

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

LIGHTRICKS LTD.,
Petitioner,

v.

PLOTAGRAPH, INC. and SASCHA CONNELLY,
Patent Owner.

Case IPR2023-00153
U.S. Patent No. 10,346,017
Filing Date: Aug. 13, 2018
Issue Date: July 9, 2019

**PETITION FOR *INTER PARTES* REVIEW OF
U.S. PATENT NO. 10,346,017 UNDER 35 U.S.C. §312**

TABLE OF CONTENTS

I.	INTRODUCTION	1
II.	MANDATORY NOTICES	1
A.	Real Party-in-Interest (37 C.F.R. §42.8(b)(1)).....	1
B.	Related Matters (§42.8(b)(2))	1
C.	Lead and Back-Up Counsel (§42.8(b)(3))	2
D.	Service Information (§42.8(b)(4)).....	2
III.	REQUIREMENTS FOR IPR	2
A.	Grounds For Standing (§42.104(a))	2
B.	Identification of Challenge (§42.104(b)(1)-(2)).....	2
1.	Claims for Which Review Is Requested:	2
2.	Grounds of Challenge	3
IV.	LEVEL OF ORDINARY SKILL IN THE ART	5
V.	THE '017 PATENT	6
A.	Prosecution History	7
B.	Claim Construction.....	8
VI.	THE CHALLENGED CLAIMS ARE UNPATENTABLE UNDER 35 U.S.C. §§102(A)(1) AND 103.....	9
A.	Ground 1: Claims 1-18 Are Anticipated or Otherwise Rendered Obvious by AEM.....	9
1.	AEM.....	9
a.	Workspace and Panels	9
b.	Compositions and Layers	9
c.	Modifying Transparency of Certain Pixels	10
d.	Puppet Effect	12
e.	Animating with the Puppet Effect	13
2.	Independent Claim 1	17
a.	[1pre].....	17
b.	[1a].....	17
c.	[1b]-[1c].....	18

	d.	[1d]-[1e].....	20
	e.	[1f]	21
	f.	[1g].....	23
	g.	[1h].....	29
	h.	[1i]-[1o].....	35
3.		Claim 2.....	39
	a.	[2a]-[2b].....	39
4.		Claim 3.....	43
5.		Claim 4.....	44
	a.	[4a]-[4c]	44
6.		Claim 5.....	46
7.		Claim 6.....	47
8.		Claim 7.....	49
	a.	[7a].....	49
	b.	[7b].....	50
9.		Independent Claim 8	51
	a.	[8pre].....	51
	b.	[8a]-[8b].....	51
	c.	[8c]-[8d].....	51
	d.	[8e].....	52
	e.	[8f]	52
	f.	[8g].....	52
	g.	[8h]-[8n].....	52
10.		Claim 9.....	52
11.		Claim 10.....	53
	a.	[10a]-[10b].....	53
12.		Claim 11.....	53
13.		Claim 12.....	53
	a.	[12a]-[12b].....	53
14.		Claim 13.....	53

a.	[13a].....	53
b.	[13b].....	54
15.	Independent Claim 14.....	54
a.	[14pre].....	54
b.	[14a].....	54
c.	[14b]-[14c].....	54
d.	[14d]-[14e].....	54
e.	[14f]	54
f.	[14g].....	54
g.	[14h].....	54
h.	[14i]-[14o].....	55
16.	Claim 15.....	55
a.	[15a]-[15b].....	55
17.	Claim 16.....	55
18.	Claim 17.....	55
a.	[17a]-[17c]	55
19.	Claim 18.....	55
B.	Ground 2: IMU and Okabe Render Obvious Claims 1-3, 5-11, 13-16, and 18.....	55
1.	IMU.....	55
a.	Applying Effects and Creating an Animation	56
b.	Modifying Transparency of Certain Pixels Using a Matte	59
2.	Okabe	60
3.	The IMU-Okabe Combination.....	61
a.	Motivation to Combine IMU with Okabe	61
b.	Resulting Combination.....	64
4.	Independent Claim 1	66
a.	[1pre].....	66
b.	[1a].....	66

c.	[1b]-[1c].....	67
d.	[1d]-[1e].....	69
e.	[1f]	70
f.	[1g].....	71
g.	[1h].....	72
h.	[1i]-[1o].....	74
5.	Claim 2.....	76
a.	[2a]-[2b].....	76
6.	Claim 3.....	77
7.	Claim 5.....	77
8.	Claim 6.....	78
9.	Claim 7.....	79
a.	[7a].....	79
b.	[7b].....	80
10.	Claim 8.....	81
a.	[8pre].....	81
b.	[8a]-[8b].....	81
c.	[8c]-[8d].....	82
d.	[8e].....	82
e.	[8f]	82
f.	[8g].....	82
g.	[8h]-[8n].....	82
11.	Claim 9.....	82
12.	Claim 10.....	83
a.	[10a]-[10b].....	83
13.	Claim 11	83
14.	Claim 13.....	83
a.	[13a].....	83
b.	[13b].....	83
15.	Independent Claim 14.....	84

a.	[14pre].....	84
b.	[14a].....	84
c.	[14b]-[14c].....	84
d.	[14d]-[14e].....	84
e.	[14f]	84
f.	[14g].....	84
g.	[14h].....	84
h.	[14i]-[14o].....	84
16.	Claim 15	84
a.	[15a]-[15b].....	84
17.	Claim 16	85
18.	Claim 18.....	85
C.	Ground 3: IMU, Okabe, and Li Render Obvious Claims 2-4, 10-12, and 15-17.....	85
1.	Li	85
2.	The IMU-Okabe-Li Combination.....	87
a.	Motivation to Combine the IMU-Okabe Combination with Li.....	87
b.	Resulting Combination	90
3.	Claim 2	90
a.	[2a]-[2b].....	90
4.	Claim 3	91
5.	Claim 4	92
a.	[4a]-[4c]	92
6.	Claim 10	93
a.	[10a]-[10b].....	93
7.	Claim 11	93
8.	Claim 12	93
a.	[12a]-[12b].....	93
1.	Claim 15	93
a.	[15a]-[15b].....	93

IPR2023-00153

U.S. Patent No. 10,346,017

2.	Claim 16.....	93
3.	Claim 17.....	93
a.	[17a]-[17c]	93
VII.	DISCRETIONARILY DENIAL IS UNWARRANTED	93
VIII.	CONCLUSION.....	94

Petitioner's Exhibit List

Exhibit	Shorthand	Description
1001	'017 Patent	U.S. Patent No. 10,346,017
1002	Greenspun	Declaration of Philip Greenspun, Ph.D.
1003	AEM	Adobe® After Effects® Help and tutorials, Adobe (2013)
1004	IMU-Home	Wayback Machine Capture dated Mar. 27, 2012 of Anthony Thyssen, <i>Examples of ImageMagick Usage (Version 6)</i> , ImageMagick (Mar. 15, 2011), http://www.imagemagick.org/Usage/ [https://web.archive.org/web/20120327064501/http://www.imagemagick.org/Usage/]
1005	IMU-Distorting	WayBack Machine Capture dated Mar. 29, 2012 of Anthony Thyssen, <i>ImageMagick v6 Examples - - Distorting Images</i> , ImageMagick (Mar. 21, 2012), http://www.imagemagick.org/Usage/distorts/ [https://web.archive.org/web/20120329131929/http://www.imagemagick.org/Usage/distorts/]
1006	IMU-Masking	Wayback Machine Capture dated Sept. 28, 2012 of Anthony Thyssen, <i>ImageMagick v6 Examples - - Masks</i> , ImageMagick (Mar. 10, 2011), http://www.imagemagick.org/Usage/masking/ [https://web.archive.org/web/20120928070642/http://www.imagemagick.org/Usage/masking/]
1007	IMU-Animating	Wayback Machine Capture dated Mar. 10, 2012 of Anthony Thyssen, <i>ImageMagick v6 Examples - - Animation Basics</i> , ImageMagick (Feb. 8, 2011), http://www.imagemagick.org/Usage/anim_basics/ [https://web.archive.org/web/20120310193613/http://www.imagemagick.org/Usage/anim_basics/]
1008	IMU-Windows ¹	Wayback Machine Capture dated Apr. 5, 2012 of Anthony Thyssen, <i>ImageMagick v6 Examples -- Usage under Windows</i> , ImageMagick (Mar. 21,

¹ IMU-Home, IMU-Warping, IMU-Distorting, IMU-Masking, IMU-Animating, and IMU-Windows are collectively referred to hereinafter as “IMU.”

		2012), http://www.imagemagick.org/Usage/windows/ [https://web.archive.org/web/20120405151502/http://www.imagemagick.org/Usage/windows/]
1009	Okabe	Makato Okabe, et al., <i>Creating Fluid Animation from a Single Image using Video Database</i> , 30 Computer Graphics Forum 1973 (2011)
1010	Li	Yin Li, et al., Lazy Snapping, 23 ACM Transactions on Graphics 303 (2004)
1011	'017 PH	Prosecution history of U.S. Patent No. 10,346,017
1012	Archive	Declaration of Nathaniel E Frank-White of the Internet Archive
1013	Lindahl	U.S. Patent No. 9,466,127
1014	McNally	U.S. Patent No. 9,070,222
1015	Siddharth	U.S. Patent App. Pub. No. 2014/0087797
1016	Brookhart	U.S. Patent No. 9,400,942
1017	Sakiyama	U.S. Patent No. 9,692,974
1018	Forbes	Angus Graeme Forbes et al., <i>Analyzing Intrinsic Motion Textures Created from Naturalistic Video Captures</i> , 2014 International Conference on Information Visualization Theory and Applications 107 (2014)
1019	Luo	U.S. Patent No. 8,280,164
1020	Tang	U.S. Patent No. 8,913,847
1021		Notice of Docketing, <i>Plotagraph, Inc. v. Lightricks, Ltd.</i> , No. 23-1048, Dkt. No. 1 (Fed. Cir. Oct. 17, 2022)
1022		Plaintiffs' Unopposed Sworn Motion for Leave to Extend Time to Comply with P.R. 3-1 and P.R. 3-2 Dislcoure [sic] of Asserted Claims and Preliminary Infringement Contentions and Make Document Production, <i>Plotagraph, Inc. v. Lightricks Ltd.</i> , Civil Action No. 4:21-cv-03873, Dkt. No. 38 (S.D. Tex. May 5, 2022)

CLAIM LISTING

Claim	Limitation No.	Limitation
1	[1pre]	A computer system for automating the 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:
	[1a]	access, from memory, a digital image file, wherein the digital image file comprises information that corresponds to individual pixels within the digital image;
	[1b]	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;
	[1c]	render a first visual indication of the first starting point on the user interface;
	[1d]	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 digital image;
	[1e]	render a second visual indication of the first ending point on the user interface;
	[1f]	render, on the user interface, a first rendered link between the first starting point and the first ending point, wherein the first rendered 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;
	[1g]	identify a first set of pixels that are: parallel to the first rendered link,

Claim	Limitation No.	Limitation
		within a user-defined threshold distance from the first rendered link, and extending from the first starting point to the first ending point;
	[1h]	automatically shift 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;
	[1i]	receive a second starting point through the user interface, wherein the second starting point is received through a user selection of a second starting portion of the digital image, the second starting portion being different than both the first starting portion and the first ending portion;
	[1j]	render a third visual indication of the second starting point on the user interface;
	[1k]	receive, through the user interface, a second ending point associated with the second starting point, wherein the second ending point is received through a user selection of a second ending portion of the digital image, the second ending portion being different than both the first starting portion and the first ending portion;
	[1l]	render a fourth visual indication of the second end point on the user interface;
	[1m]	render, on the user interface, a second link between the second starting point and the second ending point;
	[1n]	identify a second set of pixels that are: parallel to the second rendered link, within the user-defined threshold distance from the first rendered link, and

Claim	Limitation No.	Limitation
		extending from the second starting point to the second ending point; and
	[1o]	automatically shift the second set of pixels between the second starting point and the second ending point, wherein shifting the second set of pixels comprises rendering in a loop the second set of pixels being shifted.
2	[2a]	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,
	[2b]	wherein the mask prevents pixels covered by the mask from being shifted.
3	The computer system of claim 2, wherein 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.	
4	[4a]	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:
	[4b]	identifying one or more edges that form a boundary around the particular pixel; and
	[4c]	generating the mask to cover area within the boundary.
5	The computer system of claim 1, wherein the first direction is different from the second direction.	
6	The computer system of claim 1, wherein the magnitude of the shifting of the first set of pixels is proportionally related to the first length and the magnitude of the shifting of the second set of pixels is proportionally related to the second length.	
7	[7a]	The computer system of claim 1, wherein: the first set of pixels comprise pixels that intersect the first rendered link; and

Claim	Limitation No.	Limitation
	[7b]	the first rendered link comprises a non-linear pathway extending between the first starting point and the first ending point.
8	[8pre]	A method for automating the shifting of pixels within a digital image, comprising:
	[8a]	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 first starting portion of the digital image;
	[8b]	render a first visual indication of the first starting point on the user interface;
	[8c]	receiving, through the user interface, a first ending point associated with the first starting point, wherein the first ending point is received through a user selection of a first ending portion of the digital image;
	[8d]	render a second visual indication of the first ending point on the user interface;
	[8e]	render, on the user interface, a first rendered link between the first starting point and the first ending point;
	[8f]	selecting a first set of pixels that intersect with the first rendered link;
	[8g]	automatically shifting the first set of pixels along the first rendered link between the first starting point and the first ending point, wherein shifting the first set of pixels comprises rendering in a loop the first set of pixels being shifted;
	[8h]	receiving a second indication of a second starting point through the user interface, wherein the second starting point is received through a user selection of a second starting portion of the digital image, the second starting portion being different than both the first starting portion and the first ending portion;
	[8i]	render a third visual indication of the second starting point on the user interface;
	[8j]	receiving, through the user interface, a second ending point associated with the second starting

Claim	Limitation No.	Limitation
		point, wherein the second ending point is received through a user selection of a second ending portion of the digital image, the second ending portion being different than both the first starting portion and the first ending portion;
	[8k]	render a fourth visual indication of the second end point on the user interface;
	[8l]	render, on the user interface, a second rendered link between the second starting point and the second ending point;
	[8m]	selecting a second set of pixels that intersect with the second rendered link;
	[8n]	automatically shifting the second set of pixels along the second rendered link between the second starting point and the second ending point, wherein shifting the second set of pixels comprises rendering in a loop the second set of pixels being shifted.
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	[10a]	The method as recited in claim 8, further comprising receiving an indication to generate a mask over a portion of the digital image,
	[10b]	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	[12a]	The method of claim 11, further comprising: identifying one or more edges that form a boundary around the particular pixel; and
	[12b]	generating the mask to cover area within the boundary.
13	[13a]	The method of claim 8, wherein: the first set of pixels further comprise:

Claim	Limitation No.	Limitation
		pixels that are parallel to the first rendered link, within a user-defined threshold distance from the first rendered link, and that extend from the first starting point to the first ending point; and
	[13b]	the first rendered link comprises a non-linear pathway extending between the first starting point and the first ending point.
14	[14pre]	A method for automating the shifting of pixels within a digital image, comprising:
	[14a]	accessing, from memory, a digital image file, wherein the digital image file comprises information that corresponds to individual pixels within the digital image;
	[14b]	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;
	[14c]	rendering a first visual indication of the first starting point on the user interface;
	[14d]	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 digital image;
	[14e]	rendering a second visual indication of the first ending point on the user interface;
	[14f]	rendering, on the user interface, a first rendered 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;

Claim	Limitation No.	Limitation
	[14g]	identifying a first set of pixels that lie along the first rendered link;
	[14h]	automatically 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;
	[14i]	receiving a second starting point through the user interface, wherein the second starting point is received through a user selection of a second starting portion of the digital image, the second starting portion being different than both the first starting portion and the first ending portion;
	[14j]	rendering a third visual indication of the second starting point on the user interface;
	[14k]	receiving, through the user interface, a second ending point associated with the second starting point, wherein the second ending point is received through a user selection of a second ending portion of the digital image, the second ending portion being different than both the first starting portion and the first ending portion;
	[14l]	rendering a fourth visual indication of the second end point on the user interface;
	[14m]	rendering, on the user interface, a second link between the second starting point and the second ending point;
	[14n]	selecting a second set of pixels along the second link; and
	[14o]	automatically shifting the second set of pixels between the second starting point and the second ending point, wherein shifting the second set of pixels comprises rendering in a loop the second set of pixels being shifted.
15	[15a]	The method of claim 14, further comprising identifying a particular portion of the digital image to mask,
	[15b]	wherein the mask prevents pixels covered by the mask from being shifted.

Claim	Limitation No.	Limitation
16		The method of claim 15, wherein 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	[17a]	The method of claim 15, further comprising: generating the mask, wherein generating the mask comprises:
	[17b]	identifying one or more edges that form a boundary around the particular pixel; and
	[17c]	generating the mask to cover area within the boundary.
18		The method of claim 15, wherein: the first set of pixels further comprise: pixels that are parallel to the first rendered link, within a user-defined threshold distance from the first rendered link, and that extend from the first starting point to the first ending point.

I. INTRODUCTION

The '017 Patent's claims are directed to a computer system and method for “automating the shifting of pixels within a digital image.” Pixels are shifted according to user inputs, i.e., the user indicates starting points and ending points in the image for links and sets of pixels to be shifted along the links, and the pixels are shifted from the starting points to the ending points along the links in a rendered loop.

However, the foregoing features were included in image editing programs prior to the '017 Patent's filing date, including Adobe After Effects CS6. Separately, the claimed inventions would have been obvious given the state of the art and other previously-available image editing programs, including ImageMagick Version 6 and features described in Okabe and Li. Therefore, claims 1-18 are unpatentable.

II. MANDATORY NOTICES

A. Real Party-in-Interest (37 C.F.R. §42.8(b)(1))

The real party-in-interest is: Petitioner, Lightricks Ltd.

B. Related Matters (§42.8(b)(2))

Related matters are *Plotagraph, Inc. v. Lightricks Ltd.*, Civil Action No. 4:21-cv-03873 (S.D. Tex. 2022) (the “Litigation”), the dismissal of which is on appeal in *Plotagraph, Inc. v. Lightricks, Ltd.*, No. 23-1048 (Fed. Cir. 2022) (the “Appeal”).

EX1021.

C. Lead and Back-Up Counsel (§42.8(b)(3))

Lead Counsel	Back-Up Counsel
Robert Greeson, Reg. No. 52,966	R. Ross Viguet, Reg. No. 42,203 Lu Ling, Reg. No. 78,121

D. Service Information (§42.8(b)(4))

Lead Counsel	Back-Up Counsel
Robert Greeson, Reg. No. 52,966 Norton Rose Fulbright US LLP 2200 Ross Ave, Suite 3600 Dallas, Texas 75201 214.855.7430 (telephone) robert.greeson@nortonrosefulbright.com	R. Ross Viguet, Reg. No. 42,203 Norton Rose Fulbright US LLP 2200 Ross Ave, Suite 3600 Dallas, Texas 75201 214.855.8185 (telephone) ross.viguet@nortonrosefulbright.com Lu Ling, Reg. No. 78,121 Norton Rose Fulbright US LLP 1301 McKinney, Suite 5100 Houston, Texas 77010 713.651.5327 (telephone) lu.ling@nortonrosefulbright.com

Petitioner consents to electronic service.

III. REQUIREMENTS FOR IPR**A. Grounds For Standing (§42.104(a))**

Petitioner certifies that the '017 Patent is available for *inter partes* review (“IPR”) and Petitioner is not barred or estopped from requesting IPR on the identified grounds. 37 C.F.R. §42.104(a).

B. Identification of Challenge (§42.104(b)(1)-(2))**1. Claims for Which Review Is Requested:**

1-18 (the “Challenged Claims”).

2. Grounds of Challenge²

Petitioner requests IPR of the Challenged Claims on the grounds set forth in the table below.

Ground	Claims	Statutory Basis	Prior Art
1	1-18	§§102(a)(1), 103	AEM
2	1-3, 5-11, 13-16, and 18	§103	IMU and Okabe
3	2-4, 10-12, and 15-17	§103	IMU, Okabe, and Li

AEM is a user manual for the popular Adobe After Effects CS6 software program (“AECS6”). *See, e.g.*, AEM, 3; Greenspun, ¶43. AEM is prior art because by September 7, 2012, it was downloadable by the interested public from Adobe’s website. Archive, ¶¶1-7 (¶¶5-6: explaining URL and attachment), pp. 97-99 (declarant-cited attachment at 07-September-2012 URL); Greenspun, ¶43 (testifying AEM, which he provided, matches copy at Archive, 99-699). Other references confirm the public availability of AECS6 and its user manual, AEM, before the earliest claimed priority date (“ECPD”). *See, e.g.*, Lindahl, code (56); McNally, code (56).

² All grounds include a POSITA’s general knowledge at the time of the alleged invention, which is evidenced by the prior art, including that which is cited in support of, but not explicitly listed as part of, the grounds. *Koninklijke Philips N.V. v. Google LLC*, 948 F.3d 1330, 1337-38 (Fed. Cir. 2020).

IMU contains Wayback Machine captures dated 2012 of the ImageMagick.com website, specifically the website's section titled "Examples of ImageMagick Usage (Version 6)," which provides guidance and examples on how to use the popular ImageMagick Version 6 ("IMV6") opensource image processing software. *See* IMU-Home, 1; Greenspun, ¶44. The section's homepage (IMU-Home) contains links to different subpages explaining how to use IMV6's various effects and capabilities, including "Distorting Images" (IMU-Distorting), "Masking and Background Removal" (IMU-Masking), "Animation Basics" (IMU-Animating), and "Usage under Windows" (IMU-Windows).³ IMU-Home, 1-2; Greenspun, ¶44. The above homepage and subpages were each accessible by the interested public on the ImageMagick.com website by September 28, 2012, establishing IMU's prior art status. Archive, pp. 3-5, 9-10, 45-46, 65-66, 79-80; Greenspun, ¶44 (testifying IMU, which he provided, matches copy at Archive, 4-96,

³ IMU is thus a single reference. Greenspun, ¶44 n.6. Separately, a POSITA would have also been motivated to consider the IMU webpages together in order to gain a more complete understanding of IMV6. *Id.*; *see also Haliburton Energy Servs., Inc. v. Dynamic 3d Geosolutions LLC*, IPR2014-01186, Paper 13, 19-20 (PTAB Jan. 12, 2015).

excluding declarant-provided cover pages). Other references confirm the pre-ECPD public availability of IMU. *See, e.g.*, Siddharth, ¶¶120-22; Brookhart, 10:53-59.

Okabe was published in Volume 30, Number 7 of Computer Graphics Forum in September 2011 and is therefore prior art. Okabe, 1; Greenspun, ¶45. Other references confirm the pre-ECPD public availability of Okabe. *See, e.g.*, Sakiyama, code (56); Forbes, 3, 7.

Li was published in Volume 23, Issue 3 of ACM Transactions on Graphics in August 2004 and is therefore prior art. Li, 1; Greenspun, ¶46. Other references confirm the pre-ECPD public availability of Li. *See, e.g.*, Luo, code (56), 1:43-53; Tang, 1:21-24; Okabe, 7, 10.

IV. LEVEL OF ORDINARY SKILL IN THE ART

A POSITA as of the ECPD⁴ would have had at least a bachelor's degree in computer science, electrical engineering, or a related field, and at least 1-2 years of experience in image processing and animation. Greenspun, ¶¶23-24. Less education

⁴ All Petition statements about the knowledge and skills of, and what would have been understood by, a POSITA are from this perspective as of this date. *See* Greenspun, ¶¶27-29; *see also id.*, ¶¶1-2 (scope of work), 3-20 (experience/qualifications), 21-29 (legal understanding and basis for opinions), 30-40 (technology background).

could have been compensated with more experience, and vice versa. *Id.* A POSITA would have also been familiar with existing systems for image processing and animation and would have understood how to implement such systems. *Id.* The '017 Patent and the prior art discussed in Section VI evidence this level of ordinary skill.

V. THE '017 PATENT

The '017 Patent explains that known “digital image software tools” can be used to “edit or otherwise manipulate digital images,” such as “to incorporate movement within a digital image.” '017 Patent, 1:35-40, 1:50-52; Greenspun, ¶41. The '017 Patent itself is directed to “interfaces and tools for automating the shifting of pixels within a digital image.” '017 Patent, 3:34-36, Abst.; Greenspun, ¶41. To use such interfaces and tools, a user provides a “starting point” on a digital image, as well as “a direction associated with the starting point.” '017 Patent, Abst.; Greenspun, ¶41. A set of pixels that “extend[s] in the direction away from the starting point” is then selected and shifted in the first direction, where the shifting comprises “rendering and re-rendering in a loop the set of pixels being shifted.” '017 Patent, Abst.; Greenspun, ¶41. Figure 7 illustrates the claimed invention:

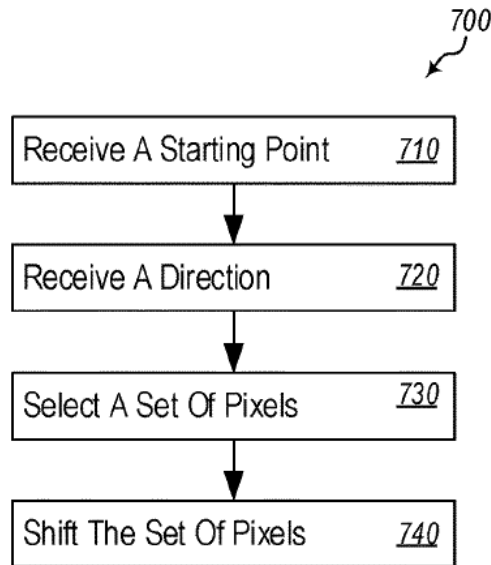


Fig. 7

'017 Patent, Fig. 7; Greenspun, ¶41.

