that when executed by the one or more processors configure the computer system to perform at least the following:

access, from memory, a digital image file, wherein the digital image file comprises information that corresponds to individual pixels within the digital image;

receive a first starting point through a user interface, wherein the first starting point is received through a user selection of a first beginning portion of the digital image;

receive a first ending point through the user interface, wherein the first ending point is received through a user selection of a first ending portion of the user interface;

create a first link between the first starting point and the first ending point, wherein the first link comprises:

a non-linear pathway extending between the first starting point and the first ending point; and

a first length extending along the non-linear pathway between the first starting point and the first ending point;

identify a first set of pixels that lie along the first link between the first starting point and the first ending point; and

shift the first set of pixels along the first length of the non-linear pathway, wherein shifting the first set of pixels comprises rendering and re-rendering in a loop the first set of pixels being shifted.

2. The computer system of claim **1**, wherein the executable instructions include instructions that are executable to configure the computer system to identify a particular portion of the digital image to mask, wherein:

a mask prevents pixels covered by the mask from being shifted, and

receiving an indication of a particular portion of the digital image to mask comprises receiving through a user interface a selection of a particular pixel within the digital image.

3. The computer system of claim **2**, wherein the executable instructions include instructions that are executable to configure the computer system to generate the mask by:

identifying one or more edges that form a boundary around the particular pixel; and

generating the mask to cover area within the boundary.

4. The computer system of claim **1**, wherein the executable instructions include instructions that are executable to configure the computer system to:

receive a second starting point through the user interface, wherein the second starting point is received through a user selection of a second beginning portion of the digital image and the second beginning portion of the digital image is different than the first beginning portion of the digital image;

receive a second ending point through the user interface, wherein the second ending point is received through a user selection of a second ending portion of the user interface and the second ending portion of the digital image is different than the first ending portion of the digital image;

create a second link between the second starting point and the second ending point, wherein the second link comprises:

a second direction extending from the second starting point to the second ending point; and

a second length between the second starting point and the second ending point;

identify a second set of pixels that lie between the second starting point and the second ending point; and

shift the second set of pixels in the second direction, wherein shifting the second set of pixels comprises rendering and re-rendering in a loop the second set of pixels being shifted.

5. The computer system of claim **4**, wherein the non-linear pathway is different from the second direction.

6. The computer system of claim **4**, wherein the first ending portion of the user interface is outside the edges of the digital image.

7. A method for automating a shifting of pixels within a digital image, comprising:

receiving a first indication of a first starting point through a user interface, wherein the first starting point is received through a user selection of a portion of the digital image;

receiving, through the user interface, an indication of a non-linear pathway extending away from the first starting point;

selecting a first set of pixels extending along a length of the non-linear pathway and away from the first starting point; and

shifting the first set of pixels along the non-linear pathway, wherein shifting the first set of pixels comprises rendering and re-rendering in a loop the first set of pixels being shifted.

8. The method as recited in claim **7**, further comprising receiving an indication of an ending point through a user interface, wherein the ending point is received through a user selection of a different portion of the digital image.

9. The method as recited in claim **8**, further comprising when a pixel selected from the first set of pixels reaches the ending point, rendering and re-rendering in the loop the pixel being shifted from the first starting point to the ending point.

10. The method as recited in claim **7**, further comprising receiving an indication to generate a mask over a portion of the digital image, wherein pixels under the mask are prevented from shifting.

11. The method as recited in claim **10**, further comprising receiving through a user interface a selection of a particular pixel within the digital image from which the mask should be generated.

12. The method of claim **11**, further comprising:

identifying one or more edges that form a boundary around the particular pixel; and

generating the mask to cover area within the boundary.

13. The method as recited in claim **7**, further comprising:

receiving a second indication of a second starting point through a user interface, wherein the second starting point is received through a user selection of a portion of the digital image;

receiving, through the user interface, a second direction associated with the second starting point;

selecting a second set of pixels extending in the second direction away from the second starting point; and

shifting the second set of pixels in the second direction, wherein shifting the second set of pixels comprises rendering and re-rendering in a loop the second set of pixels being shifted.

14. The method as recited in claim **13**, wherein the non-linear pathway is different from the second direction.

15. A method for automating a shifting of pixels within a digital image, comprising:

accessing, from memory, a digital image file, wherein the digital image file comprises information that corresponds to individual pixels within the digital image;

receiving a first starting point through a user interface, wherein the first starting point is received through a user selection of a first beginning portion of the digital image;

receiving a first ending point through the user interface, wherein the first ending point is received through a user selection of a first ending portion of the user interface;

creating a first link between the first starting point and the first ending point, wherein the first link comprises:

a non-linear pathway extending between the first starting point and the first ending point; and

a first length extending along the non-linear pathway between the first starting point and the first ending point;

identifying a first set of pixels that lie along the first length of the non-linear pathway between the first starting point and the first ending point; and

shifting the first set of pixels along the first length of the non-linear pathway, wherein shifting the first set of pixels comprises rendering and re-rendering in a loop the first set of pixels being shifted.

16. The method of claim **15**, further comprising:

identifying a particular portion of the digital image to mask, wherein: a mask prevents pixels covered by the mask from being shifted, and

receiving an indication of a particular portion of the digital image to mask comprises receiving through a user interface a selection of a particular pixel within the digital image.

17. The method of claim **15**, further comprising:

generating a mask, wherein generating the mask comprises:

identifying one or more edges that form a boundary around a particular pixel; and

generating the mask to cover area within the boundary.

18. The method of claim **15**, further comprising:

receiving a second starting point through the user interface, wherein the second starting point is received through a user selection of a second beginning portion of the digital image and the second beginning portion of the digital image is different than the first beginning portion of the digital image;

receiving a second ending point through the user interface, wherein the second ending point is received through a user selection of a second ending portion of the user interface and the second ending portion of the digital image is different than the first ending portion of the digital image;

creating a second link between the second starting point and the second ending point, wherein the second link comprises:

a second direction extending from the second starting point to the second ending point; and

a second length between the second starting point and the second ending point;

identifying a second set of pixels that lie between the second starting point and the second ending point; and

shifting the second set of pixels in the second direction, wherein shifting the second set of pixels comprises rendering and re-rendering in a loop the second set of pixels being shifted.

19. The method of claim **18**, wherein the non-linear pathway is different from the second direction.

20. The method of claim **18**, wherein the first ending portion of the user interface is outside the edges of the digital image.

* * * * *