

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 IPR2023-00568
U.S. Patent No. 11,182,641
Filing Date: Feb. 26, 2020
Issue Date: Nov. 23, 2021

**PETITION FOR *INTER PARTES* REVIEW OF
U.S. PATENT NO. 11,182,641 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 '641 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.....	8
A.	Ground 1: Claims 1-4, 8-15, and 19-20 Are Anticipated or Otherwise Rendered Obvious by AEM.....	8
1.	Summary of AEM.....	8
a.	Workspace and Panels	10
b.	Compositions and Layers	10
c.	Extracting a Single Frame from a Video.....	11
d.	Modifying Transparency of Certain Pixels	12
e.	Puppet Effect	14
f.	Animating with the Puppet Effect	15
2.	Independent Claim 1	17
a.	[1pre].....	17
b.	[1a].....	18

c.	[1b].....	20
d.	[1c].....	26
e.	[1d].....	27
f.	[1e].....	28
g.	[1f]	32
3.	Claim 2.....	35
4.	Claim 3.....	36
5.	Claim 4.....	36
6.	Claim 8.....	36
7.	Claim 9.....	40
a.	[9a]-[9e]	40
8.	Claim 10.....	43
9.	Claim 11	43
10.	Independent Claim 12	46
a.	[12pre].....	46
b.	[12a].....	47
c.	[12b].....	47
d.	[12c].....	47
e.	[12d].....	47
f.	[12e].....	48
11.	Claim 13.....	48
a.	[13a]-[13b]	48
12.	Claim 14.....	51
13.	Claim 15	52
a.	[15a]-[15b]	52
14.	Independent Claim 19	54
a.	[19pre].....	54
b.	[19a].....	54
c.	[19b].....	55
d.	[19c].....	55

e.	[19d].....	55
f.	[19e].....	55
g.	[19f]	55
15.	Claim 20.....	55
B.	Ground 2: IMU and Okabe Render Obvious Claims 1-4, 8-14, and 19-20.....	55
1.	Summary of IMU	55
a.	Applying Effects and Creating an Animation	56
b.	Extracting a Single Frame of a GIF Animation	59
c.	Modifying Transparency of Certain Pixels Using a Matte	61
2.	Summary of Okabe	62
3.	The IMU-Okabe Combination.....	63
a.	Motivation to Combine IMU with Okabe	63
b.	Resulting Combination	67
4.	Independent Claim 1	68
a.	[1pre].....	68
b.	[1a].....	69
c.	[1b].....	69
d.	[1c].....	72
e.	[1d].....	73
f.	[1e].....	74
g.	[1f]	75
5.	Claim 2.....	76
6.	Claim 3.....	76
7.	Claim 4.....	76
8.	Claim 8.....	76
9.	Claim 9.....	77
a.	[9a]-[9e]	77
10.	Claim 10.....	79
11.	Claim 11	80

12.	Independent Claim 12	81
a.	[12pre]	81
b.	[12a]	82
c.	[12b]	82
d.	[12c]	82
e.	[12d]	83
f.	[12e]	83
13.	Claim 13	83
a.	[13a]-[13b]	83
14.	Claim 14	84
15.	Independent Claim 19	84
a.	[19pre]	84
b.	[19a]	85
c.	[19b]	85
d.	[19c]	85
e.	[19d]	85
f.	[19e]	85
g.	[19f]	85
16.	Claim 20	85
C.	Ground 3: IMU, Okabe, and Li Render Obvious Claims 13-15	86
1.	Summary of Li	86
2.	The IMU-Okabe-Li Combination	88
a.	Motivation to Combine the IMU-Okabe Combination with Li	88
b.	Resulting Combination	90
3.	Claim 13	91
a.	[13a]-[13b]	91
4.	Claim 14	91
5.	Claim 15	92
a.	[15a]-[15b]	92

IPR2023-00568

U.S. Patent No. 11,182,641

VII. DISCRETIONARY DENIAL IS UNWARRANTED 93

VIII. CONCLUSION.....93

Petitioner's Exhibit List

Exhibit	Shorthand	Description
1001	'641 Patent	U.S. Patent No. 11,182,641
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	'641 PH	Excerpts of Prosecution history of U.S. Patent No. 11,182,641
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' Opposed Motion for Leave to Serve Second Supplemental Disclosures Pursuant to PR. 3-1 and P.R. 3-2, <i>Plotagraph, Inc. v. Lightricks Ltd.</i> , Civil Action No. 4:21-cv-03873, Dkt. No. 42 (S.D. Tex. May 21, 2022)
1023	Hair	U.S. Patent No. 6,014,491
1024	Nakagawa	U.S. Patent No. 5,835,911
1025		WayBack Machine Capture dated Sept. 7, 2012 of <i>Downloads</i> , Adobe (Sept. 7, 2012), http://www.adobe.com/downloads/ [https://web.archive.org/web/20120907000000/http://www.adobe.com/downloads/]