A. Prosecution History

During prosecution, the Applicant requested, and was granted, “Track One” prioritized examination under 37 C.F.R. §1.102(e)(1). '017 PH, 1, 81. The Examiner rejected the originally filed claims 1-20 for, among other reasons,⁵ reciting unpatentable subject matter under 35 U.S.C. §101 (*id.*, 86) and being obvious over several references (*id.*, 113, 205).

In response to the §101 rejection, the Applicant amended independent claim 1's “shift the first set of pixels” limitation (currently, [1h]) to recite “automatically

⁵ The Examiner also rejected the claims for nonstatutory double patenting over a related, then-pending US Patent Application No. 15/660,667. *Id.*, 98, 188, 201.

shift the first set pixels...,” and amended other claims similarly. *Id.*, 173, 175-76, 178-79. The Applicant argued that, because “[t]he automatic shifting necessarily requires the use of a computer and is not performed by hand,” the claims were now “directed towards patent-eligible subject matter and do not include the abstract idea of an ‘idea of itself.’” *Id.*, 182-83. The Examiner accepted these arguments and withdrew the rejection. *Id.*, 202.

As for the obviousness rejection, the Applicant provided numerous amendments to the claims, including adding a “user interface” in various steps of the claims (*e.g.*, *id.*, 251, 265), specifying the claimed “set of pixels” includes pixels having a certain relationship with the claimed “rendered link” (*e.g.*, *id.*, 252, 265), and requiring the receipt of not one but a plurality of different “starting point[s]” and “ending point[s]” (*e.g.*, *id.*, 173, 183-85). In view of these amendments, the Examiner withdrew the obviousness rejection and ultimately allowed the claims. *Id.*, 287-91.

B. Claim Construction

For purposes of this IPR, all terms should be given their plain and ordinary meaning. Greenspun, ¶42.

VI. THE CHALLENGED CLAIMS ARE UNPATENTABLE UNDER 35 U.S.C. §§102(A)(1) AND 103**A. Ground 1: Claims 1-18 Are Anticipated or Otherwise Rendered Obvious by AEM****1. AEM**

AEM is a user manual for AECS6—a software that animates digital images from, e.g., “animat[ing] a simple title” to “creat[ing] complex motion graphics, or composit[ing] realistic visual effects.” AEM, 25; Greenspun, ¶47. AEM explains how to navigate AECS6’s workspace and panels, define “compositions” and “layers,” apply a “mask” or “matte” to a digital image, and animate the image using the “Puppet” effect.

a. Workspace and Panels

According to AEM, AECS6’s user interface comprises an application window housing “panels” organized in a “workspace.” AEM, 39; Greenspun, ¶48. Different panels contain different tools and effects for editing and animating images. Greenspun, ¶48; *see also, e.g.*, AEM, 25, 39.

b. Compositions and Layers

AEM explains that a composition is the “framework” for animation in AECS6. AEM, 75; Greenspun, ¶49. “Footage items,” such as images, are imported into a composition. AEM, 75, 99-101 (listing exemplary footage item file formats, including “Still-image formats” such as JPEG, PNG, and PDF); Greenspun, ¶49.

Once imported, each footage item becomes a “layer” in that composition. AEM, 75, 98; Greenspun, ¶49.

Layers are “the elements that make up a composition.” AEM, 120; Greenspun, ¶50. Layers are stacked in a “vertical arrangement” such that the uppermost layer is visible. AEM, 125; Greenspun, ¶50.

Additionally, “[AECS6] includes a variety of effects, which you apply to layers to add or modify characteristics of still images, video, and audio.” AEM, 335; Greenspun, ¶51. Such effects are previewed and applied to a composition in the “Composition panel,” or to a single layer in the “Layer panel.” AEM, 75, 120, 335; Greenspun, ¶51. When ready, the user “render[s] [the] composition to create the frames of a final output movie.” AEM, 75; Greenspun, ¶51.

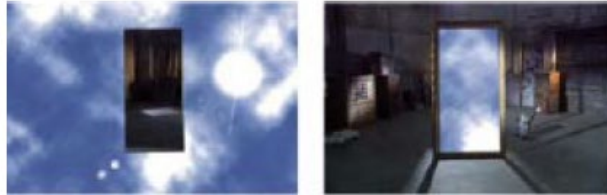
c. Modifying Transparency of Certain Pixels

AEM describes tools that modify the transparency of a layer’s pixels and thus the visibility of effects applied to the layer. “You can make portions of a layer transparent using any of several features in After Effects,” including “masks” and “mattes.” AEM, 315; Greenspun, ¶52. Such masks and mattes modify the layer’s “alpha channel,” which “determines the transparency of the layer at each pixel.” AEM, 318; Greenspun, ¶52.

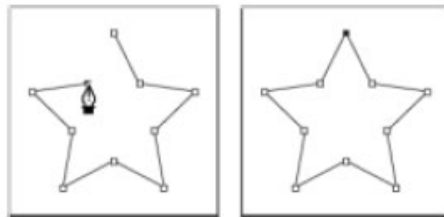
Regarding masks, AEM explains a user applies a mask to all or a portion of a layer so pixels enclosed within the mask are made nontransparent, while the

remaining pixels are made transparent, or vice versa. AEM, 318; Greenspun, ¶53.

AEM provides an example composition illustrating this functionality using two layers and a rectangular mask:



AEM, 318; Greenspun, ¶53. The user creates a mask by using, e.g., AECS6's "Pen" tool to click on different points on the layer in the "Composition" or "Layer panel[s]" to specify the path and vertices of the mask. AEM, 264-66; Greenspun, ¶53. AEM provides an example of creating a star-shaped mask using the Pen tool:



AEM, 265; Greenspun, ¶53. Masks can also be automatically created for a layer using AECS6's "Auto-trace" function, which creates a mask by "searching for edges" across the entire layer and tracing such edges. AEM, 262; Greenspun, ¶53. Such edges are detected based on "the alpha, red, green, blue, or luminance channel of [the] layer." AEM, 262; Greenspun, ¶53.

Mattes similarly "define[] the transparent areas" of a layer. AEM, 317; Greenspun, ¶54. Specifically, a matte isolates a layer's "foreground" from its

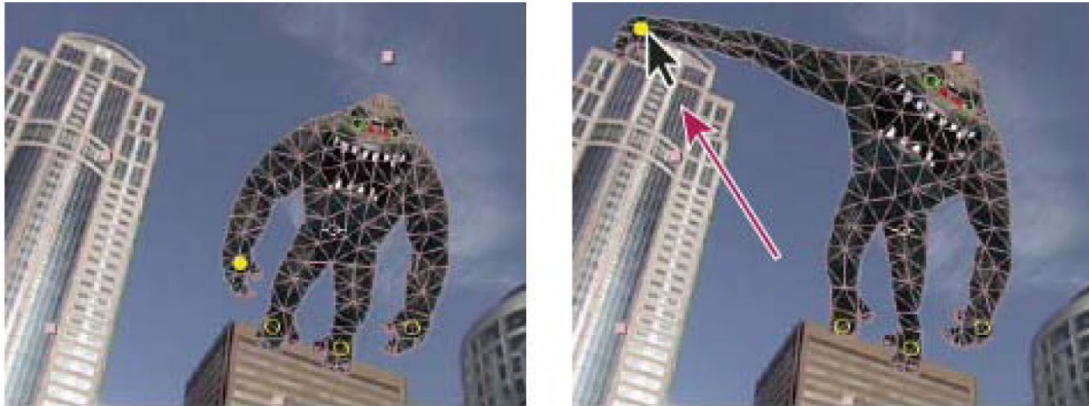
“background,” making the latter transparent. AEM, 328; Greenspun, ¶54. The user creates a matte by using, e.g., AECS6’s “Roto Brush” tool to draw “strokes” in the “Layer panel” over “representative areas of the foreground and background elements.” AEM, 328; Greenspun, ¶54. AECS6 then uses “Edge Detection” to determine a “segmentation boundary” separating the foreground and background elements drawn over by the user’s strokes. AEM, 328-31; Greenspun, ¶54.

d. Puppet Effect

One available effect is AECS6’s “Puppet” effect, which “deform[s] part of an image according to the positions of pins that you place and move.” AEM, 218; Greenspun, ¶55. A user first places a Puppet “Deform” pin on the layer via the “Composition” or “Layer panel[s],” specifically on a “nontransparent pixel” in a portion of the layer to be moved. AEM, 219; Greenspun, ¶55. AECS6 then creates an “outline” by “auto-tracing the alpha channel of [the] layer”—i.e., outlining the nontransparent pixels of the layer. AEM, 219, 220-21; Greenspun, ¶55. This outline is “automatically divided into a mesh of triangles,” where “[e]ach part of the mesh is also associated with the pixels of the image, so the pixels move with the mesh.” AEM, 218; Greenspun, ¶55. Thus, when the user repositions the Deform pin, “the mesh changes shape to accommodate this movement, while keeping the overall mesh as rigid as possible,” resulting in a corresponding movement of the layer’s

nontransparent pixels. AEM, 218; Greenspun, ¶55. The user can add more Deform pins to move other portions of the layer and its mesh. AEM, 218; Greenspun, ¶55.

In one example, AEM depicts a foreground layer containing an image of a gorilla, which is stacked over a background layer containing several buildings:



AEM, 218; Greenspun, ¶56. A Deform pin has been placed on each of the gorilla's limbs, and a mesh has been created from the nontransparent pixels of the gorilla. AEM, 218; Greenspun, ¶56. As shown, the Deform pin on the gorilla's right arm is repositioned, causing that arm to move, while the other three Deform pins are not repositioned, causing the other limbs to remain "as rigid as possible" during the animation. *See* AEM, 218; Greenspun, ¶56; *see also* AEM, 219.

e. Animating with the Puppet Effect

A user not only can apply the Puppet effect to move a portion of a layer, but can also animate this movement. AEM, 218-19; Greenspun, ¶57. To do so, AEM instructs the user to define a starting time point and starting position for a Deform pin, and then defining an ending time point and ending position for the Deform pin.

AEM, 177, 219; Greenspun, ¶57. This is known as defining starting and ending “keyframes” for the Deform pin, respectively. AEM, 177, 219; Greenspun, ¶57. AECS6 then interpolates the Deform pin’s position from the starting time and position (i.e., the starting “keyframe”) to the ending time and position (i.e., the ending “keyframe”), and generates “in-between” frames based on such interpolation, resulting in an animated movement of the layer. AEM, 177, 193, 219; Greenspun, ¶57. This is known as “keyframe interpolation.” AEM, 177, 193; Greenspun, ¶57.

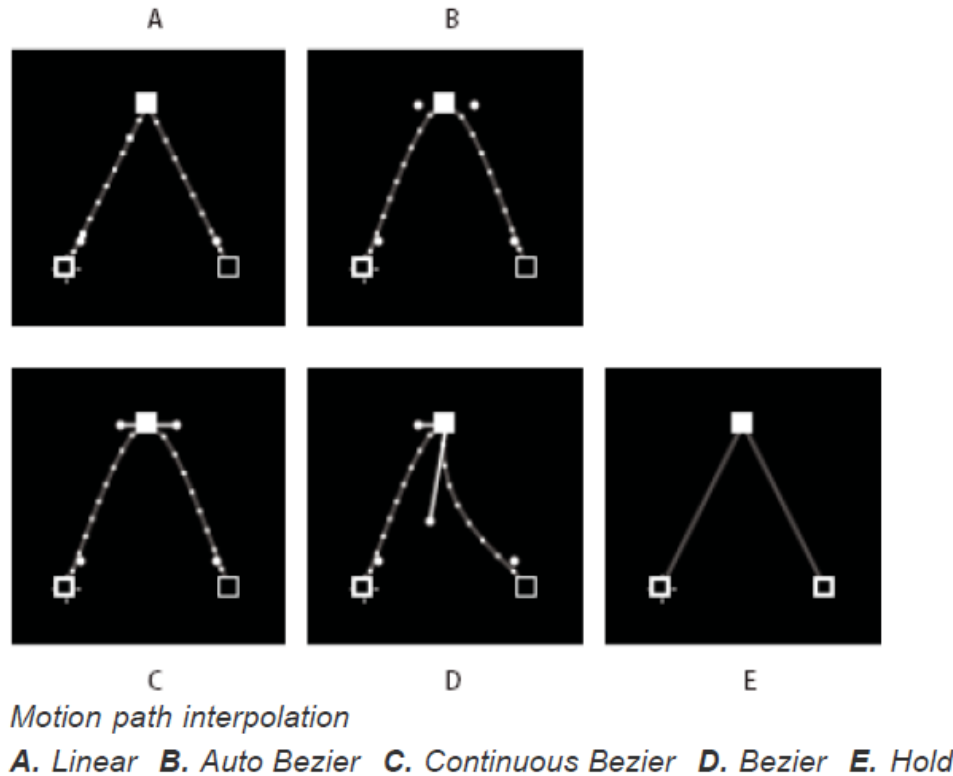
To define a starting “keyframe” for a Deform pin, AEM instructs the user to specify a time point within the animation, and then place the Deform pin on a nontransparent pixel of the layer in the “Composition” or “Layer panel[s].” AEM, 218-19; Greenspun, ¶58; *see also* AEM, 166 (describing the “current-time indicator (CTI)” in the “Timeline panel,” which is used to specify the current animation time point). AECS6 then automatically creates the starting keyframe for the Deform pin based on both the specified time point and the Deform pin’s starting position. AEM, 218-19; Greenspun, ¶58.

AEM then instructs the user to define an ending keyframe by “[g]o[ing] to another time in the composition, and mov[ing] the position of... the Deform pin[] by dragging [it] in the Composition or Layer panel.” AEM, 219; Greenspun, ¶59.

Once the keyframes for a Deform pin have been specified, AECS6 will perform keyframe interpolation by interpolating the position of the Deform pin from

the starting to ending keyframe and generating animation frames to animate the layer's corresponding movement. AEM, 193, 218-19; Greenspun, ¶60. AECS6 performs such interpolation for all keyframes of a Deform pin. AEM, 193, 219; Greenspun, ¶60. The user may place additional Deform pins to animate other parts of the layer in the same way. AEM, 219; Greenspun, ¶60.

Further, keyframe interpolation of a Deform pin's change in position follows a user-defined path—a “motion path.” AEM, 187, 219; Greenspun, ¶61. A motion path is visually indicated in both the “Composition” and “Layer panel[s]” as “a sequence of dots, where each dot marks the position of the [Deform pin] at each frame. A box in the path marks the position of a keyframe.” AEM, 187, 219; Greenspun, ¶61. A user selects between using a linear or non-linear motion path for interpolation. AEM, 194-95; Greenspun, ¶61. With a linear motion path—i.e., “Linear Interpolation”—the position of the Deform pin is interpolated linearly from the starting to ending keyframes. AEM, 194; Greenspun, ¶61. With a non-linear motion path—e.g., “Bezier interpolation”—the user can adjust the motion path to include “any combination of curves and straight lines,” and the position of the Deform pin will follow that path when interpolated from the starting to ending keyframes. AEM, 194-95; Greenspun, ¶61. AEM provides an example illustrating the available linear and non-linear motion paths:



AEM, 193; Greenspun, ¶61.

Additionally, a Deform pin's keyframe interpolation can be looped such that a layer's movement is animated in a repeating loop. This is done by applying a "loopOut()" expression to the Deform pin, which, by default, causes "all keyframes [to] loop." AEM, 562, 219; Greenspun, ¶62. Thus, after the Deform pin reaches the ending position/keyframe, its position will be reset to the starting position/keyframe, and keyframe interpolation will repeat. AEM, 562, 219; Greenspun, ¶62. This loop repeats until the user specifies otherwise. AEM, 562, 219; Greenspun, ¶62.

2. Independent Claim 1

a. [1pre]

To the extent [1pre] is limiting, AEM teaches that AECS6 is “software” that is installed on computers and run on “64-bit operating systems.” AEM, 19; Greenspun, ¶88. This necessarily discloses the claimed “computer system,” “processors,” and “computer-readable media.” Greenspun, ¶88. Indeed, AEM teaches that a user ideally should have three “local disk drives,” i.e., “one for source footage files, one from which the application runs, and one for rendered output.” AEM, 17; Greenspun, ¶88. “If you commonly work with complex source compositions, try adding RAM or a faster processor.” AEM, 38; Greenspun, ¶88.

Further, AEM describes using AECS6 to animate the movement of pixels within an imported image—the claimed “automating the shifting of pixels within a digital image.” Sections VI.A.2.b-VI.A.2.h; Greenspun, ¶89.

b. [1a]

AEM teaches importing images and other footage items into a composition in AECS6—the claimed “access... a digital image file, wherein the digital image file comprises information that corresponds to individual pixels within the digital image.” AEM, 75, 98; Greenspun, ¶90. AEM teaches that the user should have a “local disk drive” that is a “source” for such footage files—“access from memory.” AEM, 17; Greenspun, ¶90.

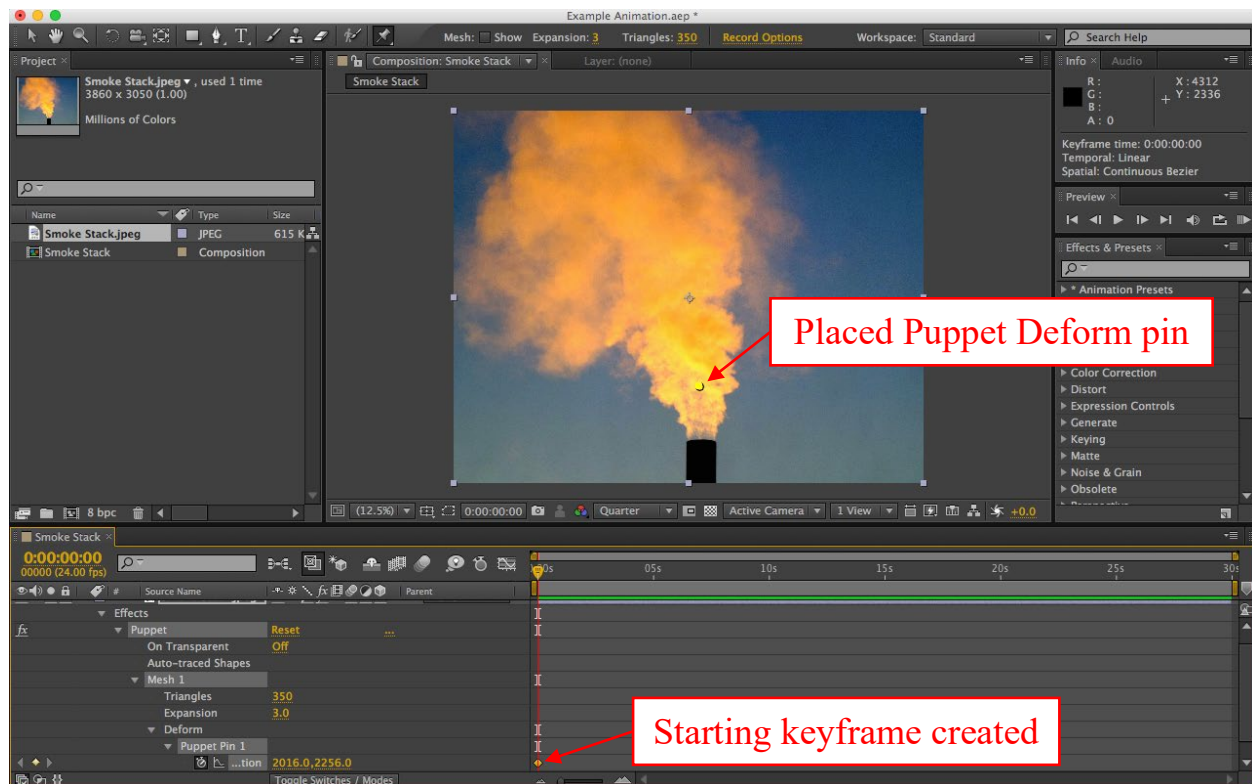
c. [1b]-[1c]

Regarding the claimed “portion of the digital image,” the ’017 Patent states the claimed “portion” can be a single pixel within the image. ’017 Patent, 6:3-5, 6:8-10, 6:14-16; Greenspun, ¶91.

AEM teaches that, when an image or other footage item is imported, the image becomes a layer of a composition—the claimed “digital image.” AEM, 75, 98; Greenspun, ¶92. AEM also teaches creating a Puppet effect animation of the layer by placing a Puppet “Deform” pin in the “Composition” or “Layer panel[s]” on a “nontransparent pixel” at a specific part of the layer that the user desires to move, causing AECS6 to create a starting “keyframe” based on both the current time within the animation and the Deform pin’s position—“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.” AEM, 218-19; ’017 Patent, 6:3-5, 6:8-1; Greenspun, ¶92. AECS6 makes this starting keyframe “visible in the Composition panel and Layer panel”—“render a first visual indication of the first starting point on the user interface.” AEM, 218; Greenspun, ¶92.

By way of further illustration, a screenshot of AECS6 is provided below⁶ showing a Deform pin placed in the “Composition panel” of an image of a smokestack, and a starting keyframe that is automatically created by the Deform Pin’s placement:

⁶ Although AECS6—including screenshots of AECS6—cannot be used as a basis of this ground because AECS6 is not a “patent[] or printed publication[]” under 35 U.S.C. §311(b), such screenshots are nevertheless “permissible evidence in an [IPR] for establishing the background knowledge possessed by a [POSITA],” including demonstrating what a POSITA would have known about AECS6’s animation capabilities, especially in light of AECS6’s teachings. *See Qualcomm Inc. v. Apple Inc.*, 24 F.4th 1367, 1376 (Fed. Cir. 2022).



Greenspun, ¶93.⁷

d. [1d]-[1e]

Once a starting keyframe for a Deform pin has been defined, AEM instructs the user to create an ending keyframe by “[g]o[ing] to another time in the composition, and mov[ing] the position of... the Deform pin[] by dragging [it] in the Composition or Layer panel”—the claimed “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.” AEM, 218-19; ’017 Patent,

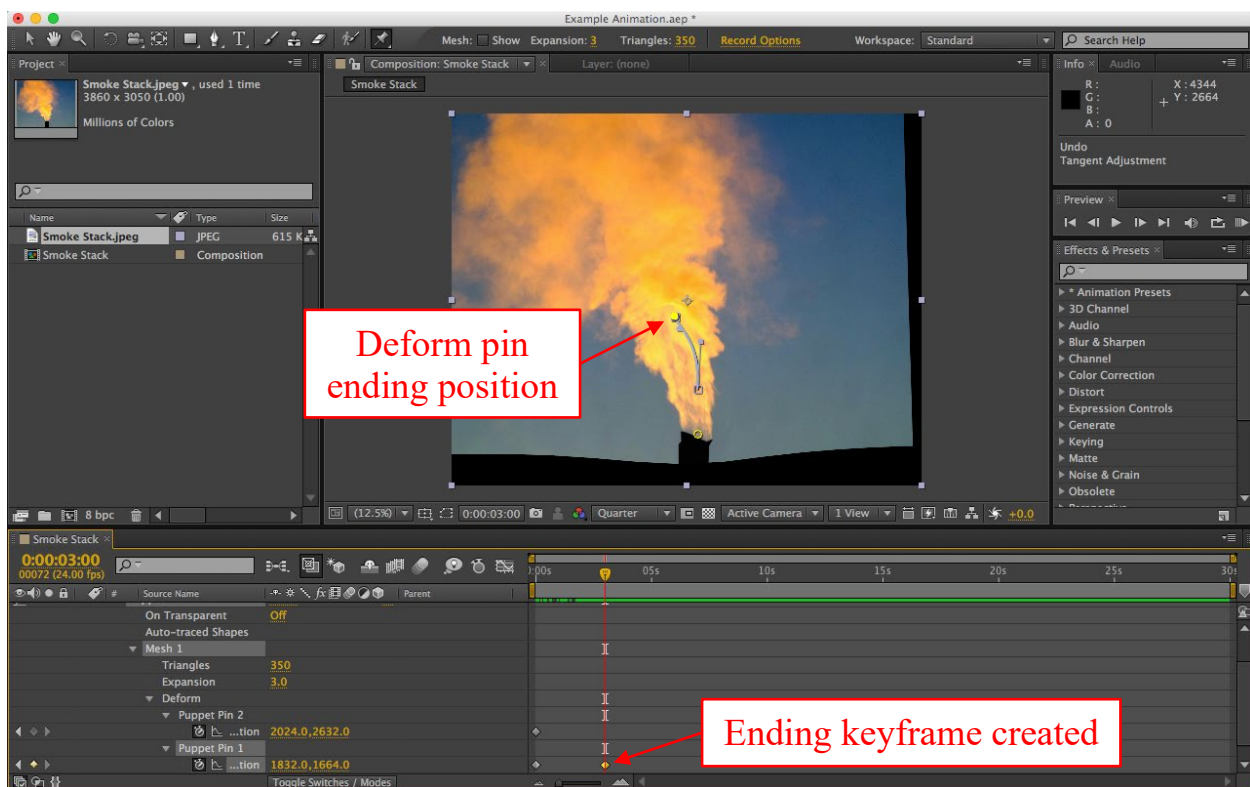
⁷ Dr. Greenspun also explains step-by-step how the screenshots here and below were created. *See, e.g.*, Greenspun, ¶¶63-87.

IPR2023-00153

U.S. Patent No. 10,346,017

6:14-16; Greenspun, ¶94. AECS6 makes this ending keyframe “visible in the Composition panel and Layer panel”—“render a second visual indication of the first ending point on the user interface.” AEM, 218; Greenspun, ¶94.

To further illustrate, the previously-discussed (in Section VI.A.2.c) annotated screenshot of AECS6 is shown again below, but now illustrating the creation of an ending keyframe for the Deform pin:



Greenspun, ¶95.

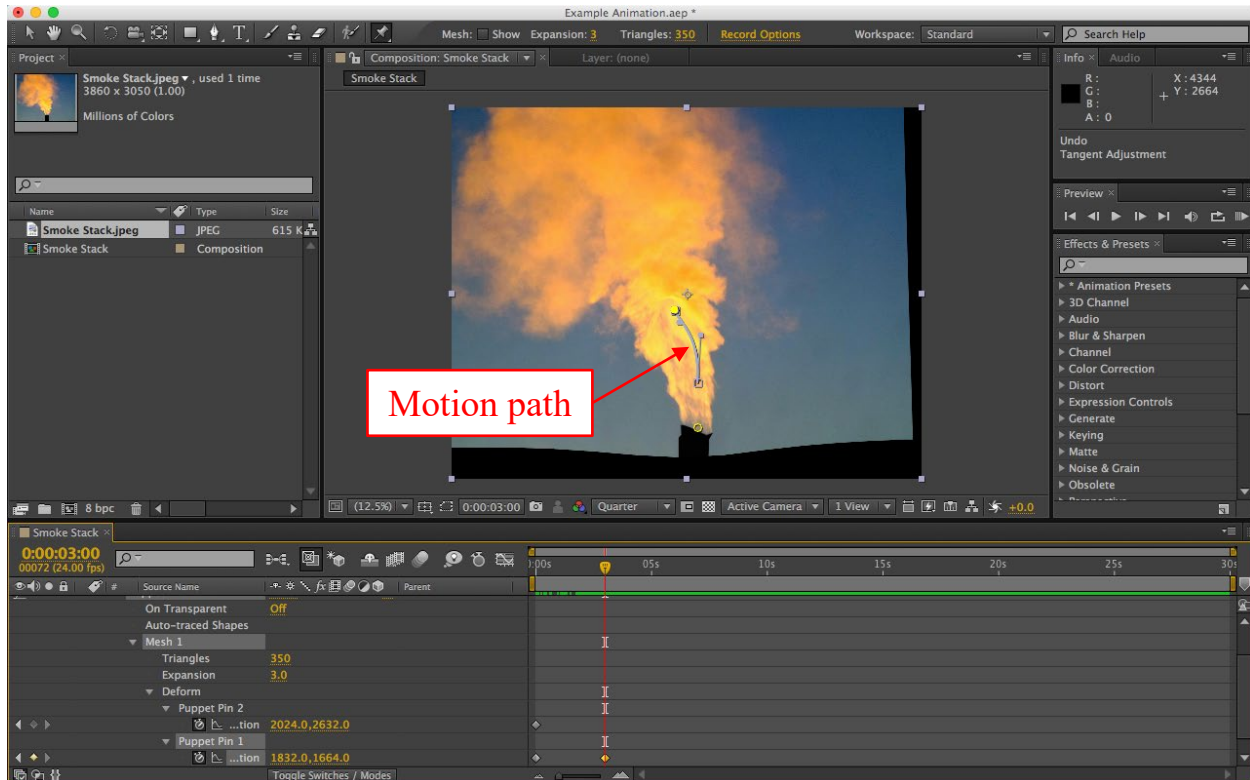
e. [1f]

AEM teaches that, when the user initially places the Deform pin to create a starting keyframe, AECS6 automatically outlines the layer’s nontransparent pixels and divides this outline into a “mesh of triangles.” AEM, 219, 220-21; Greenspun,

¶96. AEM also teaches that, when the user thereafter creates an ending keyframe by “[going] to another time in the composition and mov[ing] the position of... the Deform pin[] by dragging [it] in the Composition or Layer panel,” the shape of the layer’s mesh changes—and thus the layer’s nontransparent pixels move—to “accommodate” this repositioning. AEM, 218-19, 177; Greenspun, ¶96.

Further, the creation of an ending keyframe prompts AECS6 to perform animation via “keyframe interpolation,” where AECS6 interpolates the Deform pin’s position from the starting keyframe to the ending keyframe and generates “in-between” frames based on the interpolation, thus animating the movement of the layer’s nontransparent pixels. AEM, 177, 193, 218-19; Greenspun, ¶97. The interpolation is performed according to, and visually indicated in the “Composition” and “Layer panel[s]” by, a “motion path,” which extends from the starting position/keyframe to the ending position/keyframe. AEM, 187, 218-19; Greenspun, ¶97. The visual indication of such a motion path discloses “render, on the user interface, a first rendered link” and the accompanying language recited in [1f]. Greenspun, ¶97.

To further illustrate, the previously-discussed (in Section VI.A.2.d) screenshot of AECS6 is shown again below, but with further annotations highlighting the motion path created between the Deform pin’s starting position/keyframe and ending position/keyframe:



Greenspun, ¶98.

f. [1g]

Regarding the claimed “identify a first set of pixels,” the ’017 Patent provides additional context by stating that “[t]he size of the group of pixels may be user selectable or automatically determined.” ’017 Patent, 6:55-56; Greenspun, ¶99. For example, a user can perform the claimed “identify[ing]” of a “first set of pixels” by “increas[ing] the threshold distances that the set of pixels extend beyond the link.” ’017 Patent, 6:57-58; Greenspun, ¶99. Particularly, the user can identify certain pixels between the starting and ending points to be “covered by a mask” and excluded from the set of pixels to be shifted. ’017 Patent, 6:52-55, 4:48-52; Greenspun, ¶99. Alternatively, the software can perform the claimed “identify[ing]”

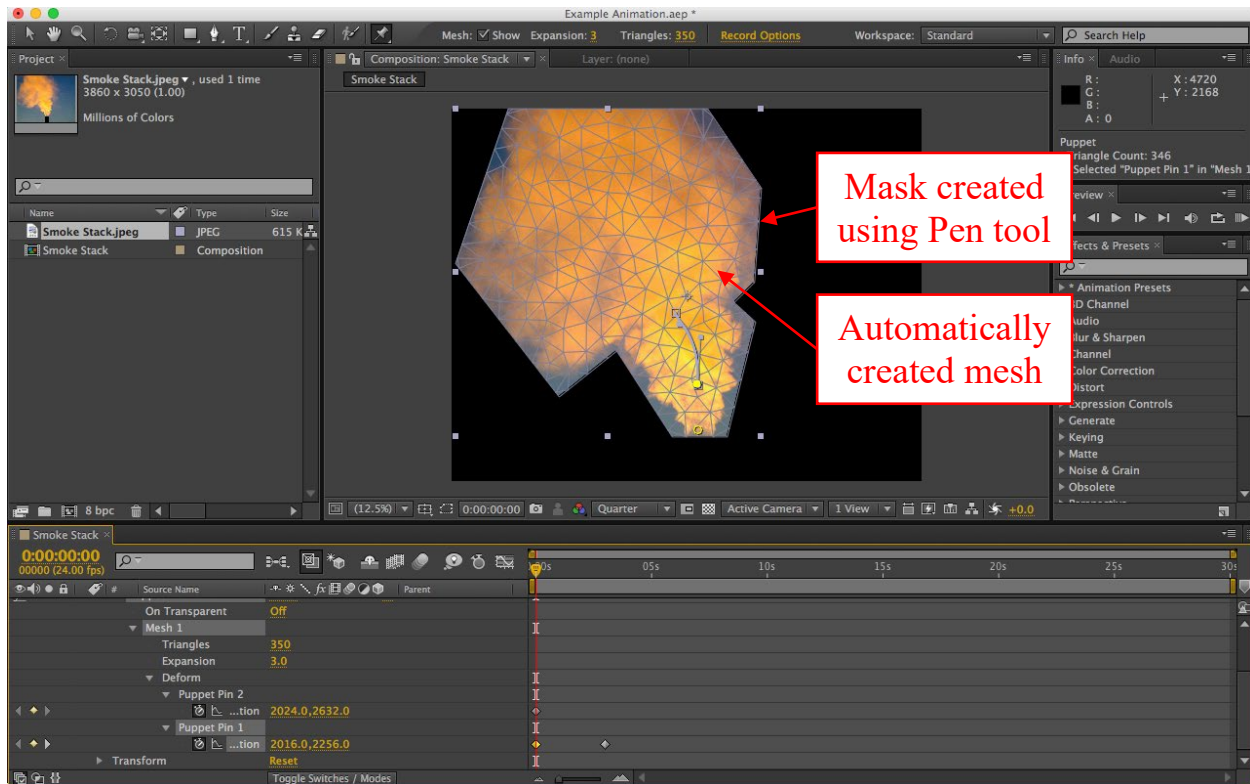
automatically by “identif[ying] sets of pixels that lie between the respective starting points and the respective ending points”—e.g., identifying pixels between the starting and ending points that are or are not “covered by a mask.” ’017 Patent, 6:49-56; Greenspun, ¶99.

AEM teaches selecting certain pixels of the layer to be transparent or nontransparent (i.e., modifying the “alpha channel” of the layer) by applying either a “mask” or a “matte,” where the former is created using, e.g., the “Pen” tool or “Auto-trace” function, and the latter is created using, e.g., the “Roto Brush” tool. AEM, 315, 318; Greenspun, ¶100.

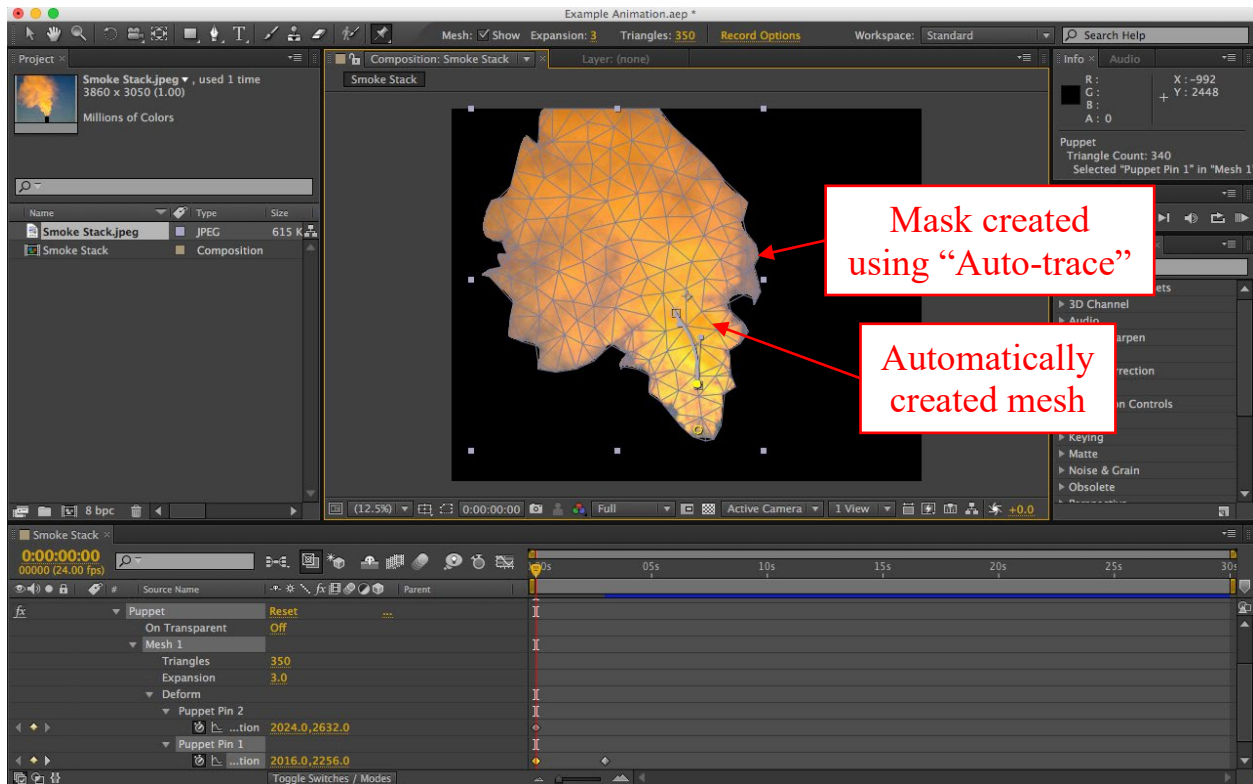
Thus, as AEM teaches, AECS6 necessarily allows a user to apply either a mask or matte to select any or all pixels of the layer to be nontransparent and thus included in the layer’s mesh to be moved and animated, including pixels parallel to the Deform pin’s motion path—the claimed “parallel to the first rendered link”—within a user-defined threshold distance from the motion path—“within a user-defined threshold distance from the first rendered link”—and extending from the Deform pin’s starting position/keyframe to the ending position/keyframe—“extending from the first starting point to the first ending point.” AEM, 218-19, 315, 194-95; ’017 Patent, 6:52-56 (“The size of the group of pixels may be user selectable....”); Greenspun, ¶101. Further, AECS6 automatically creates the mesh from the layer’s nontransparent pixels, which additionally discloses [1g]. AEM,

218-19, 194-95; '017 Patent, 6:52-56 (“The size of the group of pixels may be... automatically determined.”); Greenspun, ¶101.

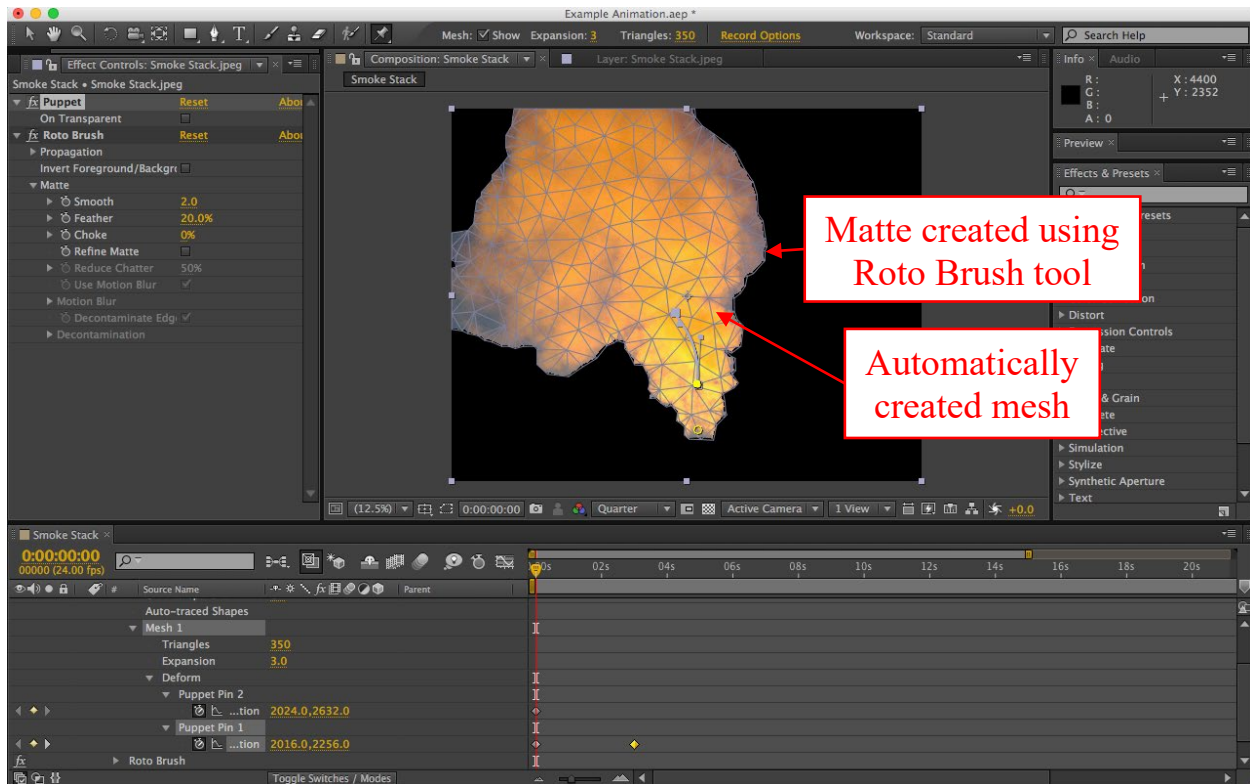
Separately, a POSITA would have found obvious selecting and making nontransparent, using a mask or matte, those pixels parallel to the Deform pin’s motion path, within a user-defined threshold distance from the motion path, and extending from the Deform pin’s starting to ending position, because such pixels would in common instances be pixels that the user desires to include in the mesh to be moved and animated. Greenspun, ¶102. For instance, to move and animate the smoke rising from the smokestack depicted in Section VI.A.2.e’s screenshot using the Puppet effect, a POSITA would have recognized to apply a Deform pin within the smoke, create for the Deform pin a motion path rising upwards through the smoke, and ensure the smoke’s pixels are included in the mesh to be moved by selecting and making nontransparent all pixels of the smoke using, e.g., a mask created with the Pen tool, as shown below:



Id.; cf. Okabe, 7, 2 (depicting a similar instance in Figure 1, which depicts user-drawn “strokes” going downwards through a waterfall, and a “matte” specifying a to-be-animated “region of interest” (in white) that includes all pixels of the waterfall, including those parallel to each stroke, within a user-defined threshold distance from the strokes to the waterfall’s edges, and extending from the starting point to the ending point of each stroke); Section VI.B.4.f; Greenspun, ¶102 n.9. As shown, the mask makes nontransparent, and thus the mesh includes, pixels parallel to the Deform pin’s motion path, within a user-defined threshold distance from the motion path (i.e., to the smoke’s edges), and extending from the Deform pin’s starting to ending position. Greenspun, ¶102. The following additionally illustrates how a similar mask and mesh is created using the “Auto-trace” function:



Id. Further, the following shows how a similar result can be achieved by creating a matte using the “Roto Brush” tool:



Id.

Such is also consistent with Plotagraph’s view of [1g]. According to Plotagraph’s infringement contentions in the Litigation, [1g]⁸ is infringed by Petitioner’s “Motionleap” app when the user, after “using the Direction tool” to draw the claimed link, “defines the threshold distance from the first digital link by using the Select tool to paint an area to be animated,” where the area also includes pixels

⁸ Plotagraph’s assertions are directed to claim 11 of the related U.S. Patent No. 10,621,469, which recites “selecting a first set of pixels” identical to that in [1g]. EX1022, 10-11; Greenspun, ¶103 n.10.

parallel to the link and extending from the link's starting to ending point. EX1022,⁹ 9-11; Greenspun, ¶103. Thus, according to Plotagraph, a user's selection of a to-be-animated area that includes pixels surrounding the link meets [1g], where the distance from the link to the area's boundary meets the specifically recited "user-defined threshold distance." EX1022, 10-11; Greenspun, ¶103. And AEM selecting pixels in this same way using AECS6's "Pen" tool, "Auto-trace" function, and "Roto Brush" tool, as discussed above. AEM, 315, 318; Greenspun, ¶103.

g. [1h]

Regarding the claimed "shifting," the '017 Patent states that the pixel shifting can use a "mesh algorithm" that "triangulates the mesh using defined points and then calculates an affine transformation for every angle." '017 Patent, 8:28-37; Greenspun, ¶104. The '017 Patent's Figure 4 depicts an example, where "a starting mesh and ending mesh [are] overlaid on [a] digital image of [a] house and landscape":

⁹ Although the contentions purport to be "CONFIDENTIAL," they in fact are not because Plotagraph submitted these contentions as a public exhibit to its publicly filed motion for leave. *See* EX1022, 2 n.1.

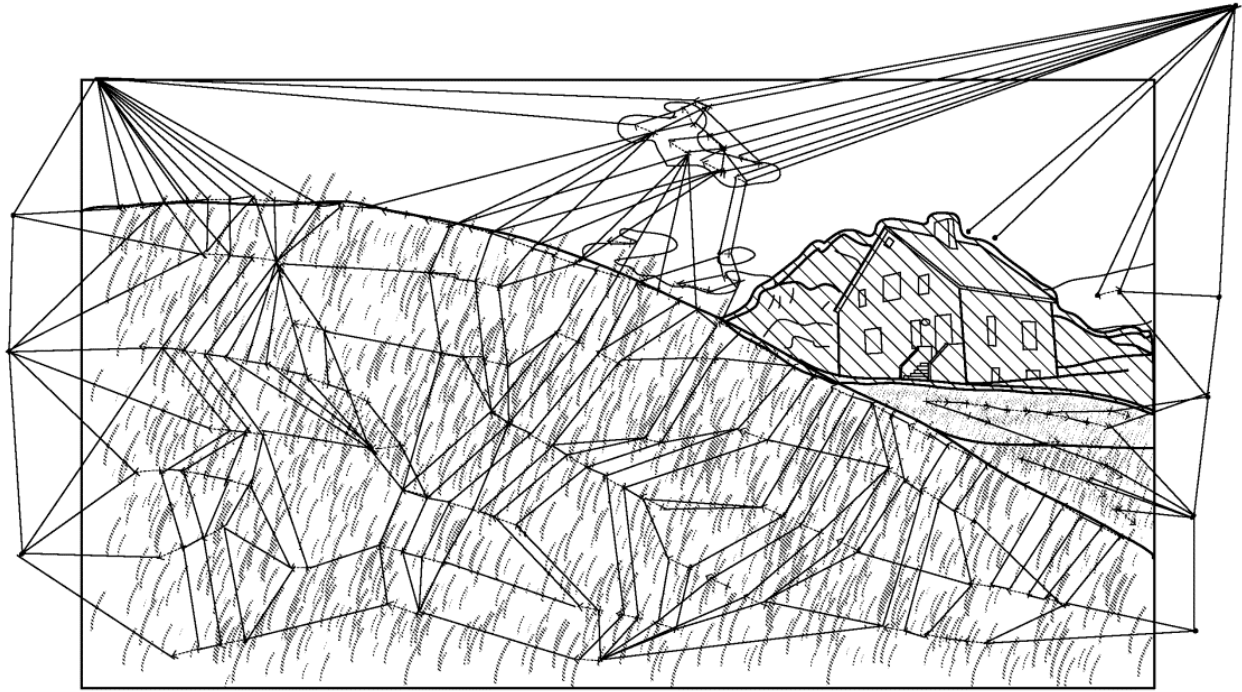


Fig. 4

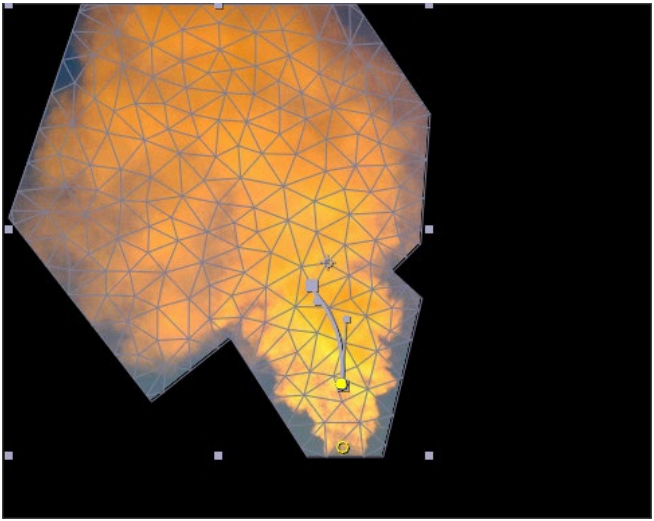
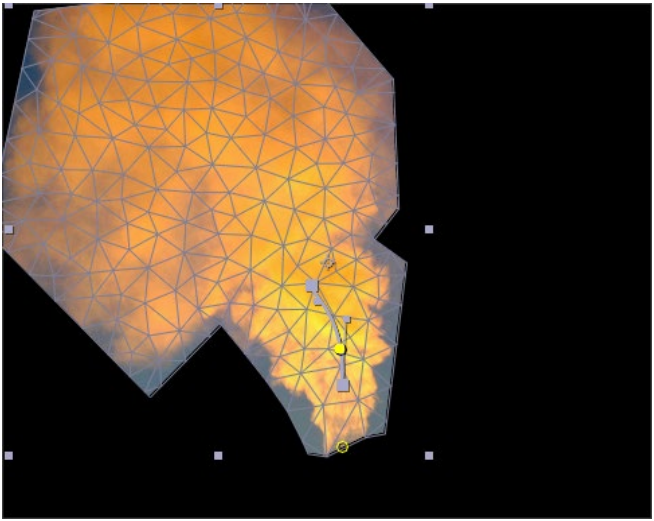
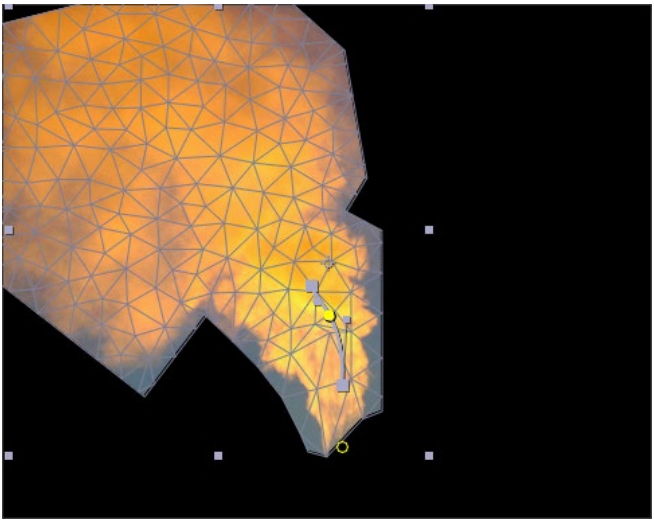
'017 Patent, Fig. 4, 6:39-48; Greenspun, ¶104.