IPR2023-00568

U.S. Patent No. 11,182,641

		archive.org/web/20120907130453/http://www.adobe.com/downloads/]
1026		WayBack Machine Capture dated Mar. 28, 2012 of <i>Download ImageMagick</i> , ImageMagick (Mar. 28, 2012), http://www.imagemagick.org/script/download.php [https://web.archive.org/web/20120328075001/http://www.imagemagick.org/script/download.php]

CLAIM LISTING

Claim	Limitation No.	Limitation
1	[1pre]	A computer system providing, to a client computing device, software for automating a shifting of pixels within a video file, the computer system comprising: one or more processors; and one or more computer-readable media having stored thereon executable instructions that are transmitted to the client computing device for execution by one or more client processors on the client computing device, the executable instructions comprising instructions that when executed by the one or more client processors configure the client computing device 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 a frame of the digital image file;
	[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 a first image frame;
	[1c]	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;
	[1d]	create a first digital link between the first starting point and the first ending point, wherein the first digital 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;
	[1e]	identify a first set of pixels that lie along the first digital link between the first starting point and the first ending point; and

Claim	Limitation No.	Limitation
	[1f]	shift the first set of pixels in the first direction.
2		The computer system of claim 1, wherein the first ending portion comprises a particular portion of the first image frame.
3		The computer system of claim 1, wherein the digital image file comprises a video file and the first image frame comprises a first video frame of the video file.
4		The computer system of claim 3, wherein the first ending portion comprises a particular portion of a second video frame within the video file.
8		The computer system of claim 1, wherein shifting the first set of pixels comprises rendering in a loop the first set of pixels being shifted within the first image frame.
9	[9a]	The computer system of claim 1, wherein the executable instructions include instructions that are executable to configure the computer system to: receive a second starting point through the user interface, wherein the second starting point is received through a user selection of a second beginning portion of the first image frame;
	[9b]	receive a second ending point through the user interface, wherein the second ending point is received through a user selection of a second ending portion;
	[9c]	create a second digital link between the second starting point and the second ending point, wherein the second digital link comprises: a second direction extending from the second starting point to the second ending point; and a second length between the second starting point and the second ending point;
	[9d]	identify a second set of pixels that lie between the second starting point and the second ending point; and
	[9e]	shift the second set of pixels in the second direction.

Claim	Limitation No.	Limitation
10		The computer system of claim 9, wherein the first direction is different from the second direction.
11		The computer system of claim 9, wherein a 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.
12	[12pre]	A computer program product comprising one or more non-transitory computer storage media having stored thereon computer-executable instructions that, when transmitted to a remote computer system for execution at a processor, cause the remote computer system to perform a method for automating a shifting of pixels within an image file, the method comprising:
	[12a]	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 portion of a first image frame;
	[12b]	receiving, through the user interface, a first direction associated with the first starting point;
	[12c]	creating a first digital link extending in the first direction from the first starting point;
	[12d]	selecting a first set of pixels that are along the first digital link and extend in the first direction away from the first starting point; and
	[12e]	shifting the first set of pixels, in the first image frame, in the first direction.
13	[13a]	The computer program product as recited in claim 12, further comprising receiving an indication to generate a first mask over a second portion of the first image frame,
	[13b]	wherein pixels under the first mask are prevented from shifting.
14		The computer program product as recited in claim 13, further comprising computer-executable instructions that, when transmitted to the remote computer system for execution at the processor, cause the remote computer system to perform a method for automating the shifting of pixels within the image file, the method comprising

Claim	Limitation No.	Limitation
		receiving through a user interface a selection of the second portion of the first image frame around which the first mask should be generated.
15	[15a]	The computer program product of claim 14, further comprising computer-executable instructions that, when transmitted to the remote computer system for execution at the processor, cause the remote computer system to perform a method for automating the shifting of pixels within the image file, the method comprising: identifying one or more edges that form a first boundary around the second portion; and
	[15b]	generating the first mask to cover area within the first boundary.
19	[19pre]	A method for transmitting to a client computing device instructions for shifting pixels within a video file, comprising: transmitting computer executable instructions to a client computing device, the computer executable instructions configured to cause the client computing device to:
	[19a]	access, from memory, a digital image file, wherein the digital image file comprises information that corresponds to individual pixels within a frame of the digital image file;
	[19b]	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 a first image frame;
	[19c]	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;
	[19d]	create a first digital link between the first starting point and the first ending point, wherein the first digital link comprises:

Claim	Limitation No.	Limitation
		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;
	[19e]	identify a first set of pixels that lie along the first digital link between the first starting point and the first ending point; and
	[19f]	shift the first set of pixels in the first direction.
20	The method of claim 19, wherein the digital image file comprises a video file and the first image frame comprises a frame of the video file.	