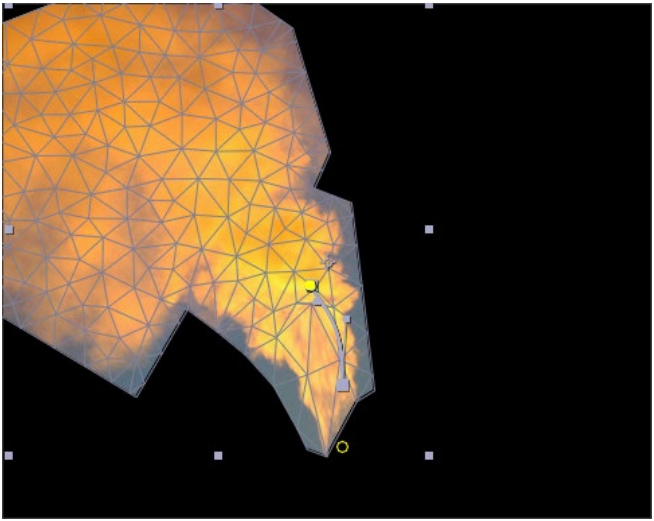
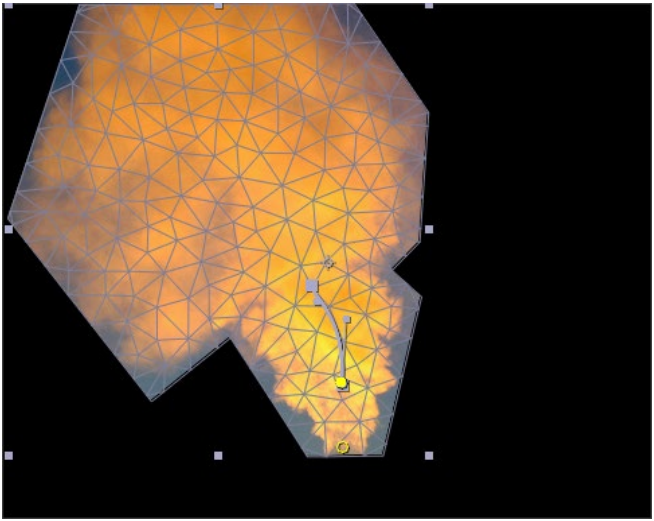
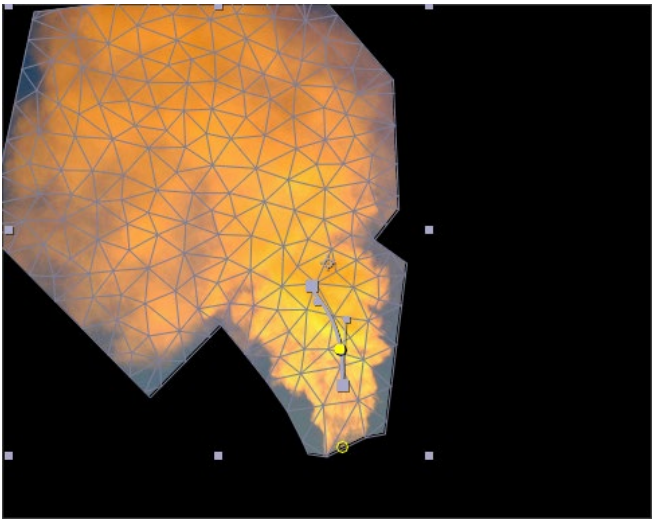
AEM similarly teaches that, when a user places an initial Deform pin, AECS6 creates a mesh from the layer's nontransparent pixels. AEM, 219, 220-21; Section VI.A.2.e; Greenspun, ¶105. AEM also teaches, and a POSITA would have found obvious, selecting and making nontransparent, using a mask or matte, pixels parallel to the Deform pin's motion path, within a user-defined threshold distance from the motion path, and extending from the Deform pin's starting position/keyframe to the ending position/keyframe—the claimed “first set of pixels.” AEM, 218-19, 315, 194-95; Section VI.A.2.f; Greenspun, ¶105. When a motion path for the Deform pin is created, AECS6 animates the corresponding change to the shape of the layer's mesh, and thus the movement of the layer's nontransparent pixels accommodating

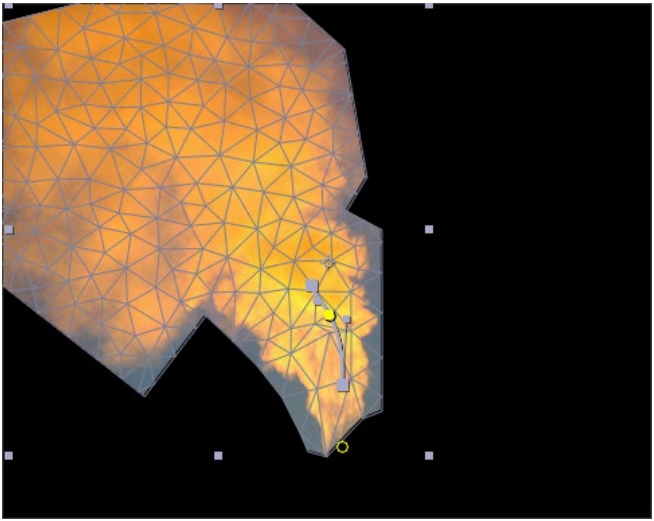
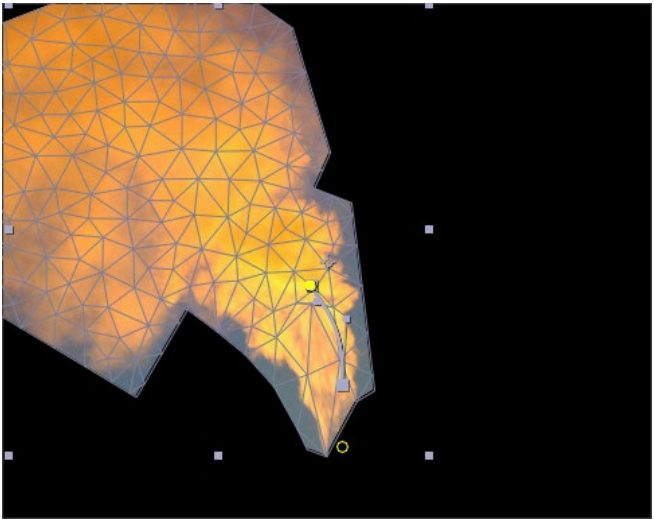
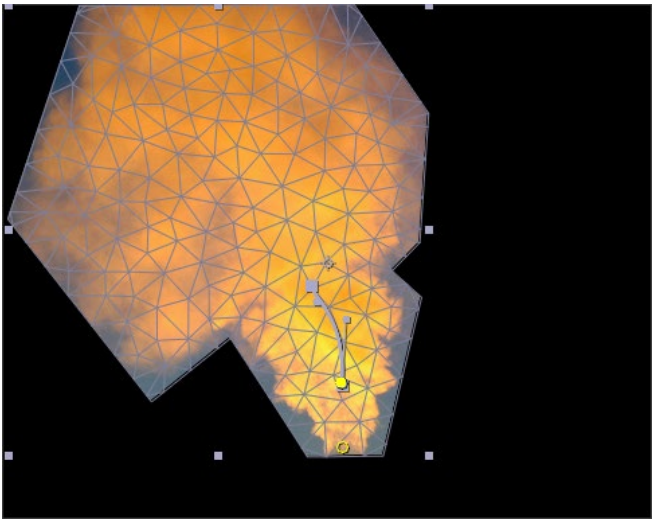
the Deform pin’s repositioning—“automatically shift the first set of pixels in the first direction.” AEM, 218-19, 194-95; ’017 Patent, 8:34-37; Section VI.A.2.e; Greenspun, ¶105.

Further, AEM teaches applying a “loopOut()” expression to the Deform pin, which, by default, causes “all keyframes [to] loop.” AEM, 562, 219; Greenspun, ¶106. Specifically, by applying a default “loopOut()” expression to a Deform pin, the Deform pin’s position will be reset to the starting keyframe upon reaching the ending keyframe during keyframe interpolation, and the interpolation will be repeatedly looped then and thereafter until otherwise specified by the user—“rendering and re-rendering in a loop the first set of pixels being shifted.” AEM, 562, 219; Greenspun, ¶106.

The following table includes a series of cropped screenshots of AECS6 showing the result of applying a “loopOut()” expression to the exemplary composition in Section VI.A.2.f’s first screenshot. Greenspun, ¶107, Attachment B. Specifically, the screenshots show the exemplary composition at different, sequential time points, mimicking the playing animation:

Animation Time Point	Composition Frame
(0:00:00:00)	 A composition frame showing a stylized, orange and yellow abstract shape with a triangular mesh overlay. The shape is set against a black background. A small yellow circle is visible near the bottom right of the shape.
(0:00:01:00)	 A composition frame showing the same stylized, orange and yellow abstract shape with a triangular mesh overlay. The shape is set against a black background. A small yellow circle is visible near the bottom right of the shape.
(0:00:02:00)	 A composition frame showing the same stylized, orange and yellow abstract shape with a triangular mesh overlay. The shape is set against a black background. A small yellow circle is visible near the bottom right of the shape.

Animation Time Point	Composition Frame
(0:00:02:23)	 A composition frame showing a stylized, orange and yellow map of a landmass with a grid overlay. A yellow line with a dot at its end is positioned on the right side of the map. A small yellow circle is located below the map. The background is black with several small white squares.
(0:00:03:01)	 A composition frame showing a stylized, orange and yellow map of a landmass with a grid overlay. A yellow line with a dot at its end is positioned on the right side of the map. A small yellow circle is located below the map. The background is black with several small white squares.
(0:00:04:00)	 A composition frame showing a stylized, orange and yellow map of a landmass with a grid overlay. A yellow line with a dot at its end is positioned on the right side of the map. A small yellow circle is located below the map. The background is black with several small white squares.

Animation Time Point	Composition Frame
(0:00:05:00)	
(0:00:05:23)	
(0:00:06:01)	

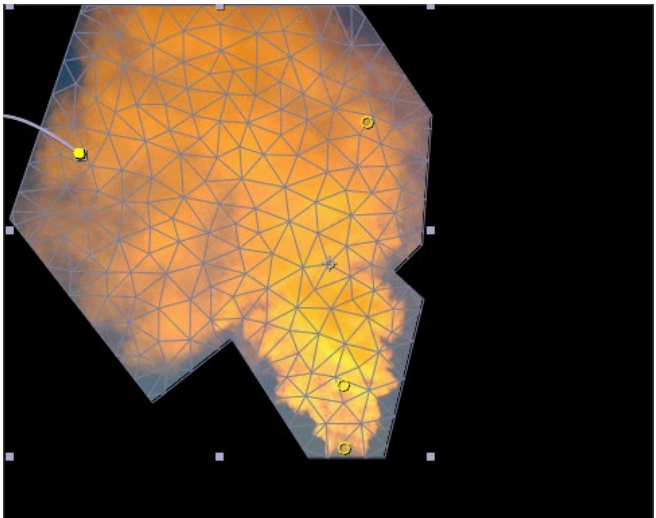
Greenspun, ¶107, Attachment B.

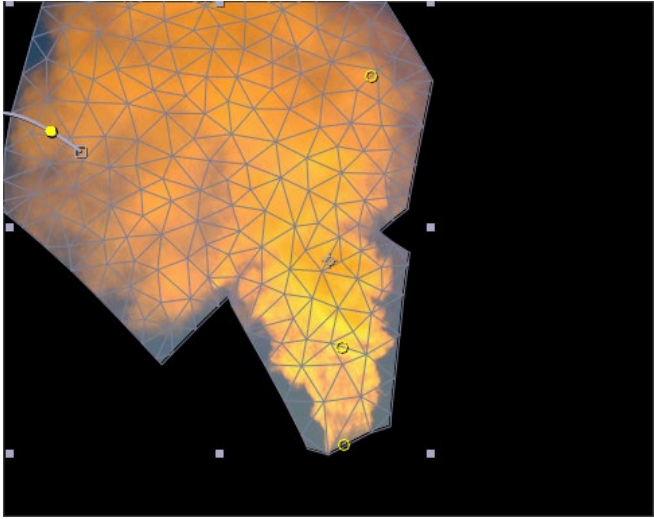
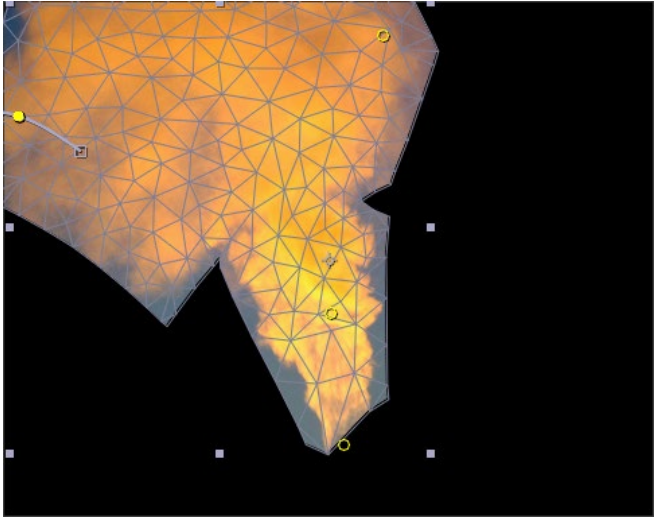
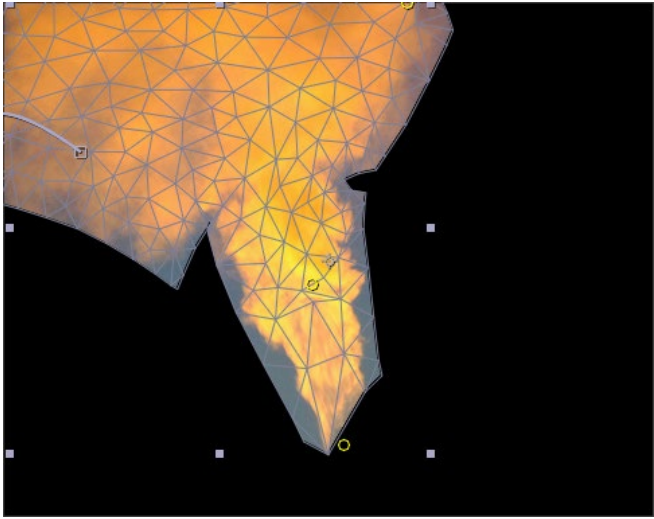
h. [1i]-[1o]

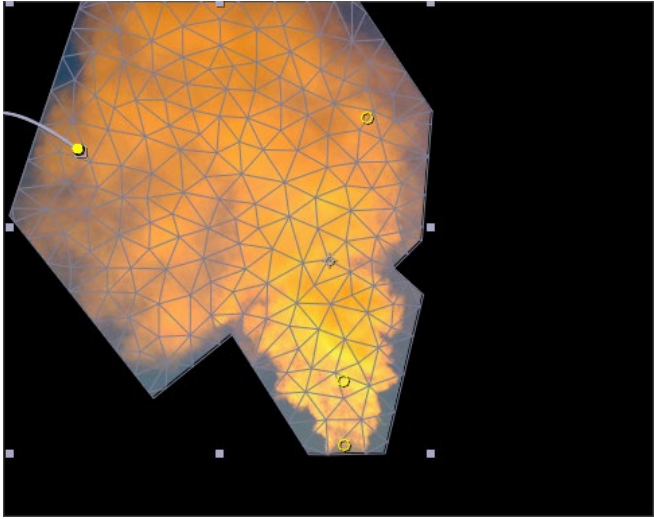
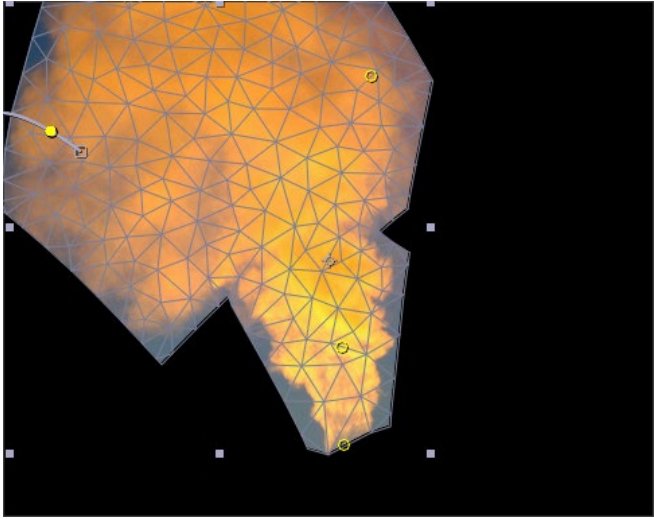
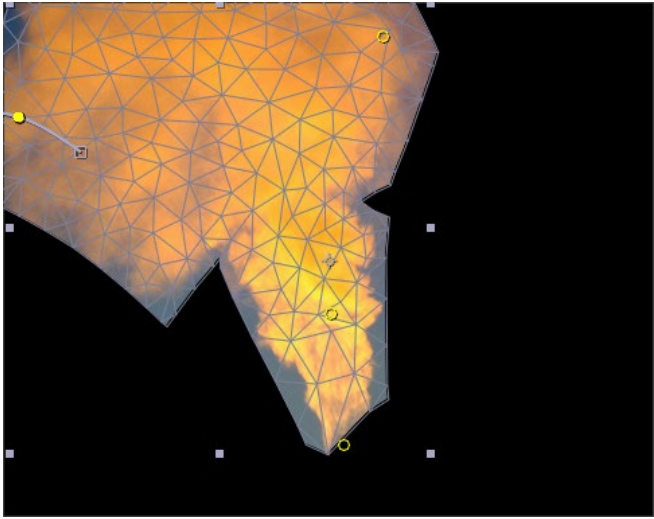
AEM teaches how to move and animate different parts of a layer through the use of multiple Deform pins. AEM, 218-19; Greenspun, ¶108. Like when placing a first Deform pin (*see* Section VI.A.2.c), a user places additional Deform pins and creates for each a starting keyframe visible in the Composition and Layer panels by simply “[c]lick[ing] in one or more places within the outline,” thus disclosing [1i]-[1j]. AEM, 219; Greenspun, ¶108. The user creates different ending keyframes and motion paths for these additional Deform pins in the same way as for the first Deform pin, thus disclosing [1k]-[1m]. AEM, 218-19; Sections VI.A.2.d-VI.A.2.e; Greenspun, ¶108. Further, for similar reasons discussed in Section VI.A.2.f, AEM teaches, and a POSITA would have found obvious, creating a mask or matte to select or and all pixels of the layer to be nontransparent and thus included in the layer’s mesh to be moved and animated, including pixels parallel each corresponding Deform pin’s motion path, within the user-defined threshold distance from the first Deform pin’s motion path (e.g., to the smoke’s edges in Section VI.A.2.f’s exemplary compositions), and extending from the starting position/keyframe to the ending position/keyframe of each Deform Pin, thus meeting [1n]. AEM, 218-19, 315, 194-95; Greenspun, ¶108; *cf.* Okabe, 7, 2 (depicting, in Figure 1, not just one but three user-drawn “strokes” going downwards through the waterfall, where the

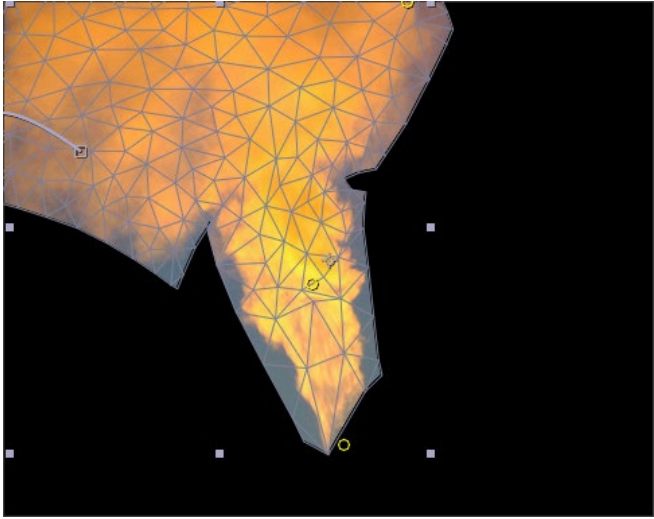
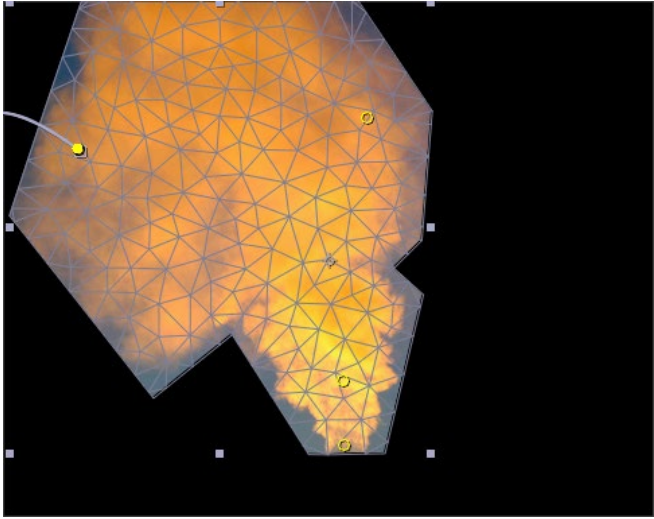
to-be-animated “region of interest” includes all pixels of the waterfall, including those parallel to each stroke, within the user-defined threshold distance from the strokes to the waterfall’s edges, and extending from the starting point to the ending point of each stroke); Section VI.B.4.h; Greenspun, ¶108 n.12; *accord* EX1022, 10-11. Finally, as similarly discussed in Section VI.A.2.g, AEM teaches that AECS6 animates the change to the shape of the layer’s mesh, and thus the movement of the layer’s nontransparent pixels, that accommodates each Deform pin’s repositioning, which is repeatedly looped by applying a “loopOut()” expression to each Deform pin, thus disclosing [1o]. AEM, 218-19, 194-95, 562; Greenspun, ¶108.

To further illustrate the foregoing, the table below provides a series of cropped screenshots of AECS6 like those in Section VI.A.2.g, but the exemplary composition now includes multiple Deform pins in addition to the first:

Animation Time Point	Composition Frame
(0:00:00:00)	

Animation Time Point	Composition Frame
(0:00:01:00)	 A composition frame showing a stylized, orange and yellow map of a landmass with a triangular grid overlay. The map is set against a black background. Several small yellow circles and a white line are visible on the map.
(0:00:02:00)	 A composition frame showing a stylized, orange and yellow map of a landmass with a triangular grid overlay. The map is set against a black background. Several small yellow circles and a white line are visible on the map.
(0:00:02:23)	 A composition frame showing a stylized, orange and yellow map of a landmass with a triangular grid overlay. The map is set against a black background. Several small yellow circles and a white line are visible on the map.

Animation Time Point	Composition Frame
(0:00:03:01)	 A composition frame showing a stylized, orange and yellow abstract shape with a triangular mesh overlay. The shape is set against a black background. Several small yellow circles are visible on the shape, and a white line segment is on the left side.
(0:00:04:00)	 A composition frame showing a stylized, orange and yellow abstract shape with a triangular mesh overlay. The shape is set against a black background. Several small yellow circles are visible on the shape, and a white line segment is on the left side.
(0:00:05:00)	 A composition frame showing a stylized, orange and yellow abstract shape with a triangular mesh overlay. The shape is set against a black background. Several small yellow circles are visible on the shape, and a white line segment is on the left side.

Animation Time Point	Composition Frame
(0:00:05:23)	 A composition frame showing a stylized, orange and yellow abstract shape with a triangular mesh overlay, set against a black background. The shape has a jagged, organic form with several small, dark, circular elements scattered around it.
(0:00:06:01)	 A composition frame showing a stylized, orange and yellow abstract shape with a triangular mesh overlay, set against a black background. The shape is similar to the one in the previous frame but has a different internal structure and a different arrangement of small, dark, circular elements.

Greenspun, ¶109, Attachment C.

3. Claim 2

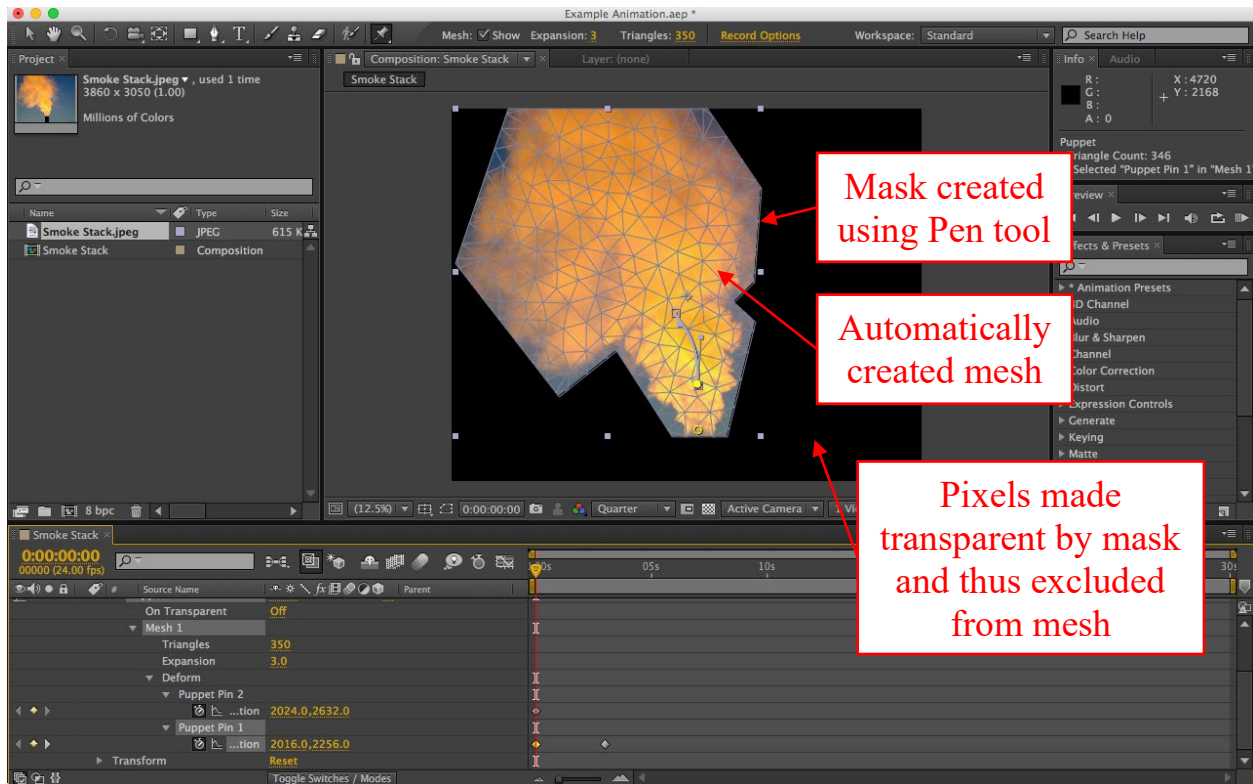
a. [2a]-[2b]

AEM teaches using a mask or matte to select pixels to be nontransparent and thus included in the layer's mesh to be moved and animated by the Puppet effect. AEM, 315, 218-19; Section VI.A.2.f; Greenspun, ¶110. Inherently, such a mask or

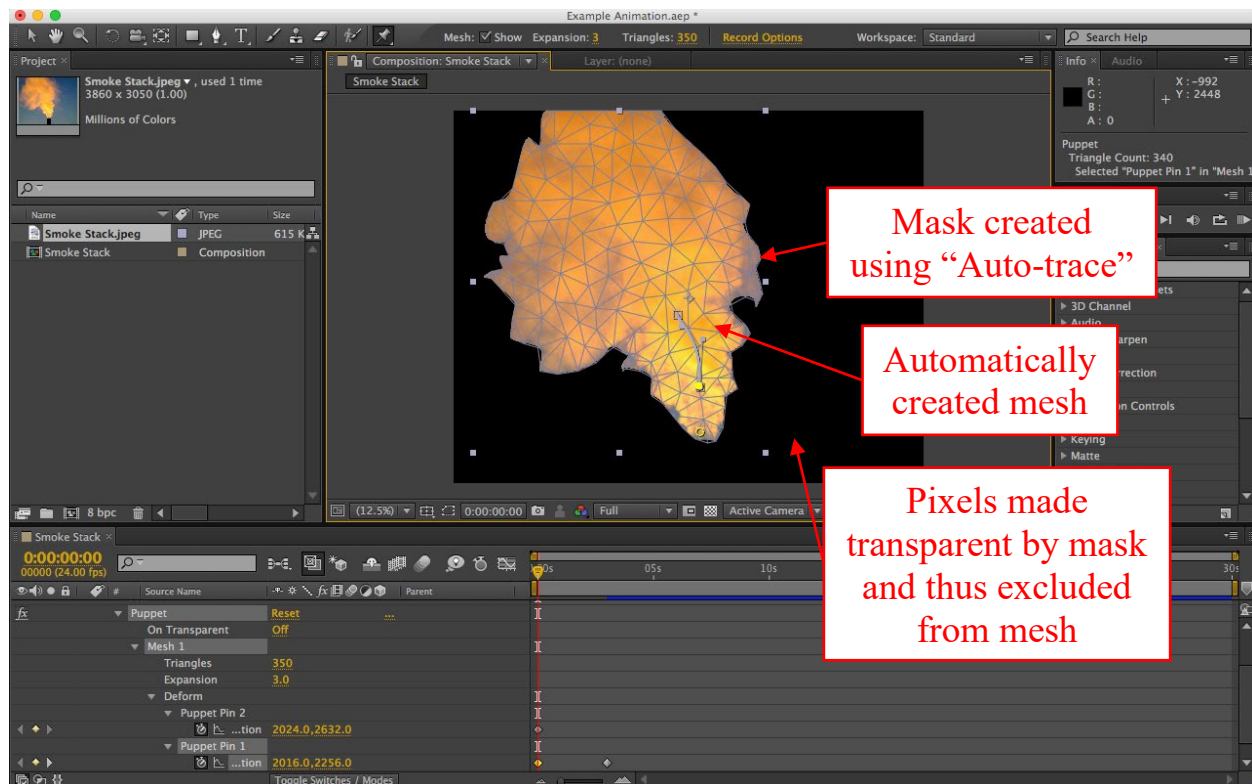
matte is therefore also used to select one or more pixels to be transparent and thus *excluded* from the layer's mesh so as to *not* be moved and animated—the claimed “mask [that] prevents pixels covered by the mask from being shifted.” Greenspun, ¶110; *see also* AEM, 104, 318, 328.

AEM also teaches creating a mask using the “Pen” tool by clicking on different points on the layer in the “Composition” or “Layer panel[s]” to specify the path and vertices of the mask, or the “Auto-trace” function to “search[] for edges” across the entire layer and create a mask tracing such edges, both of which disclose the claimed “identify a particular portion of the digital image to mask.” AEM, 262, 264-65; Greenspun, ¶111. As for a matte, AEM teaches creating a matte using the “Roto Brush” tool to draw “strokes” in the “Layer panel” over foreground and background elements to be separated, which likewise discloses the claimed “identify[ing].” AEM, 328; Greenspun, ¶111.

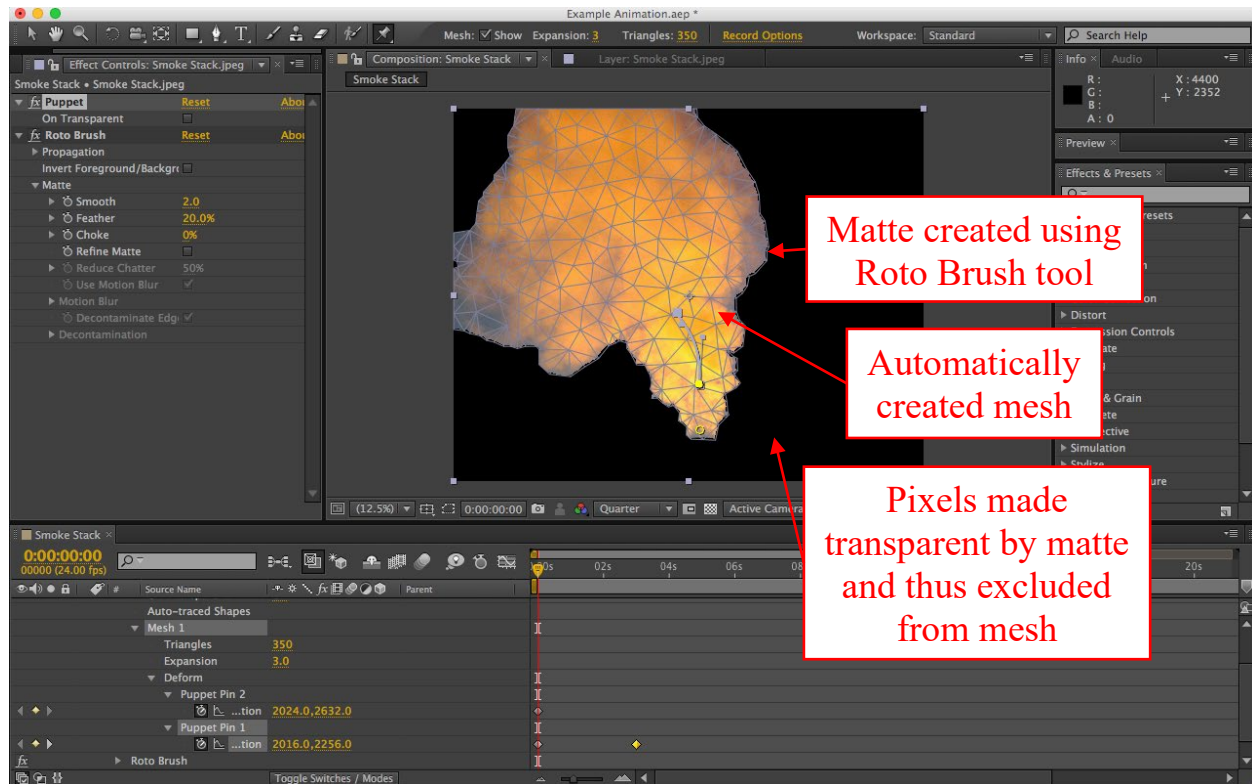
Section VI.A.2.f's first screenshot depicting a mask created using the “Pen” tool is shown below, but now indicating pixels of the layer made transparent by the mask and thus excluded from the mesh:



Greenspun, ¶112. Section VI.A.2.f's second screenshot, which depicts a mask created using the "Auto-trace" function, is shown below, but now again indicating pixels of the layer excluded from the mesh by the mask:



Id. Section VI.A.2.f's third screenshot, which shows a matte created using the "Roto Brush" tool, is provided below, but now indicating pixels of the layer excluded from the mesh by the matte:



Id.

4. Claim 3

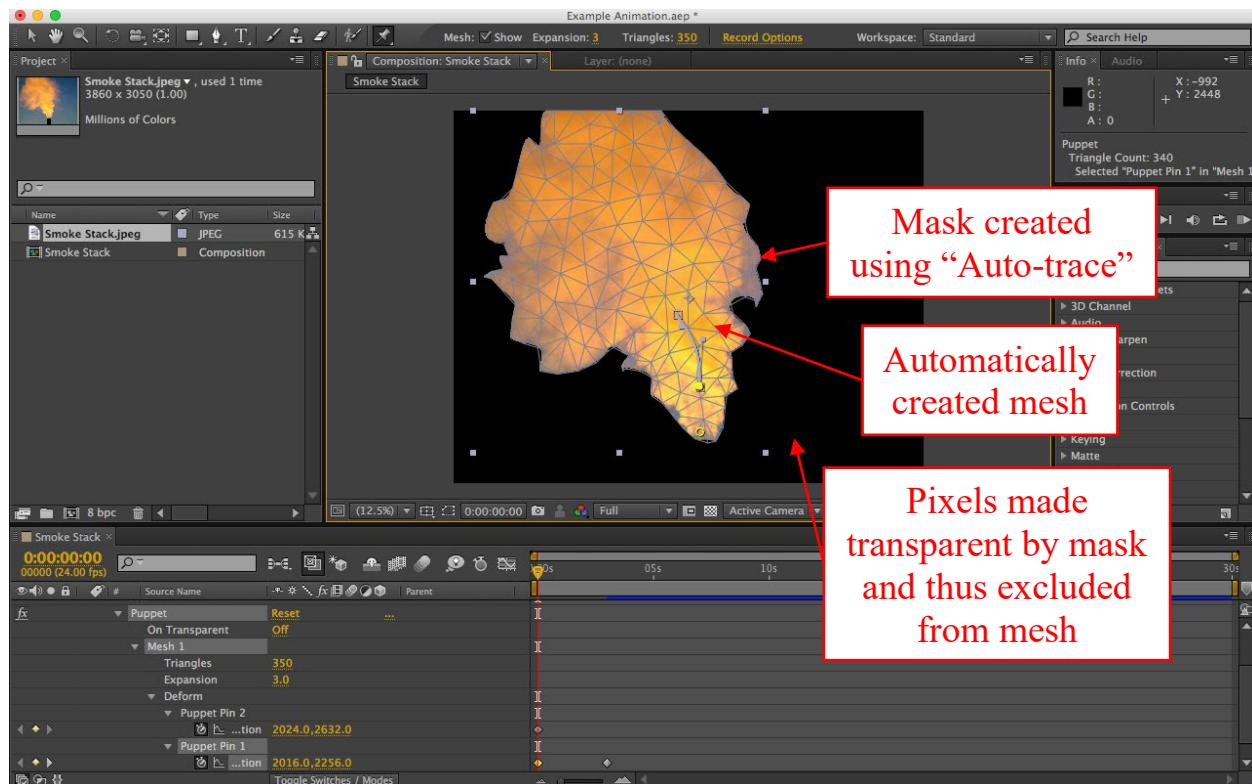
AEM teaches selecting, in the “Composition” or “Layer panel[s],” one or more pixels to be transparent and thus excluded from a layer’s mesh so as to not be moved and animated with the Puppet effect—the claimed “wherein receiving an indication of a particular portion of the digital image to mask comprises receiving through a user interface a selection of a particular pixel within the digital image”—by creating a mask using the “Pen” tool to specify the path and vertices of the mask, creating a mask using the “Auto-trace” function to “search[] for edges” across the entire layer and create a mask tracing such edges, or creating a matte using the Roto

Brush tool to draw over foreground and background elements to be separated. AEM, 318, 262, 264-65, 328; Section VI.A.3.a; Greenspun, ¶113.

5. Claim 4

a. [4a]-[4c]

AEM teaches that a mask is automatically created for a layer when using AECS6's "Auto-trace" function—the claimed "generate the mask." AEM, 262; Greenspun, ¶114. This "Auto-trace" function "search[es] for edges" across the entire layer based on the value of each pixel's "alpha, red, green, blue, or luminance channel"—"identifying one or more edges that form a boundary around the particular pixel"—and generates a mask that outlines such edges—"generating the mask to cover area within the boundary." AEM, 262; Greenspun, ¶114; *see also* AEM, 104, 318, 328. Indeed, Section VI.A.3.a's second screenshot shows that the mask traces the luminance edge between the billowing smoke and the sky, where only the smoke's pixels are nontransparent:



Greenspun, ¶114.

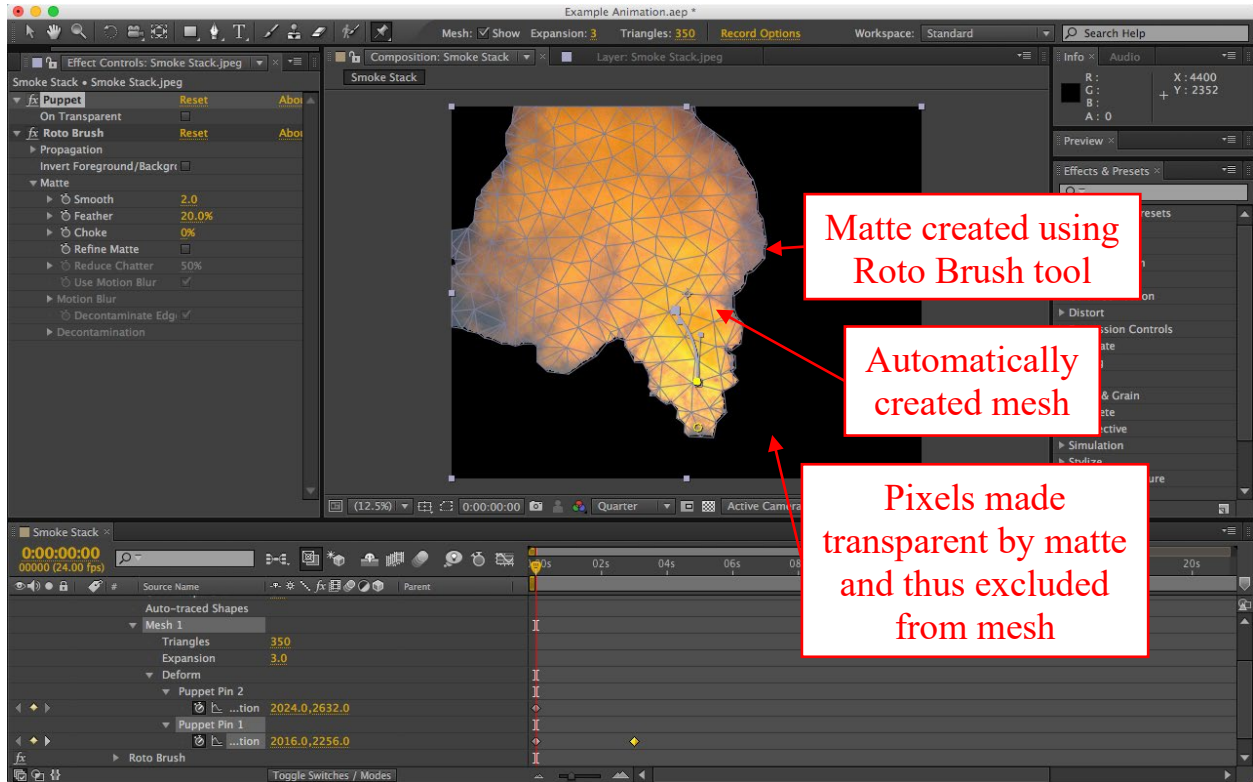
Further, AEM teaches creating a matte using the “Roto Brush” tool, which, similar to the Auto-trace function (*see* AEM, 262), uses “Edge Detection” to detect a “segmentation boundary” that separates a layer’s foreground and background elements based on “strokes” drawn by the user over “representative areas of the foreground and background elements.” AEM, 328-31; Greenspun, ¶115. This additionally discloses the claimed “generate the mask” and “identifying one or more edges that form a boundary around the particular pixel.” Greenspun, ¶115; *see also* ’017 Patent, 5:1-22. Further, as AEM teaches, AECS6 applies a matte that isolates the foreground from the background, thereby making the latter transparent, and thus

IPR2023-00153

U.S. Patent No. 10,346,017

AEM discloses the claimed “generating the mask to cover area within the boundary.”

AEM, 328, 318; Greenspun, ¶115. Indeed, Section VI.A.3.a’s third screenshot shows the creation of a matte using the “Roto Brush” tool:



Greenspun, ¶115. As shown, the segmentation boundary between the foreground (the billowing smoke) and background (the sky, which has been made transparent by the matte) traces the edge between the billowing smoke and the sky. *Id.*

6. Claim 5

AEM teaches creating different motion paths for each Deform pin placed on a layer, thereby disclosing the claimed language. AEM, 218-19; Section VI.A.2.h; Greenspun, ¶116. Indeed, as the example in Section VI.A.2.h’s table shows, each Deform pin has a separate motion path that differs from the others. Greenspun, ¶116.

7. Claim 6

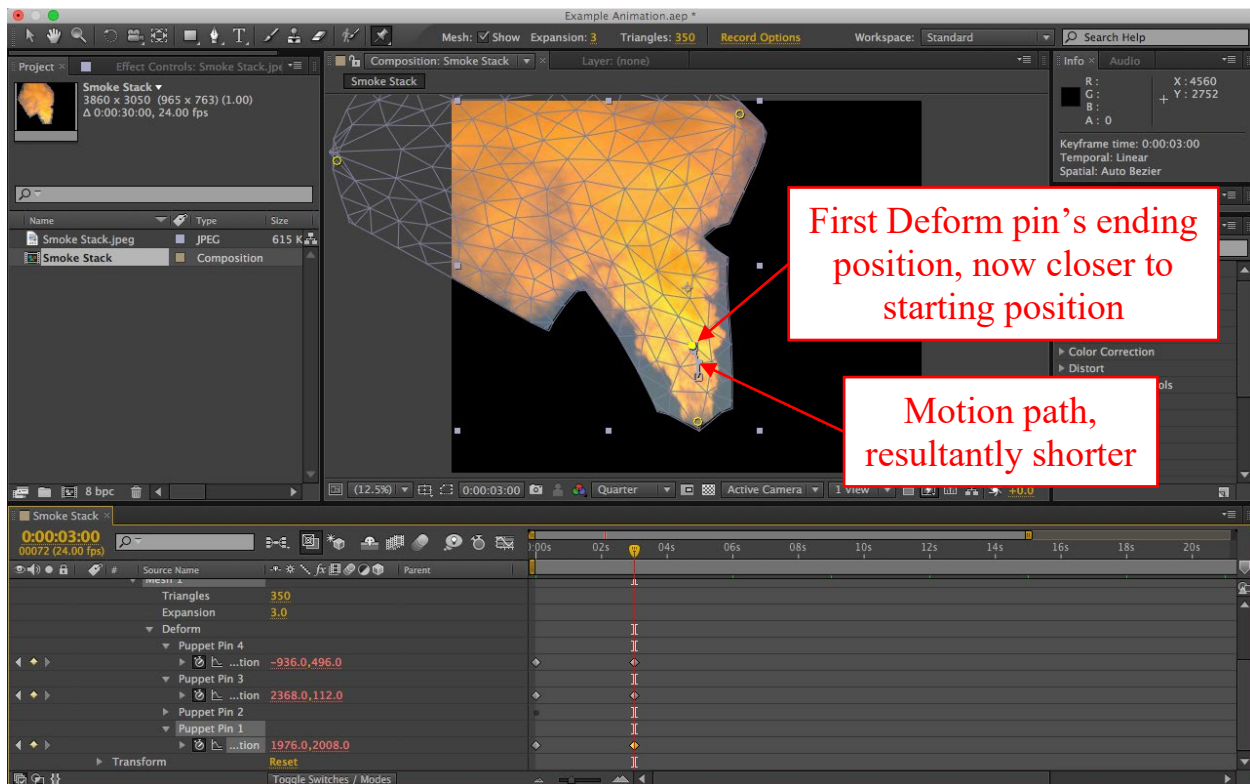
AEM teaches, after a user defines a starting keyframe for a Deform pin and a mesh of the layer's nontransparent pixels is automatically created, creating an ending keyframe by “[g]o[ing] to another time in the composition, and mov[ing] the position of... the Deform pin[] by dragging [it] in the Composition or Layer panel.” AEM, 218-19; Sections VI.A.2.d-VI.A.2.e; Greenspun, ¶117. This repositioning causes the shape of the layer's mesh to change—and thus the layer's nontransparent pixels to move—to “accommodate” this repositioning. AEM, 218-19; Section VI.A.2.e; Greenspun, ¶117. Further, AECS6 performs keyframe interpolation to animate this movement of the layer's nontransparent pixels, which is performed according to, and visually indicated by, a motion path extending from the starting point to the ending point. AEM, 187, 218-19; Section VI.A.2.e; Greenspun, ¶117. As such, for a given span of time in the composition, if a Deform pin's ending position/keyframe is repositioned a long distance away from its starting position/keyframe, the change to the shape of the layer's mesh—and thus the movement of the layer's nontransparent pixels—will be of a large magnitude to “accommodate” such a large repositioning, and the Deform pin's corresponding motion path will be of a longer length. *See* AEM, 218-19; Greenspun, ¶117. The opposite occurs if the Deform pin repositioned only a short distance. *See* AEM, 218-19; Greenspun, ¶117. Such illustrates a proportional relationship between the

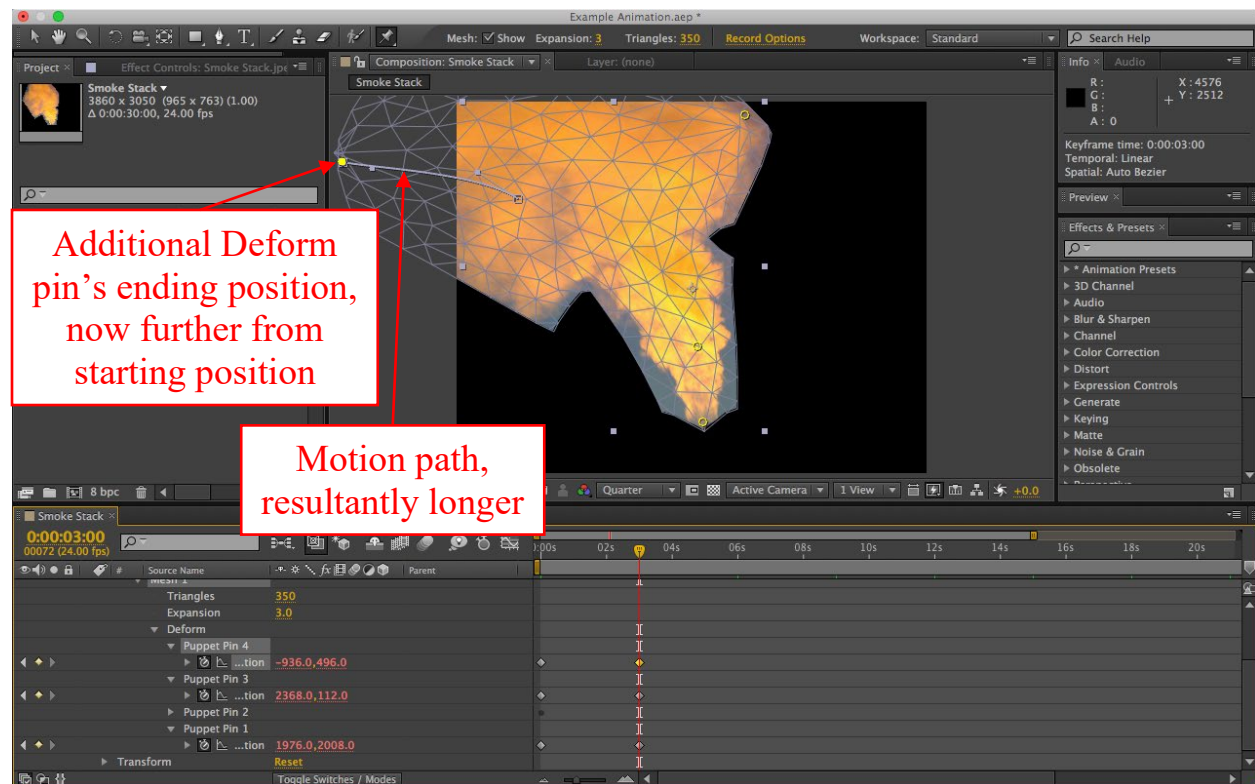
IPR2023-00153

U.S. Patent No. 10,346,017

magnitude of pixel movement and the length of the motion path for each corresponding Deform pin, thereby disclosing claim 6. Greenspun, ¶117.