I. INTRODUCTION

The '641 Patent's claims are directed to "automating a shifting of pixels within a video file." Pixels are shifted according to user inputs, i.e., the user indicates starting and ending points in a video frame for a digital link and a set of pixels to be shifted, and the pixels are shifted in the digital link's direction.

However, the foregoing features were included in image editing programs prior to the '641 Patent's priority 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 1-4, 8-15, and 19-20 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 '641 Patent is available for *inter partes* review ("IPR") and Petitioner is not barred or estopped from requesting IPR on the identified grounds.

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

Claims 1-4, 8-15, and 19-20 (the "Challenged Claims").

2. Grounds of Challenge²

Petitioner requests IPR of the Challenged Claims on the following grounds:

Ground	Claims	Statutory Basis	Prior Art
1	1-4, 8-15, and 19-20	§§102(a)(1), 103	AEM
2	1-4, 8-14, and 19-20	§103	IMU and Okabe
3	13-15	§103	IMU, Okabe, and Li

Supporting analysis is provided in the Declaration of Dr. Philip Greenspun.

AEM is a user manual for Adobe After Effects CS6 (“AECS6”). *See, e.g.*, AEM, 3; Greenspun, ¶49. 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, ¶49 (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 ImageMagick Version 6 ("IMV6"). *See* IMU-Home, 1; Greenspun, ¶50. The section's homepage (IMU-Home) 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, ¶50. Each of the above was publicly accessible 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, ¶50 (testifying IMU, which he provided, matches copy at Archive, 4-96, excluding declarant-provided cover pages). Other references confirm IMU's pre-ECPD public availability. *See, e.g.,* Siddharth, ¶¶120-22; Brookhart, 10:53-59.

³ IMU is thus a single reference. Greenspun, ¶50 n.7. Separately, a POSITA would have also been motivated to consider the IMU webpages together to gain a more complete understanding of IMV6. *Id.*

Okabe was published in Volume 30, Number 7 of Computer Graphics Forum in September 2011 and is therefore prior art. Okabe, 1; Greenspun, ¶51; *see also*, *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, ¶52; *see also*, *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 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 understood how to implement such systems. *Id.* The '641 Patent and prior art discussed in Section VI evidence this level of ordinary skill.

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), 30-46 (technology background).

V. THE '641 PATENT

The '641 Patent is directed to “systems, methods, and computer-readable media that automate the shifting of pixels within a digital video file.” '641 Patent, 3:13-16; Greenspun, ¶47. A user provides a video file and selects a “starting point” on a video frame, as well as an “ending point” on the same or a different frame. '641 Patent, Abst., 6:33-54; Greenspun, ¶47. The system or user then creates a “digital link” between the starting and ending points and identifies a set of pixels that includes at least “a line of individual pixels extending from the starting point to the ending point” but may, “at another extreme,” include “a relatively wide swatch of pixels that are parallel to the link that extends between the starting point and the ending point.” '641 Patent, 6:58-67, 7:22-36; Greenspun, ¶47. The set of pixels is then shifted in the link’s direction. '641 Patent, 7:37-40; Greenspun, ¶47. Figure 9 is representative:

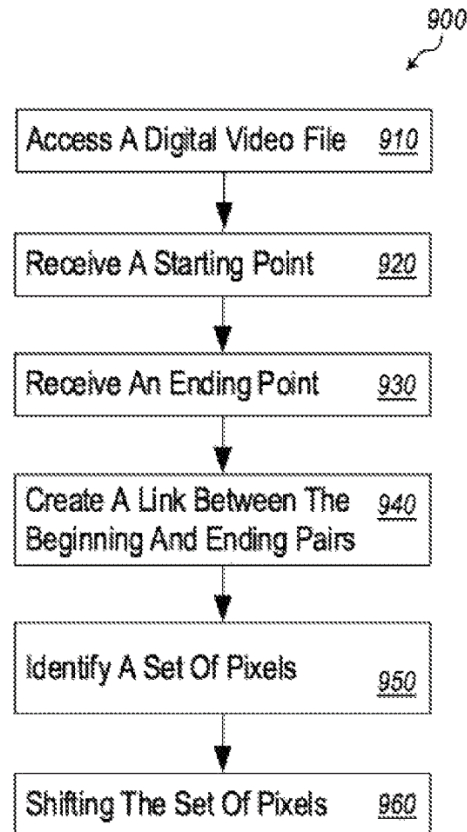


Fig. 9

'641 Patent, Fig. 9 ;Greenspun, ¶47.

A. Prosecution History

During prosecution, the '641 Patent faced no rejections on the basis of prior art. *See* '641 PH, 97-103, 188-190, 208, 218. Applicant's only amendments were of claims 12-18, which were solely to overcome a 35 U.S.C. §101 rejection for failing to recite "non-transitory computer storage media" and to address informalities. *See id.*, 132-140, 98-99.