By way of further illustration, the exemplary composition from Section VI.A.2.h's table is depicted in the screenshots below, but now the first Deform pin is repositioned a lesser amount (shown in the first screenshot) while one additional Deform pin is repositioned a larger amount (shown in the second screenshot) for the same span of time:





Greenspun, ¶118. As shown, the first Deform pin's motion path is now shorter, and the surrounding pixels have a proportionally lower magnitude of movement, whereas the opposite is true of the additional Deform pin. Greenspun, ¶118.

8. Claim 7

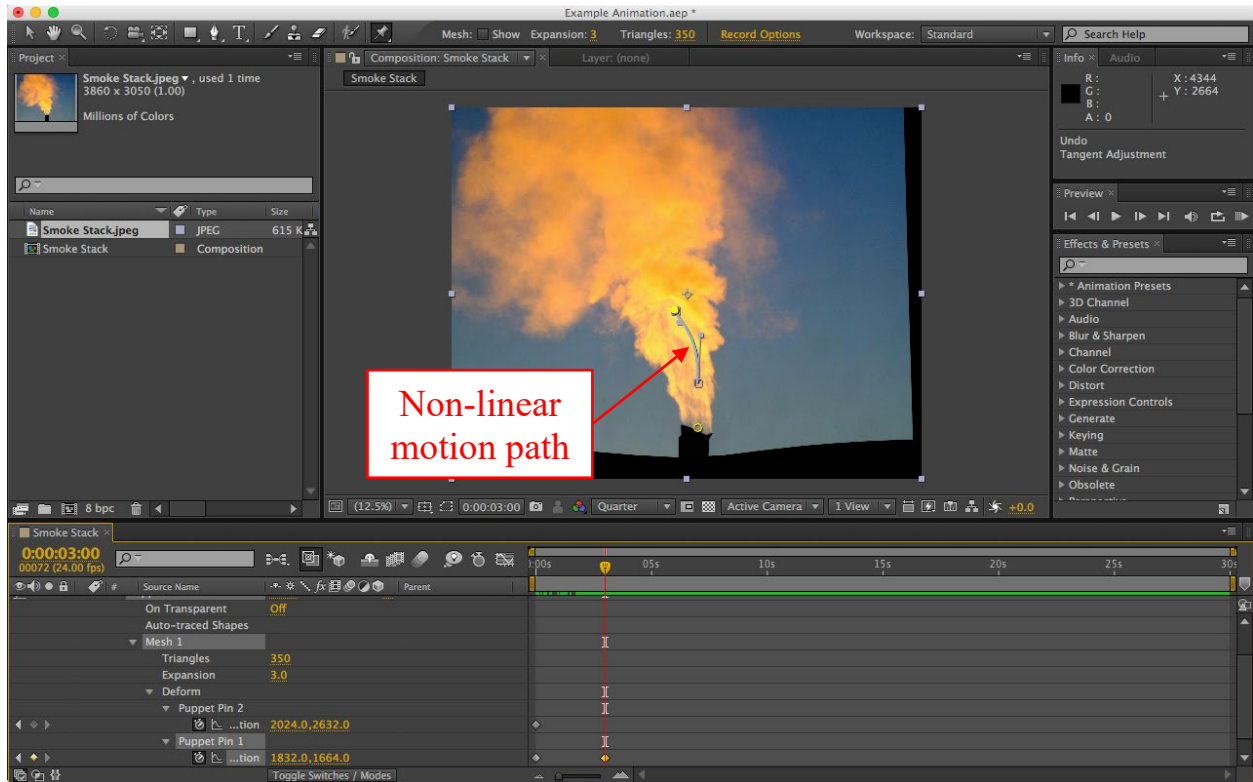
a. [7a]

As similarly discussed in Section VI.A.2.f, AEM teaches that AECS6's user applies a mask or matte to select any or all pixels of the layer to be nontransparent and thus included in the layer's mesh to be moved and animated, including pixels that intersect the first Deform pin's motion path—i.e., the claimed “first set of pixels comprise pixels that intersect the first rendered link.” AEM, 218-19, 315, 194-95; Greenspun, ¶119.

Separately, for the same reasons discussed in Section VI.A.2.f, a POSITA would have found obvious selecting and making nontransparent, using a mask or matte, not only pixels parallel to the Deform pin's motion path, within a user-defined threshold distance from the motion path, and extending from the Deform pin's starting position/keyframe to the ending position/keyframe, but also pixels intersecting the motion path. Greenspun, ¶120. Indeed, each mask/matte and mesh depicted in the screenshots in Section VI.A.2.f covers pixels intersecting the first Deform pin's motion path. Greenspun, ¶120.

b. [7b]

AEM teaches not only creating a motion path extending between a Deform pin's starting position/keyframe and ending position/keyframe as discussed in Section VI.A.2.e, but also making such a motion path non-linear using, e.g., "Bezier interpolation," thus disclosing [7b]. AEM, 193-95; Greenspun, ¶121. In fact, the motion path depicted in Section VI.A.2.e's screenshot, further annotated below, is non-linear:



Greenspun, ¶121.

9. Independent Claim 8

a. [8pre]

To the extent the preamble of independent claim 8 is limiting, AEM meets it by teaching a method for animating the movement of pixels within a still image using AECS6, as explained further below. Sections VI.A.9.b-VI.A.9.g; Greenspun, ¶122.

b. [8a]-[8b]

See Section VI.A.2.c.

c. [8c]-[8d]

See Section VI.A.2.d. Additionally, the Deform pin's ending position/keyframe is, as claimed, "associated with" its starting position/keyframe via

a motion path for the Deform pin, which extends between these two positions. AEM, 187, 218-19; Section VI.A.2.e; Greenspun, ¶124.

d. [8e]

See Section VI.A.2.e.

e. [8f]

See Section VI.A.8.a.

f. [8g]

See Sections VI.A.2.g, VI.A.8.a.

g. [8h]-[8n]

See Sections VI.A.2.h, VI.A.8.a.

10. Claim 9

AEM teaches, and a POSITA would have found obvious, applying a mask or matte in AECS6 to select any or all pixels of a layer to be nontransparent and thereby included in the layer’s mesh to be moved and animated, including pixels intersecting the Deform pin’s motion path and away from the starting keyframe—the claimed “first set of pixels.” AEM, 218-19, 315, 194-95; Section VI.A.9.e; Greenspun, ¶129. The user creates the starting keyframe—“first starting point”—by specifying a time point and placing a Deform pin on a “nontransparent pixel” of the layer—“a pixel selected from the first set of pixels.” AEM, 218-19; Section VI.A.9.b; Greenspun, ¶129. The user then repositions the Deform pin to create the ending keyframe—“ending point.” AEM, 218-19, 177; Section VI.A.9.c; Greenspun, ¶129. AEM

explains that AECS6 then performs animation by interpolating the position of the Deform pin from the starting to ending keyframe along the user-specified motion path and moving the layer's mesh and thus the layer's nontransparent pixels correspondingly. AEM, 218-19, 194-95; Section VI.A.9.f; Greenspun, ¶129. Further, the user applies a “loopOut()” expression to loop and repeat such interpolation and animation—“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.” AEM, 219, 562; Section VI.A.9.f; Greenspun, ¶129.

11. Claim 10

a. [10a]-[10b]

See Section VI.A.3.a.

12. Claim 11

See Section VI.A.4.

13. Claim 12

a. [12a]-[12b]

See Section VI.A.5.a.

14. Claim 13

a. [13a]

See Section VI.A.2.f.

b. [13b]

See Section VI.A.8.b.

15. Independent Claim 14

a. [14pre]

To the extent the preamble of independent claim 14 is limiting, AEM meets it by teaching a method for animating the movement of pixels within a still image using AECS6, as explained further below. Sections VI.A.15.b-VI.A.15.h; Greenspun, ¶135.

b. [14a]

See Section VI.A.2.b.

c. [14b]-[14c]

See Section VI.A.2.c.

d. [14d]-[14e]

See Section VI.A.2.d.

e. [14f]

See Sections VI.A.2.e, VI.A.8.b.

f. [14g]

See Sections VI.A.2.f, VI.A.8.a-VI.A.8.b.

g. [14h]

See Sections VI.A.2.g, VI.A.8.a-VI.A.8.b.

h. [14i]-[14o]

See Sections VI.A.2.h, VI.A.8.a-VI.A.8.b.

16. Claim 15

a. [15a]-[15b]

See Section VI.A.3.a.

17. Claim 16

See Section VI.A.4.

18. Claim 17

a. [17a]-[17c]

See Section VI.A.5.a.

19. Claim 18

See Sections VI.A.8.a-VI.A.8.b, VI.A.2.f.

B. Ground 2: IMU and Okabe Render Obvious Claims 1-3, 5-11, 13-16, and 18

1. IMU

IMU contains Wayback Machine captures from 2012 of the ImageMagick.com website, specifically the website’s section titled “Examples of ImageMagick Usage (Version 6),” which provides guidance and examples on how to use IMV6’s various effects and capabilities. IMU-Home, 1-2; Greenspun, ¶147.

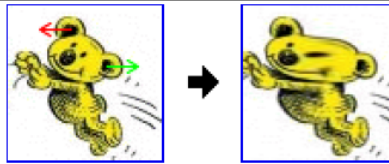
IMU teaches that IMV6 is operated in the command-line. IMU-Home, 2; Greenspun, ¶148. As such, IMV6 “*is not a GUI image editor.*” IMU-Home, 2; Greenspun, ¶148.

Further, IMU teaches how to manually write text commands in IMV6 to apply effects to an image, such as IMV6's "Shepard's Distortion" pixel shifting effect, as well as how to create infinitely looping animations from such effects. IMU also teaches how to manually write commands to modify the transparency of an image's pixels—and thus the visibility of effects that are applied to the image—using a "matte."

a. Applying Effects and Creating an Animation

IMU teaches that IMV6 is an "image-to-image converter" that utilizes a "library of Image Processing Algorithms" to apply effects and convert one image into another. IMU-Home, 2; Greenspun, ¶150. One such effect is "Shepard's Distortion," a distortion effect applied by placing "control points" on pixels located at user-specified coordinates of an image and moving such control points to new coordinates "to distort the image in terms of 'local' effects." IMU-Distorting, 59, 19-20; Greenspun, ¶150. IMU provides an example of placing and moving Shepard's Distortion control points on an image of a koala to "torture the 'koala' by pulling on his ears" from coordinates (30,11) and (48,29) to coordinates (20,11) and (59,29), respectively. IMU-Distorting, 59; Greenspun, ¶150. The example's command and koala image before and after the "torture," is reproduced below, with original annotations from IMU indicating the corresponding placement and movement of the two control points:

```
convert koala.gif -virtual-pixel Black \  
-distort Shepards '30,11 20,11 48,29 58,29' \  
koala_ear_pull.png
```



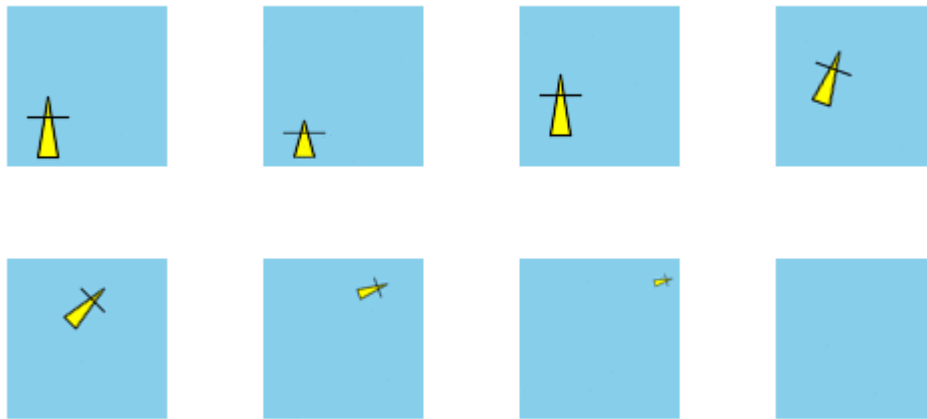
IMU-Distorting, 59; Greenspun, ¶150.

Although IMV6 is an “image-to-image converter,” IMU further teaches how to incrementally apply IMV6’s effects to an image to produce a sequence of frames that, when played sequentially, resulting in an animation of the original image. *See* IMU-Distorting, 17-18; Greenspun, ¶151. IMU provides an example of applying IMV6’s “SRT Distortion” effect—another “[d]istortion” effect that, like Shepard’s Distortion, uses “control points”—to an image of a “stylized space ship,” by placing and moving an SRT Distortion control point in seven increments to create a sequence of eight frames that, when played sequentially, animate the launch of the space ship along a curved path through the sky. IMU-Distorting, 17-18; Greenspun, ¶151. The example’s command and the eight resulting frames in their sequential order are reproduced below:

```

convert -size 80x80 xc:skyblue -fill yellow -stroke black \
  -draw 'path "M 15,75 20,45 25,75 Z M 10,55 30,55" ' \
  spaceship.gif
convert spaceship.gif \
  \( -clone 0 -distort SRT '20,75 1.0,0.6 0' \) \
  \( -clone 0 -distort SRT '20,60 1 0 20,49' \) \
  \( -clone 0 -distort SRT '20,60 0.9 20 27,35' \) \
  \( -clone 0 -distort SRT '20,60 0.8 45 40,23' \) \
  \( -clone 0 -distort SRT '20,60 0.5 70 55,15' \) \
  \( -clone 0 -distort SRT '20,60 0.3 75 72,11' \) \
  \( -clone 0 -distort SRT '20,60 0.1 80 100,8' \) \
  -set delay 50 -loop 0 spaceship_launch.gif

```



IMU-Distorting, 17-18¹⁰; Greenspun, ¶151.

After creating a sequence of frames, IMU teaches how to save these frames in an animated GIF file that will be rendered as an animation by a variety of standard computer programs, including Web browsers, that show each frame in sequence.

¹⁰ The full animation can be viewed on the IMU-Distorting webpage as archived at <https://web.archive.org/web/20120329131929/http://www.imagemagick.org/Usage/distorts/>.

IMU-Animating, 1-2; IMU-Distorting, 17-18; Greenspun, ¶152. In creating such an animated GIF, IMU teaches applying a “-loop” operator to specify the “[n]umber of times the GIF animation is to cycle though the image sequence before stopping.” IMU-Animating, 2; Greenspun, ¶152. When the “-loop” operator’s value is set to its default value of zero, such as in the above exemplary command, the animated GIF will render the animation in an “infinite loop.” IMU-Animating, 2; IMU-Distorting, 17-18; Greenspun, ¶152.

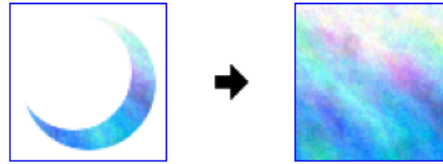
b. Modifying Transparency of Certain Pixels Using a Matte

IMU also teaches modifying transparency of an image’s pixels—and thus the visibility of effects that are applied to the pixels—by modifying the image’s “transparency (alpha) channel,” also known as a “‘matte’ channel” or simply a “matte.” IMU-Masking, 2-3; Greenspun, ¶153. This matte “is just a plain grey scale image of values which range from white, for full-transparent (or clear), to black for fully-opaque.” IMU-Masking, 2; Greenspun, ¶153.

IMU provides an example matte shaped like a crescent moon:



IMU-Masking, 2; Greenspun, ¶154. IMU further illustrates how this matte is applied or not applied to correspondingly change the transparency of an image’s pixels:



IMU-Masking, 3-4; Greenspun, ¶154.

2. Okabe

Okabe describes a “method for synthesizing fluid animation from a single image.” Okabe, 1; Greenspun, ¶155. While acknowledging that “[m]any methods have been proposed for creating an animation from a single image,” Okabe asserts that its method of creating such an animation provides, among other advantages, “markedly reduce[d]... user burden.” Okabe, 2; Greenspun, ¶155.

Okabe teaches that, to animate an image, the user first “inputs a target painting or photograph of a fluid scene along with its alpha matte that extracts the fluid region of interest in the scene.” Okabe, 1; Greenspun, ¶156. Thereafter, the user “sketch[es] the flow direction and paint[s] a speed map” by drawing “a sparse set of user-drawn strokes” on the image, “which is a simple task and takes less than 1 min[ute].” Okabe, 3, 7; Greenspun, ¶156. Okabe’s Figure 1 provides an example and shows that the “strokes”—“shown as orange arrows” in Okabe’s interface—may be non-linear:



Okabe, 2; Greenspun, ¶156. From these inputs, Okabe's algorithm then generates a flow animation in the target image corresponding to the flow direction and speed specified by the user's strokes, and permits "infinite repetition" of such an animation. Okabe, 1, 8; Greenspun, ¶156. According to Okabe, such simple and graphical commands allow images to be animated "with less effort than with previous methods." Okabe, 1; Greenspun, ¶156.

3. The IMU-Okabe Combination

a. **Motivation to Combine IMU with Okabe**

As explained below, a POSITA would have been motivated to combine IMV6's animation capabilities described in IMU with the simple and graphical user commands for creating an animation taught by Okabe (hereinafter, the "IMU-Okabe Combination").

According to IMU, IMV6 is an "image-to-image converter" operated in the command-line and therefore "*not a GUI image editor.*" IMU-Home, 2; Greenspun, ¶158. As such, to animate an image using IMV6, a user must create each frame of the animation by manually writing text commands that apply IMV6's effects to each frame. *E.g.*, IMU-Distorting, 17-18; Greenspun, ¶¶158-59.

Given this, a POSITA would have recognized that, to animate an image in IMV6 using, e.g., Shepard's Distortion, a user would write commands that place and incrementally move Shepard's Distortion control points within the image to new

coordinates for each successive frame, such that each control point moves across the animation at the user's desired—and potentially non-linear—direction and speed. *See* IMU-Distorting, 17-18 (creating an animation in the same way, but using IMV6's "SRT distortion" effect, to animate the launch of a space ship along a non-linear path), 59, 19-20; Greenspun, ¶160.

Further, to make distortion and animation visible only in a particular region of the image, the user would write further commands that apply a corresponding matte to the image such that only the particular region is visible during the distortion and animation. IMU-Masking, 2; Greenspun, ¶160. Lastly, to save the animation as a looping animated GIF, the user would write further commands to save the animation using IMV6's "-loop" operator. IMU-Animating, 1-2; Greenspun, ¶161; *see also, e.g.*, IMU-Distorting, 17-18.

But given this laborious, non-graphical, text-based process for creating an animation in IMV6, a POSITA would have been motivated to modify IMV6 by enabling a user to animate an image using simpler and more intuitive graphical user commands. Greenspun, ¶162. In other words, a POSITA would have been motivated to modify IMV6 such that it *is* a "*GUI image editor*." *See* IMU-Home, 2; Greenspun, ¶162.

The motivation is underscored in IMU itself, which acknowledges the disadvantages of IMV6's lack of a GUI when applying effects to a single image,

noting “there are a lot of readily available image manipulation programs, such as Adobe Photoshop, Corel’s Paint Shop Pro, [etc]. So why should you bother to perform image processing by IM[V6]’s command line programs and scripts?” IMU-Windows, 1; Greenspun, ¶163. Indeed, AECS6 was another such “readily available image manipulation program[]” that combined the ability to create a looping animation of effects applied to images (e.g., Puppet effect) with simple and graphical user commands for defining such animation (e.g., placing Deform pins, creating motion paths between keyframes for the Deform pins, and specifying which pixels move using a mask or matte, all within the “Composition” or “Layer panel[s]”), confirming a POSITA would have been—and in fact was—motivated to make such a combination of features. *See* AEM, 218-19, 187-89, 193-95, 317, 562; Section VI.A.1; Greenspun, ¶163.

Okabe teaches simple and graphical user commands that a POSITA would have found well-suited for implementing a GUI to create animations in IMV6. Greenspun, ¶164. Specifically, Okabe teaches that a user animates an image by first using an “alpha matte” to specify a “region of interest” the user wants to animate. Okabe, 1; Greenspun, ¶164. The user also draws on the image a “sparse set of user-drawn strokes,” which may be non-linear, to specify the user’s desired direction and speed of animation, “which is a simple task and takes less than 1 min[ute].” Okabe, 7, 2, 3; Greenspun, ¶164. From just these inputs, Okabe describes automatically

producing an infinitely repeating animation that follows the user-specified direction and speed. Okabe, 1-2, 8; Greenspun, ¶164. As such, Okabe’s simple and graphical user commands enables a user to create looping animations “with less effort than with previous methods” and “markedly reduces the user burden.” Okabe, 1-2; Greenspun, ¶164.

Given such advantages, a POSITA would have recognized that Okabe’s simple and graphical user commands would have been desirable for animating an image in IMV6, i.e., obviating IMV6’s laborious and non-graphical animation procedure. Greenspun, ¶165. As such, a POSITA would have been motivated to modify IMV6 to include Okabe’s simple and graphical user commands to make the animation process easier, graphical, and less time consuming. Greenspun, ¶165.

b. Resulting Combination

So modified, rather than require a user to write complex commands that place and move Shepard’s Distortion control points on an image frame-by-frame such that each control point moves across the animation at the user’s desired—and potentially non-linear—direction and speed (*see* IMU-Distorting, 17-18, 59, 19-20), the IMU-Okabe Combination enables a user to achieve the same result by drawing non-linear

strokes viewable by the user¹¹ on the image to specify the direction and speed of corresponding Shepard's Distortion control points across the animation (*see* Okabe, 7, 2-3). Greenspun, ¶166.

Also, similar to how IMV6 allows a user to apply a matte to make distortion and animation effects visible in only a portion of an image (IMU-Masking, 2), the IMU-Okabe Combination's user applies a matte to make the Shepard's Distortion effect from the user-drawn strokes visible in only a portion of the image (*see* Okabe, 1). Greenspun, ¶167. Further, as IMV6 allows an animation to be saved as an infinitely looping animated GIF using the “-loop” operator (IMU-Animating, 1-2), the IMU-Okabe Combination likewise permits infinite repetition of the resulting animation (*see* Okabe, 1, 8). Greenspun, ¶167.

¹¹ Should Plotagraph contend Okabe's strokes are not disclosed as user-viewable, a POSITA would have nevertheless been motivated to make Okabe's strokes user-viewable to aid users in visualizing the direction and speed of their animations, thereby making the animation process simpler, graphical, and more intuitive. Greenspun, ¶167, n.14; *see also, e.g.*, AEM, 187 (“Motion paths are simply an alternative visual, spatial way of viewing and working with spatial properties and their keyframes, in addition to the ways that you work with properties in the Timeline panel.”).

4. Independent Claim 1**a. [1pre]**

To the extent the preamble of claim 1 is limiting, IMU teaches that IMV6 is operated in the command-line and installed and run on a computer. IMU-Home, 2; IMU-Windows, 3-4; Greenspun, ¶168. Okabe teaches that, when animating an image using Okabe's simple and graphical user commands, "the designer specifies a single target image along with several characteristics regarding motion and uses a computer to synthesize animated sequences derived from the input." Okabe, 1; Greenspun, ¶168. Given both IMV6 and Okabe's commands require the use of a computer, the IMU-Okabe Combination would thus also be installed and run on a computer, which would include a processor—the claimed "one or more processors"—and storage storing instructions for running the IMU-Okabe Combination—"one or more computer-readable media." Section VI.B.3.b; Greenspun, ¶168.

Further, the IMU-Okabe Combination enables a user to automatically animate an image—"automating the shifting of pixels within a digital image." Sections VI.B.4.b-VI.B.4.h; Greenspun, ¶169.

b. [1a]

According to IMU, IMV6 is an "image-to-image converter" that enables a user to apply effects to animate an image—the claimed "digital image file, wherein the digital image file comprises information that corresponds to individual pixels

within the digital image.” IMU-Home, 2; IMU-Distorting, 17-18; Greenspun, ¶170.

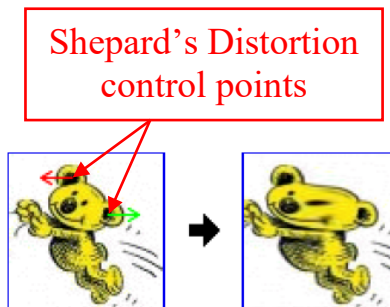
The image accessed and animated by IMV6 can be “an IM[V6] built-in image (such as ‘logo:’, or ‘rose:’)” or an “external image” from an external source “placed in the ‘images’ and ‘img_photos’ sub-directories”—“access, from memory, a digital image file.” IMU-Home, 3-4; Greenspun, ¶170.

Further, Okabe teaches that, to utilize its simple and graphical user commands, “[t]he user inputs a target image and an alpha matte that extracts the fluid region of interest, while providing a few optional suggestions about fluid motion (i.e., flow direction and speed),” which likewise meets the claimed “access, from memory, a digital image file.” Okabe, 1-2; Greenspun, ¶171. The IMU-Okabe Combination would thus enable a user to input a digital image from a memory, meeting [1a]. Section VI.B.3.b; Greenspun, ¶171.

c. [1b]-[1c]

IMU teaches that IMV6 is an “image-to-image converter” that enables a user to apply effects, such as distortion effects, to convert one image—the claimed “digital image”—into a sequence of frames that animate the original image when played sequentially. IMU-Home, 2; IMU-Distorting, 17-18; Greenspun, ¶172. Distortion effects include “Shepard’s Distortion.” IMU-Distorting, 59; Greenspun, ¶172. To animate with distortion effects, the user places a “control point” on a pixel located at a user-specified coordinate of the image and then moves the control point

to a new coordinate. IMU-Distorting, 19-20; Greenspun, ¶172; *see also, e.g.*, IMU-Distorting, 17-18. This is done incrementally for each successive frame such that the control point moves across the animation at the user's desired direction and speed. *See* IMU-Distorting, 19-20; Greenspun, ¶172; *see also, e.g.*, IMU-Distorting, 17-18. IMU provides an example of one such increment, where two Shepard's Distortion control points are placed on an image of a koala at coordinates (30, 11) and (48, 29), before being moved to coordinates (20,11) and (59,29), respectively. *See* IMU-Distorting, 59; Greenspun, ¶172. IMU also provides a figure showing the koala image before and after the Shepard's Distortion is applied:



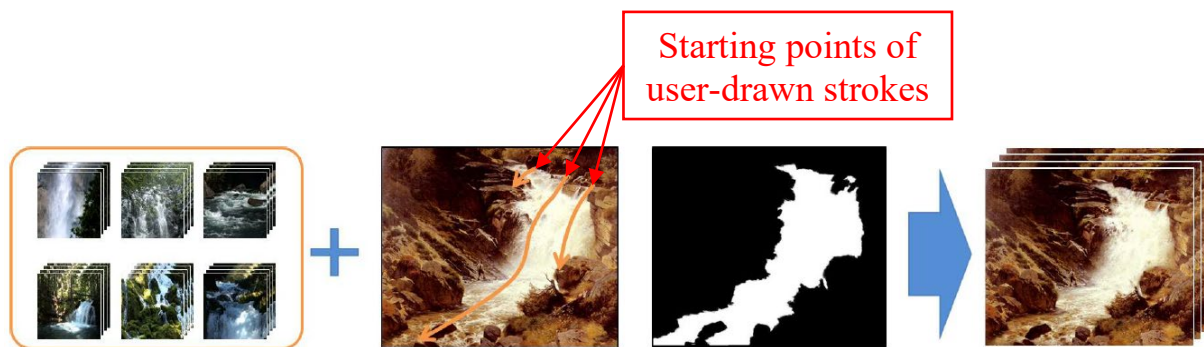
IMU-Distorting, 59 (additional annotations added); Greenspun, ¶172.

Further, in the IMU-Okabe Combination, IMU's placement and movement of a Shepard's Distortion control point corresponds to Okabe's user-drawn, user-viewable "stroke[]" on an image. *See* Okabe, 7, 3; Section VI.B.3.b; Greenspun, ¶173. That is, Okabe's user-drawn stroke includes a user-viewable indication of a starting point, direction, and speed across the animation, much like how a user in IMV6 indicates a control point's starting point, direction, and speed across an

animation according to IMU. *See* Okabe, 7, 3; IMU-Distorting, 17-18, 59, 19-20; Section VI.B.3.b; Greenspun, ¶173.

Receiving the starting point of a user-drawn, user-viewable stroke corresponds to the placement of a Shepard's Distortion control point by the user on a specific coordinate of the image and thus meets [1b]-[1c]. *See* IMU-Distorting, 59, 19-20; Okabe, 7, 3; Greenspun, ¶174.

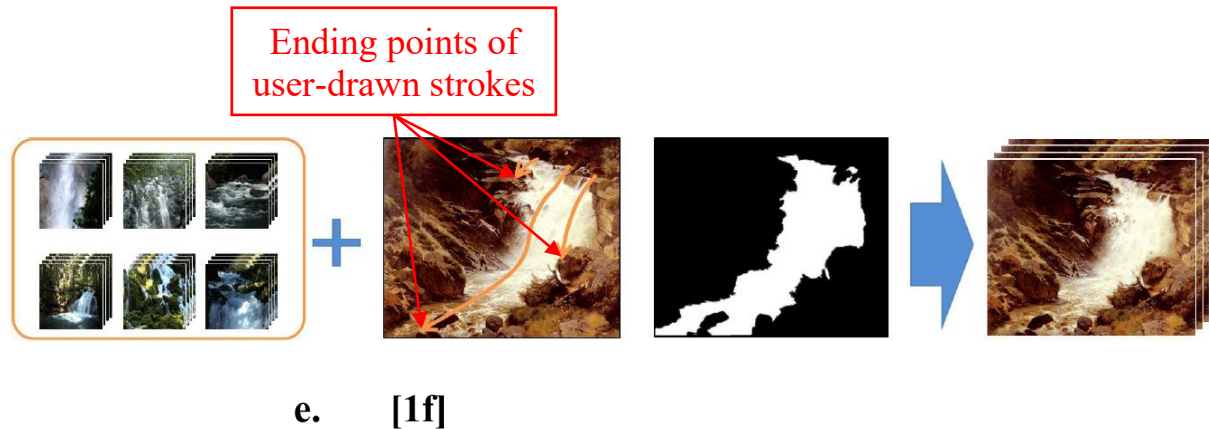
Okabe's Figure 1 illustrates the relevant disclosure by depicting several user-drawn strokes (shown in orange) on an image of a waterfall:



Okabe, 2 (Figure 1, annotations added); Greenspun, ¶175.

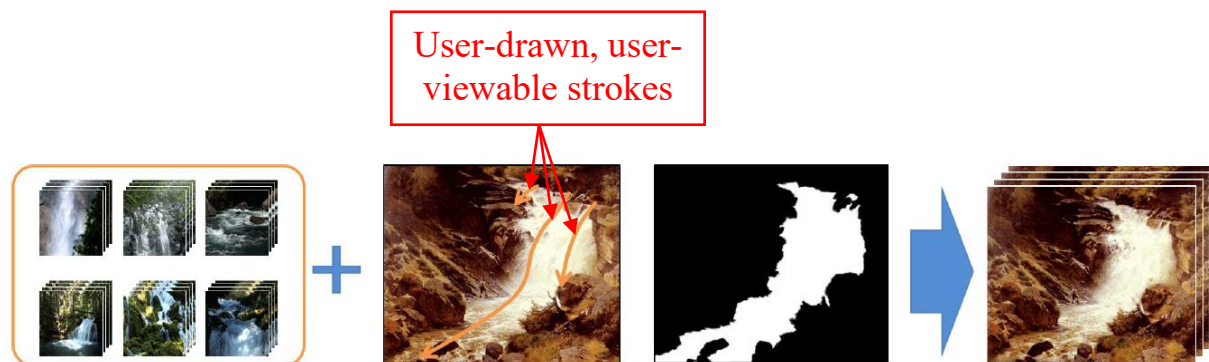
d. [1d]-[1e]

In the IMU-Okabe Combination, a user defines a Shepard's Distortion control point's direction and speed across the animation of an image by drawing a user-viewable "stroke[]" on the image. *See* Okabe, 7, 2-3; IMU-Distorting, 59; Section VI.B.3.b; Greenspun, ¶176. Receiving a user-drawn, user-viewable stroke, specifically the stroke's ending point, meets [1d]-[1e]. Greenspun, ¶176; *see also* Okabe, 2 (Figure 1, reproduced below with annotations added).



IMU teaches animating in IMV6 using distortions (e.g., Shepard's Distortion) by placing and incrementally moving control points frame-by-frame such that each control point moves across the animation at the user's desired direction and speed. *See* IMU-Distorting, 17-18, 59, 19-20; Greenspun, ¶177.

Further, in the IMU-Okabe Combination, a user specifies such a direction and speed of a Shepard's Distortion control point by drawing a corresponding user-viewable "stroke[]" from Okabe on the image, thus meeting [1f].” *See* Okabe, 7, 2-3; Section VI.B.3.b; Greenspun, ¶178. Indeed, Okabe's Figure 1 illustrates three user-viewable strokes received from the user:



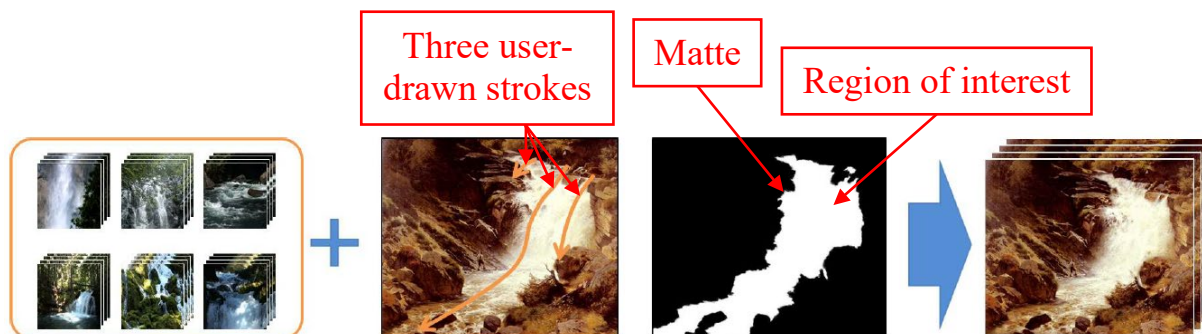
Okabe, 2 (Figure 1, annotations added); Greenspun, ¶178; *accord* Okabe, 6 (discussing Figure 8-a).

f. [1g]

IMU teaches modifying the transparency of an image's pixels in IMV6—and thus the visibility of effects that are applied to the image—by applying a “matte,” which “is just a plain grey scale image of values which range from white, for full-transparent (or clear), to black for fully-opaque.” IMU-Masking, 2; Greenspun, ¶179.

Further, the IMU-Okabe Combination's user similarly applies a matte specifying a region of interest in the image the user desires to animate (*see* Okabe, 1), in addition to a “stroke[]” that specifies the direction and speed of a Shepard's Distortion control point across the animation (*see id.*, 7, 3). Section VI.B.3.b; Greenspun, ¶180. Thus, the IMU-Okabe Combination would have permitted selecting any or all pixels of the image to be included in this region of interest, including pixels parallel to the stroke, within a user-defined threshold distance from the stroke, and extending from the starting point to the ending point of the stroke, by using a matte, thus meeting [1g]. *See* Okabe, 1; Greenspun, ¶180. Okabe's Figure 1 confirms this by depicting a matte specifying a region of interest (i.e., a to-be-animated region) in white that includes pixels parallel to each of three user-drawn strokes, within a user-defined threshold distance from the strokes (i.e., to the

waterfall's edges), and extending from the starting point to the ending point of each stroke:



Okabe, 2 (Figure 1, annotations added); Greenspun, ¶180; *accord* EX1022, 10-11.

g. [1h]

With respect to “shifting the first set of pixels along the non-linear pathway,” the ’017 Patent’s specification states the claimed “shifting” can be performed by using, for example, “a warping function, such as Shepard’s distortion.” ’017 Patent, 8:32-34; Greenspun, ¶181. IMU explicitly teaches that IMV6 performs “Shepard’s Distortion” and therefore the claimed pixel shifting. IMU-Distorting, 59; Greenspun, ¶181. That is, IMV6’s user shifts pixels as claimed by using Shepard’s Distortion to place and incrementally move a “control point” on an image frame-by-frame in the user’s specified direction and speed. *See* IMU-Distorting, 17-18, 59, 19-20; Greenspun, ¶181.

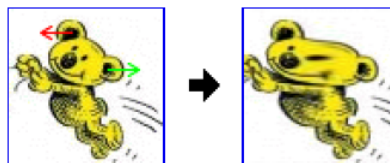
Further, the IMU-Okabe Combination’s user specifies a Shepard’s Distortion control point’s direction and speed by drawing a corresponding “stroke[]” from Okabe on the image. *See* Okabe, 7, 3, 2; Sections VI.B.3.b, VI.B.4.e; Greenspun,

¶182. The IMU-Okabe Combination’s user also applies a matte to select pixels parallel to the stroke, within a user-defined threshold distance from the stroke, and extending from the starting point to the ending point of the stroke, by using a matte—“the first set of pixels”—to be in a region of interest and thus animated. *See* Okabe, 1, 2; Section VI.B.4.f; Greenspun, ¶182. From these inputs, the IMU-Okabe Combination automatically generates animation frames by shifting the selected pixels in the stroke’s direction using IMV6’s Shepard’s Distortion—“automatically shift the first set of pixels in the first direction.” *See* Okabe, 1-2; Section VI.B.3.b; Greenspun, ¶182.

Additionally, IMU teaches that IMV6 allows a user to save their animation as an infinitely looping animated GIF using IMV6’s “-loop” operator. IMU-Animating, 1-2; Greenspun, ¶183. Similarly, Okabe describes automatically producing an infinitely repeating animation that follows a direction and speed based on a user-drawn stroke. Okabe, 1-2, 8; Greenspun, ¶183. The IMU-Okabe Combination thus likewise permits infinite repetition of an animation by, e.g., applying the “-loop” operator to an animation created by Shepard’s Distortion and the user’s stroke—“rendering and re-rendering in a loop the first set of pixels being shifted.” *See* IMU-Animating, 1-2; Okabe, 1, 8; Section VI.B.3.b; Greenspun, ¶183.

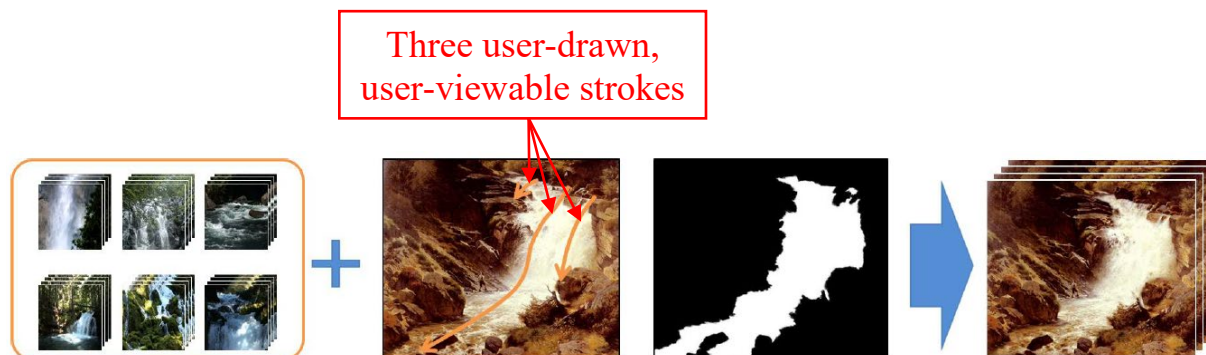
h. [1i]-[1o]

IMU teaches applying multiple Shepard's Distortion control points at different coordinates of an image in IMV6 and incrementally moving the control points to new coordinates to animate the image. *See* IMU-Distorting, 17-18, 59, 19-20; Greenspun, ¶184. IMU provides an example that places and moves two Shepard's Distortion control points on an image of a koala:



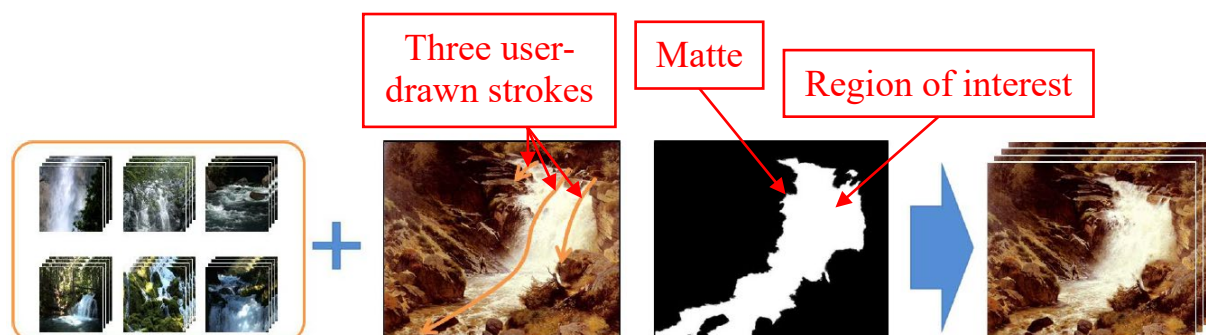
IMU-Distorting, 59; Greenspun, ¶184.

Further, in the IMU-Okabe Combination, this placement and movement of multiple Shepard's Distortion control points is performed by a user drawing multiple “strokes” from Okabe on the image, each of which is user-viewable and specifies a direction and speed of a Shepard's Distortion control point. *See* Okabe, 7, 2-3; Section VI.B.3.b; Greenspun, ¶185. Indeed, Okabe's Figure 1 depicts three user-drawn, user-viewable strokes:



Okabe, 2 (Figure 1, annotations added); Greenspun, ¶185. Receiving multiple user-drawn, user-viewable strokes—including each stroke’s starting and ending points, direction, and speed—meets [1i]-[1m]. Greenspun, ¶185; *see also* Sections VI.B.4.c-VI.B.4.e.

The IMU-Okabe Combination’s user also applies a matte specifying a region of interest in the image the user desires to animate, including pixels parallel to each stroke, within a user-defined threshold distance from the strokes, and extending from the starting point to the ending point of each stroke, thereby meeting [1n]. *See* Okabe, 1; IMU-Masking, 2; Section VI.B.3.b; Greenspun, ¶186; *accord* EX1022, 10-11; *see also* Section VI.B.4.f. Okabe’s Figure 1 confirms this by depicting a matte specifying a region of interest that includes pixels parallel to each of three strokes, within a user-defined threshold distance from the strokes (i.e., to the waterfall’s edges), and extending from the starting point to the ending point of each stroke:



Okabe, 2 (Figure 1, annotations added); Greenspun, ¶186.

After receiving multiple strokes and the matte from the user, the IMU-Okabe Combination automatically generates animation frames using Shepard's Distortion and permits infinite repetition of the animation by, e.g., applying IMV6's "-loop" operator, thereby meeting [1o]. *See* Okabe, 1-2, 8; IMU-Animating, 1-2; Section VI.B.3.b; Greenspun, ¶187; *see also* Section VI.B.4.g.

5. Claim 2

a. [2a]-[2b]

Similar to how IMV6 enables a user to apply a matte to modify the transparency of an image's pixels and therefore the visibility of effects applied to the pixels (IMU-Masking, 2), the IMU-Okabe Combination's user applies a matte specifying a region of interest in the image the user desires to animate (*see* Okabe, 1). Section VI.B.3.b; Greenspun, ¶188. Inherently, the matte also specifies a region that includes pixels the user does not desire to animate. Greenspun, ¶188. Indeed, Okabe's Figure 1 depicts a matte that includes a white region (pixels to be animated) and black region (pixels not to be animated):



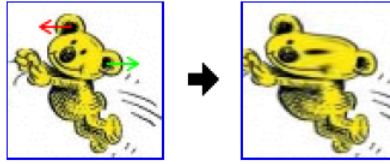
Okabe, 2 (Figure 1, annotations added); Greenspun, ¶188. Therefore, the matte includes a “mask [that] prevents pixels covered by the mask from being shifted,” such that “pixels under the mask are prevented from shifting,” as claimed. Greenspun, ¶188. Receiving and applying such a matte specified by the IMU-Okabe Combination’s user meets “identify a particular portion of the digital image to mask.” Greenspun, ¶188.

6. Claim 3

The IMU-Okabe Combination’s user applies a matte that specifies a region of an image that the user does not desire to animate. *See* Okabe, 1-2; Section VI.B.5.a; Greenspun, ¶189. As seen in Okabe’s Figure 1, such a matte includes a black region covering pixels the user does not desire to animate, each of which satisfies a “particular pixel within the digital image.” Okabe, 2; Greenspun, ¶189. Receiving the matte from the user meets the claimed “receiving.” Greenspun, ¶189.

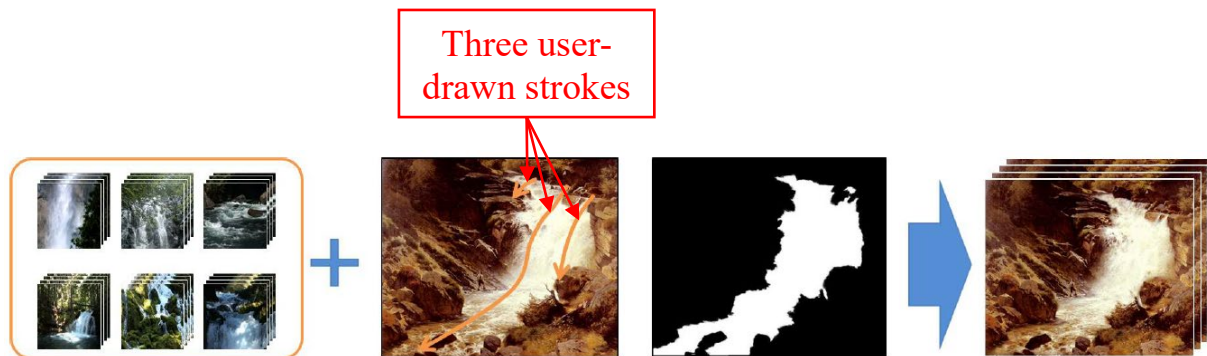
7. Claim 5

IMU teaches applying and incrementally moving control points, such as for Shepard’s Distortion, to different coordinates on an image to generate an animation. *See* IMU-Distorting, 17-18, 59, 19-20; Greenspun, ¶190. IMU provides an example that places two Shepard’s Distortion control points on an image of a koala and moves them in different directions:



IMU-Distorting, 59; Greenspun, ¶190.

Further, in the IMU-Okabe Combination, this placement and movement of multiple Shepard's Distortion control points is performed by a user drawing multiple "strokes" from Okabe on the image, which specify different directions and speeds of corresponding Shepard's Distortion control points across the animation, thus meeting claim 5. *See* Okabe, 7, 3; Section VI.B.3.b; Greenspun, ¶191. In fact, Okabe's Figure 1 depicts three user-drawn strokes with different directions:



Okabe, 2 (Figure 1, annotations added); Greenspun, ¶191.

8. Claim 6

IMU teaches that distortions using control points (e.g., Shepard's Distortion) require users to input "2 pairs of coordinates": X_i, Y_i and I_i, J_i . IMU-Distorting, 19, 59; Greenspun, ¶192. Specifically, "control point $X_i, [Y]_i$ in the source image (relative [to] its virtual canvas), is mapped to I_i, J_i on the distorted destination image."

IMU-Distorting, 19; Greenspun, ¶192. Thus, when generating animation frames using Shepard's Distortion, the magnitude of distortion from one frame to the next is directly related to the distance between the "source" and "destination" coordinates inputted for a corresponding control point. *See* IMU-Distorting, 19, 59; Greenspun, ¶192.

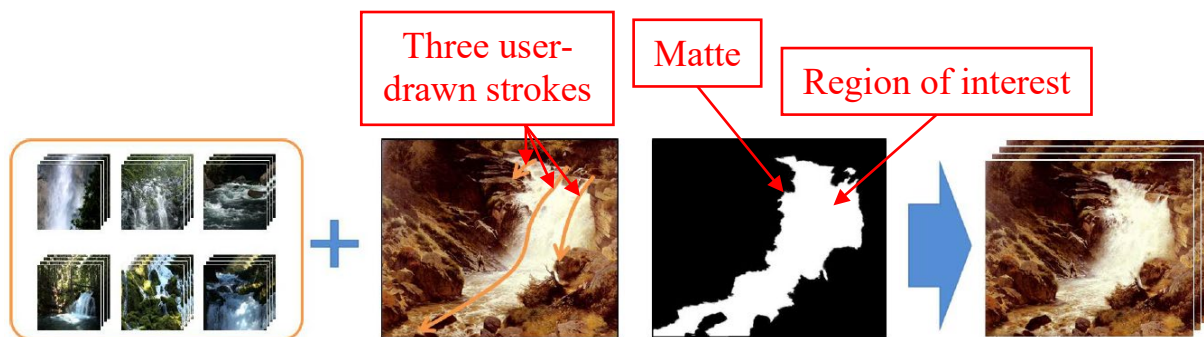
In the IMU-Okabe Combination, the placement and movement of multiple Shepard's Distortion control points is performed by drawing a "stroke[]" from Okabe on the image to specify the direction and speed of a corresponding Shepard's Distortion control point across the animation. *See* Okabe, 7, 3; Section VI.B.3.b; Greenspun, ¶193. Thus, longer strokes correspond to larger frame-by-frame distances of travel for corresponding Shepard's Distortion control points, resulting in larger shifts, and vice versa. *See* IMU-Distorting, 19, 59; Section VI.B.3.b; Greenspun, ¶193. Such establishes the claimed "proportional[]" relationships. Greenspun, ¶193.

9. Claim 7

a. [7a]

The IMU-Okabe Combination's user applies a matte (*see* IMU-Masking, 2) specifying a to-be-animated region of interest in the image the user desires to animate (*see* Okabe, 1), in addition to a "stroke[]" that specifies the direction and speed of a Shepard's Distortion control point across the animation (*see* Okabe, 7, 3).

Sections VI.B.4.f, VI.B.3.b; Greenspun, ¶194. Thus, using a matte, a user would have selected and included in this region of interest not only pixels parallel to the stroke, within a user-defined threshold distance from the stroke, and extending from the starting point to the ending point of the stroke, but also pixels intersecting the stroke, thereby meeting [7a]. *See* Okabe, 1; Greenspun, ¶194. Okabe's Figure 1 confirms this by depicting a matte specifying a region of interest in white that includes pixels intersecting three strokes:



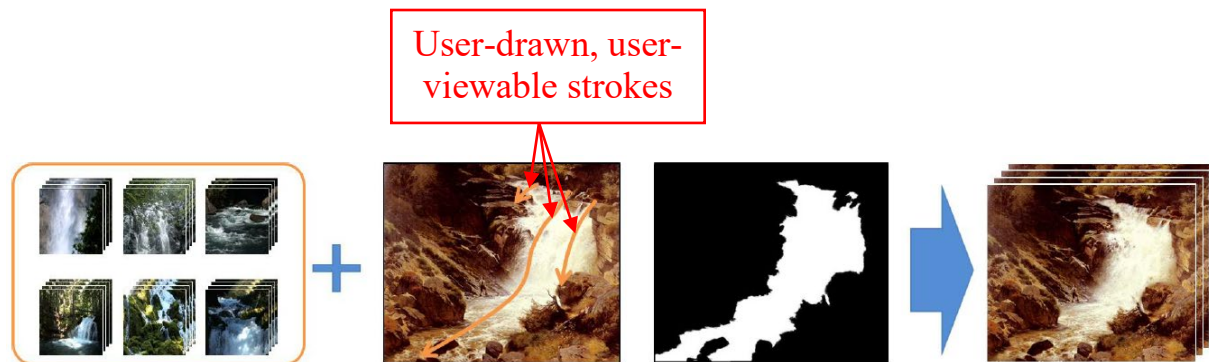
Okabe, 2 (Figure 1, annotations added); Greenspun, ¶194.

b. [7b]

IMU teaches that, when animating in IMV6 using distortion effects such as Shepard's Distortion, the direction and speed of animation desired by the user (*see* Section VI.B.4.e) may be non-linear as specified by coordinates input by the user. *See, e.g.*, IMU-Distorting, 17-18; Greenspun, ¶195; *see also* IMU-Distorting 59, 19-20; Section VI.B.3.b.

Further, in the IMU-Okabe Combination, a user specifies such a non-linear direction and speed of a Shepard's Distortion control point by drawing a

corresponding non-linear “stroke[]” from Okabe on the image extending between the stroke’s starting and ending point—i.e., the claimed “non-linear pathway extending between the first starting point and the first ending point.” *See* Okabe, 7, 2-3; Section VI.B.3.b; Greenspun, ¶196. Indeed, Okabe’s Figure 1 illustrates three non-linear strokes received from the user:



Okabe, 2 (Figure 1, annotations added); Greenspun, ¶196; *accord* Okabe, 6 (discussing Figure 8-a).

10. Claim 8

a. [8pre]

To the extent the preamble of claim 8 is limiting, the IMU-Okabe Combination meets it. Sections VI.B.10.b-VI.B.10.g; Greenspun, ¶197.

b. [8a]-[8b]

See Section VI.B.4.c.

c. [8c]-[8d]

See Section VI.B.4.d. Additionally, the stroke’s ending and starting positions are, as claimed, “associated with” one another by being positions for the same stroke.

Section VI.B.4.e; Greenspun, ¶199.

d. [8e]

See Section VI.B.4.e.

e. [8f]

See Section VI.B.9.a.

f. [8g]

See Sections VI.B.4.g, VI.B.9.a.

g. [8h]-[8n]

See Sections VI.B.4.h, VI.B.9.a.

11. Claim 9

The IMU-Okabe Combination’s user selects pixels in a region of interest of an image using a matte, including pixels extending along a length of a user-drawn “stroke[]” and away from the stroke’s starting point—the claimed “first set of pixels.” *See* IMU-Masking, 2; Okabe, 1; Section VI.B.4.f; Greenspun, ¶204.

Further, the user-drawn stroke specifies the direction and speed of a Shepard’s Distortion control point across the animation. *See* Okabe, 7, 3; IMU-Distorting, 59; Section VI.B.3.b; Greenspun, ¶205. Specifically, the stroke’s starting point represents the placement of the Shepard’s Distortion control point at a specified

coordinate—“a pixel selected from the first set of pixels.” *See* Okabe, 7, 3; IMU-Distorting, 59; Section VI.B.4.c; Greenspun, ¶205. The stroke’s ending point meets the “ending point” “reache[d]” at the end of the animation. *See* Okabe, 7, 3; Section VI.B.4.d; Greenspun, ¶205.

The IMU-Okabe Combination then automatically generates animation frames using Shepard’s Distortion and permits infinite repetition of the animation by, e.g., applying IMV6’s “-loop” operator—“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.” *See* IMU-Animating, 1-2; Okabe, 1-2, 8; Section VI.B.4.g; Greenspun, ¶206.

12. Claim 10

a. [10a]-[10b]

See Section VI.B.5.a.

13. Claim 11

See Section VI.B.6.

14. Claim 13

a. [13a]

See Section VI.B.4.f.

b. [13b]

See Section VI.B.9.b.

15. Independent Claim 14

a. [14pre]

To the extent the preamble of claim 14 is limiting, the IMU-Okabe Combination meets it. Sections VI.B.15.b-VI.B.15.h; Greenspun, ¶211.

b. [14a]

See Section VI.B.4.b.

c. [14b]-[14c]

See Section VI.B.4.c.

d. [14d]-[14e]

See Section VI.B.4.d.

e. [14f]

See Sections VI.B.4.e, VI.B.9.b.

f. [14g]

See Sections VI.B.4.f, VI.B.9.a-VI.B.9.b.

g. [14h]

See Section VI.B.4.g, VI.B.9.a-VI.B.9.b.

h. [14i]-[14o]

See Sections VI.B.4.h, VI.B.9.a-VI.B.9.b.

16. Claim 15

a. [15a]-[15b]

See Section VI.B.5.a.

17. Claim 16

See Section VI.B.6.

18. Claim 18

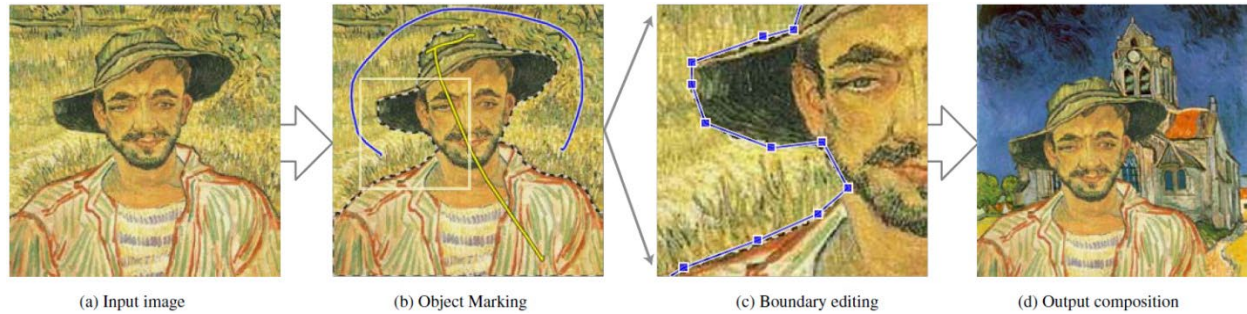
See Sections VI.B.9.a-VI.B.9.b, VI.B.4.f.

C. Ground 3: IMU, Okabe, and Li Render Obvious Claims 2-4, 10-12, and 15-17

1. Li

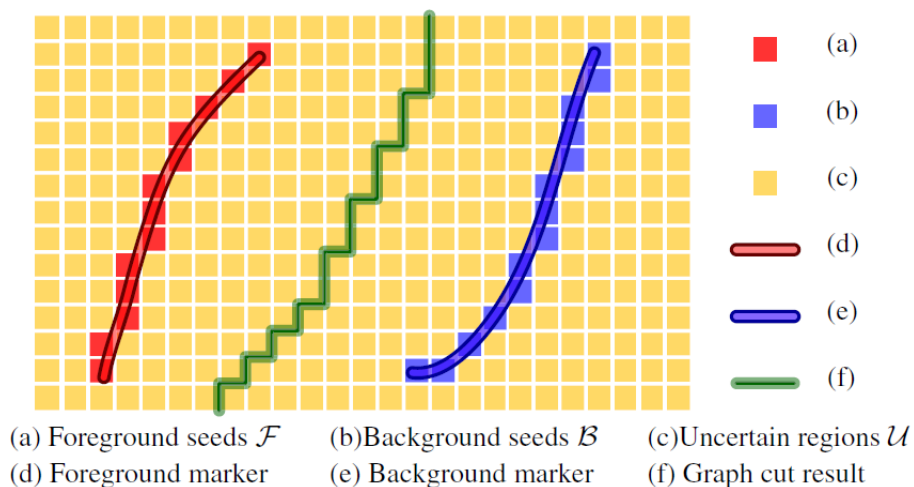
Li describes “a novel coarse-to-fine UI design for image cutout” named “Lazy Snapping.” Li, 2; Greenspun, ¶222. “The task in image cutout is specifying which parts of the image are ‘foreground’ (the part you want to cut out) and which belong to the background.” Li, 1; Greenspun, ¶222.

Li teaches that performing image cutout on a computer historically required a user to specify the foreground by “marking every pixel individually”—a “tedious” and “particularly frustrating task for users.” Li, 1; Greenspun, ¶223. Li’s “Lazy Snapping” tool, however, allows a user to easily perform image cutout in two steps: “a quick *object marking* step (b) and a simple *boundary editing* step (c).” Li, 2; Greenspun, ¶223. Li’s Figure 1 illustrates how these two steps are performed in “Lazy Snapping”:



Li, 1; Greenspun, ¶223.

As illustrated, Li teaches that the first “object marking” step “works at a coarse scale,” where the user “marks a few lines on the image by dragging the mouse cursor while holding a button (left button indicating the foreground, and right button for the background). A yellow line or a blue line is displayed for the foreground marker or background marker respectively.” Li, 2; Greenspun, ¶224. A “segmentation process” is then triggered that uses a “novel interactive graph cut algorithm” to detect a “cutout boundary” between the foreground and background lines. Li, 2; Greenspun, ¶224. Li’s Figure 2 depicts an example of detecting such a boundary (in green) between a foreground line (in red) and a background line (in blue):



The second “boundary editing” step allows the user to further refine the boundary. Li, 3; Greenspun, ¶225. This second step “works at a finer scale” by converting the boundary into “editable polygons” and allowing the user to manually refine the boundary “by simply clicking and dragging polygon vertices.” Li, 3, 2; Greenspun, ¶225.

2. The IMU-Okabe-Li Combination

a. **Motivation to Combine the IMU-Okabe Combination with Li**

The prior art contains express teachings, suggestions, and motivations for combining the IMU-Okabe Combination (Section VI.B.3.b) with Li’s “Lazy Snapping” tool (hereinafter, the “IMU-Okabe-Li Combination”).

IMU teaches that, in IMV6, a user applies a matte to make only a particular region of an image visible when animating the image using Shepard’s Distortion. IMU-Masking, 2; Section VI.B.3.a; Greenspun, ¶227. Similarly, in the IMU-Okabe Combination, a user applies a matte that, as taught in Okabe, specifies regions that the user desires to animate (in white) and not animate (in black). Okabe, 1-2; Sections VI.B.3.b, VI.B.5.a; Greenspun, ¶227.

Okabe teaches that its matte can be created “using a scribble-based image segmentation tool,” specifically Li’s “Lazy Snapping” tool. Okabe, 7, 10; Greenspun, ¶228. As such, Okabe contains an explicit teaching, suggestion, and

motivation for using Li's "Lazy Snapping" tool to create a matte in Okabe and the IMU-Okabe Combination, and thus for combining the IMU-Okabe Combination with Li. Okabe, 7; Greenspun, ¶228.

Separately, Li itself teaches that its "Lazy Snapping" tool allows a user to easily perform "image cutout"—separating and removing an image's background, leaving only the foreground visible—by using a "novel image segmentation algorithm" to detect a boundary between the foreground and background based on "a quick *object marking* step" and "a simple *boundary editing* step" performed by the user. Li, 1-2; Greenspun, ¶229. Compared to other existing methods and tools for image cutout, Li's tool "outperforms in terms of ease of use, efficiency, and quality of results." Li, 1-2; Greenspun, ¶229.

Given such benefits, a POSITA would have been motivated to utilize Li's "Lazy Snapping" tool to remove the background of an image and make only the foreground visible—i.e., to create a matte in IMV6 that makes the former portion of the image transparent and the latter portion visible. *See* IMU-Masking, 2; Section VI.B.3.a; Greenspun, ¶230. Further, given that a user of the IMU-Okabe Combination applies a matte that, as taught in Okabe, specifies regions in an image to be or not to be animated (*see* Okabe, 1; Section VI.B.3.b), a POSITA would have likewise been motivated to utilize Li's "Lazy Snapping" tool in the IMU-Okabe Combination to create such a matte for the same benefits of "ease of use, efficiency,

and quality of results” (*see* Li, 2). Greenspun, ¶230. Indeed, Okabe describes utilizing Li’s tool in this exact way, and states doing so allows a matte to be created in “less than 5 min[utes],” confirming that a POSITA would have been motivated to modify the IMU-Okabe Combination in view of Li to reach the IMU-Okabe-Li Combination. Okabe, 7; Greenspun, ¶230.

Additionally, AECS6 was an image manipulation program “readily available” by the ECPD that included not only the functionality of the IMU-Okabe Combination (*see* Section VI.B.3), but also functionality similar to Li’s “Lazy Snapping” tool (*see* Section VI.C.1). Greenspun, ¶231. In particular, compared to Li’s “object marking” step (Li, 2-3), AECS6’s “Roto Brush” tool similarly allows a user to create a matte that isolates a layer’s foreground from its background by having the user draw “strokes” on the layer over “representative areas of the foreground and background elements,” which AECS6 then uses to determine a “segmentation boundary” between the foreground and background elements via “Edge Detection” (AEM, 328-331). Greenspun, ¶231. Given AECS6 already contained functionality similar to the IMU-Okabe-Li Combination, a POSITA therefore would have been—and in fact was—motivated to further modify the IMU-Okabe Combination in view of Li to reach the IMU-Okabe-Li Combination. Greenspun, ¶231.

b. Resulting Combination

So modified, the IMU-Okabe-Li Combination allows a user to create a matte by first performing an “object marking” step of drawing marker lines on an image to indicate the foreground—the region of interest that the user desires to animate—and the background—the region of the image that the user does not desire to animate. *See* Li, 2; Okabe, 1, 7; Greenspun, ¶232. The IMU-Okabe-Li Combination utilizes Li’s “novel image segmentation algorithm” to detect the boundary between the image’s foreground and background based on the user’s marker lines. *See* Li, 1-2; Okabe, 7; Greenspun, ¶232. The IMU-Okabe-Li Combination also allows the user to perform a “boundary editing” step to manually refine the boundary. *See* Li, 2; Okabe, 7; Greenspun, ¶232. The resulting matte is then used in the remaining animation steps discussed in Section VI.B.3.b. *See* Okabe, 1, 7; Greenspun, ¶232.

3. Claim 2**a. [2a]-[2b]**

The IMU-Okabe-Li Combination allows a user to create a matte that specifies a region to animate (foreground) and a region not to animate (background). *See* Li, 1-2; Section VI.C.2.b; Greenspun, ¶233. Such a matte thus meets the claimed “mask [that] prevents pixels covered by the mask from being shifted.” Greenspun, ¶233.

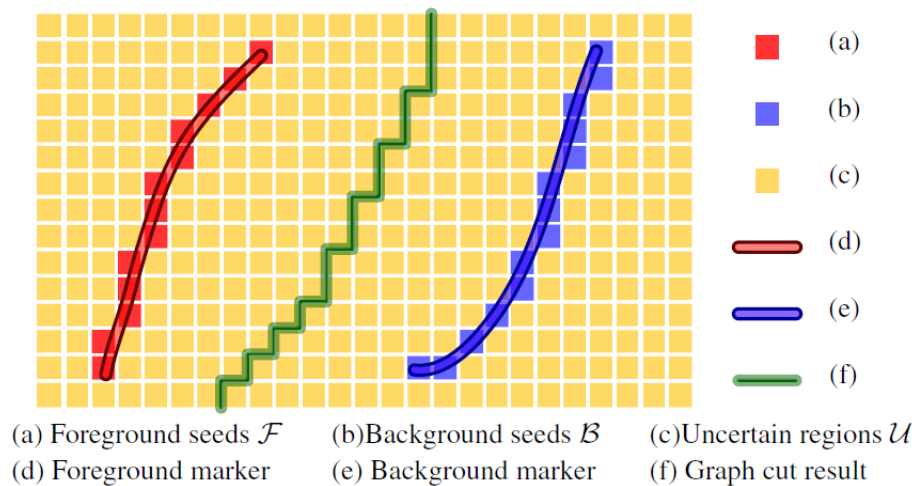
To create this matte, the IMU-Okabe Combination allows a user to perform Li’s first “object marking” step of drawing marker lines indicating either the foreground or background. *See* Li, 2; Okabe, 1, 7; Section VI.C.2; Greenspun, ¶234.

Receiving such marker lines from the user, from which the matte is created, meets “identify a particular portion of the digital image to mask.” Greenspun, ¶234.

The IMU-Okabe-Li Combination then generates a boundary based on the user’s marker lines and thereafter allows the user to perform Li’s second “boundary editing” step to manually refine the boundary. *See* Li, 2; Okabe, 7; Section VI.C.2.b; Greenspun, ¶235. Receiving such refinements from the user to modify the boundary—and the resulting matte—additionally meets the claimed “identify[ing].” Greenspun, ¶235.

4. Claim 3

Li teaches that the “object marking” step of drawing marker lines comprises drawing marker lines over the particular pixels forming the foreground and background. Li, 2; Greenspun, ¶236. Specifically, “[o]nce the user marks the image, two sets of pixels intersecting with the foreground and background markers are defined as *foreground seeds* $\mathcal{F}$ and *background seeds* $\mathcal{B}$ respectively, as shown in Figure 2”:



Li, 2-3; Greenspun, ¶236. Such foreground and background “pixels” or “seeds” are then used in Li’s segmentation algorithm to detect the boundary between the foreground and background. Li, 2-3; Greenspun, ¶236. Each “pixel[]” or “seed[]” meets a “particular pixel within the digital image.” Li, 2-3; Greenspun, ¶236. Receiving the foreground and background marker lines, and their intersecting “pixels” or “seeds,” meets the claimed “receiving.” Li, 2-3; Greenspun, ¶236.

5. Claim 4

a. [4a]-[4c]

Li teaches that, in the “object marking” step, a user draws marker lines over the particular “pixels” or “seeds” that form the foreground and background. Li, 2-3; Section VI.C.4; Greenspun, ¶237. Such “pixels” or “seeds” are used to detect the boundary between the foreground and background for generating the matte, meeting the claimed language. Li, 2-3; Greenspun, ¶237.

6. Claim 10

a. [10a]-[10b]

See Sections VI.C.3.a, VI.C.4.

7. Claim 11

See Section VI.C.4.

8. Claim 12

a. [12a]-[12b]

See Section VI.C.5.a.

1. Claim 15

a. [15a]-[15b]

See Sections VI.C.3.a, VI.C.4.

2. Claim 16

See Section VI.C.4.

3. Claim 17

a. [17a]-[17c]

See Section VI.C.5.a.

VII. DISCRETIONARILY DENIAL IS UNWARRANTED

In the Litigation, the court granted Petitioner’s motion to dismiss because the ’017 Patent recites patent-ineligible subject matter under 35 U.S.C. §101. EX1021, 17-35. Plotagraph appeals the dismissal, which is now pending. *Id.*, 1. Such a pending Appeal that reviews “only the §101 issues” does not present concerns of efficiency and duplicative efforts sufficient to warrant discretionary denial of this

IPR2023-00153

U.S. Patent No. 10,346,017

Petition under 35 U.S.C. §314(a), as “it is unlikely that any district court trial on any §[§]103 [or 102] issues would occur prior to [the Board] issuing a final written decision.” *Stripe, Inc. v. Boom! Payments, Inc.*, CBM2020-00002, Paper 22, 8-10 (PTAB May 19, 2020).

VIII. CONCLUSION

For the above reasons, institution is appropriate.

Dated: Jan. 17, 2023

/Robert Greeson/

Robert Greeson (Reg. No. 52,966)

CERTIFICATE OF WORD COUNT

Pursuant to 37 C.F.R. §42.24(a)(1)(i), the undersigned certifies that this Paper—exclusive of the table of contents, table of authorities, mandatory notices under §42.8, certificate of service, this certificate of word count, appendix of exhibits, and claim listing—includes 13,766 words based on the word count of the word-processing system used to prepare the paper.

/Robert Greeson/

Robert Greeson (Reg. No. 52,966)

CERTIFICATE OF SERVICE

Pursuant to 37 C.F.R. §42.6(e) and 37 C.F.R. §42.105(a), the undersigned certifies that on January 17, 2023, complete copies of this Petition for *Inter Partes* Review, Petitioner's power of attorney, and all exhibits were served on Patent Owner at the correspondence address of record listed below by FedEx®:

WORKMAN NYDEGGER
60 EAST SOUTH TEMPLE
SUITE 1000
SALT LAKE CITY, UT
UNITED STATES

/Robert Greeson/
Robert Greeson (Reg. No. 52,966)