B. Claim Construction

For purposes of this IPR, all terms should be given their plain and ordinary meaning. Greenspun, ¶48.

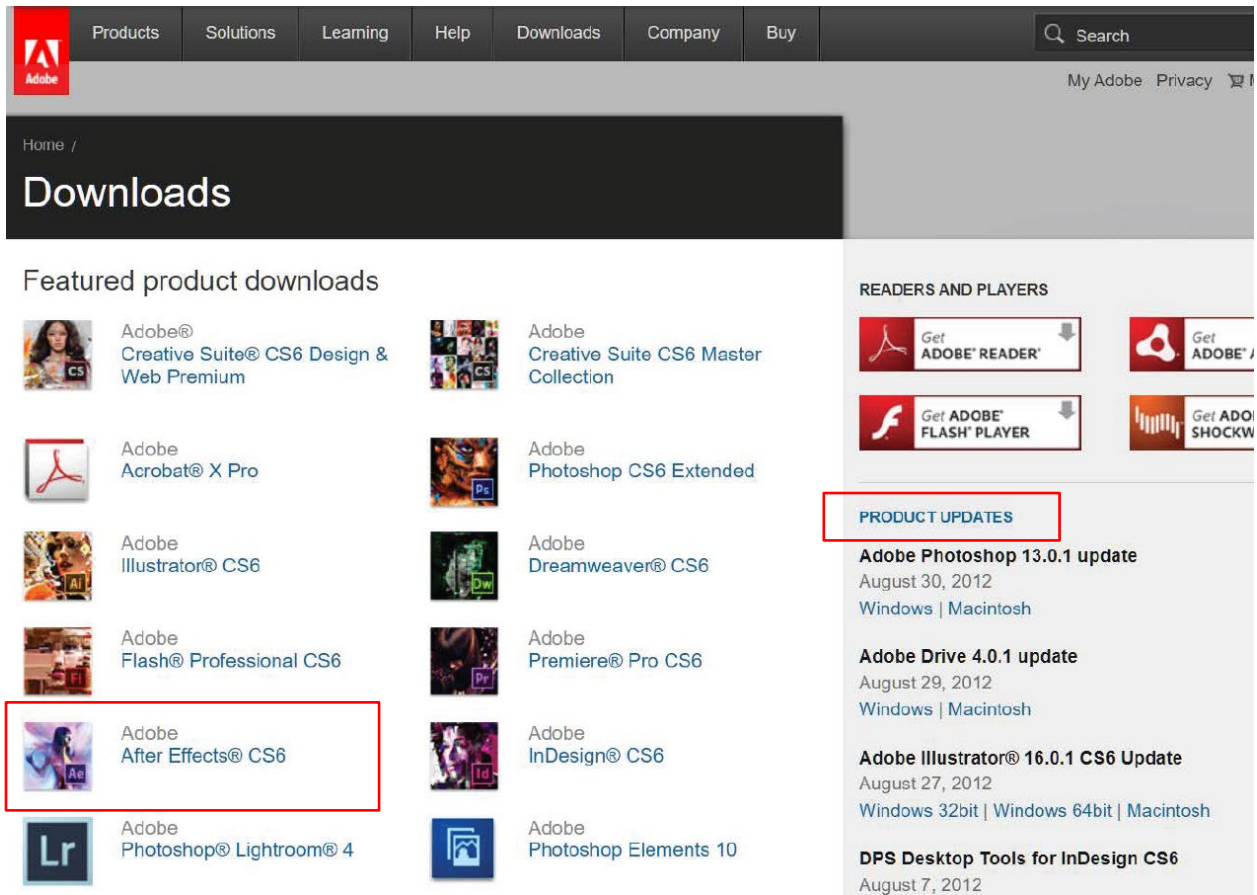
VI. THE CHALLENGED CLAIMS ARE UNPATENTABLE UNDER 35 U.S.C. §§102(A)(1) AND 103

A. Ground 1: Claims 1-4, 8-15, and 19-20 Are Anticipated or Otherwise Rendered Obvious by AEM

1. Summary of AEM

AEM is a user manual for AECS6—a software for performing a variety of animation tasks from, e.g., “animat[ing] a simple title” to “creat[ing] complex motion graphics, or composit[ing] realistic visual effects.” AEM, 25; Greenspun, ¶53.

AEM instructs to “[m]ake sure that you’ve installed the current version of [AECS6], including any available updates.” AEM, 517. To view such updates, AEM instructs to “go to the Downloads section of the Adobe website,” and provides a hyperlink for doing so. AEM, 517; Greenspun, ¶54. As a POSITA would have known, clicking such a hyperlink would have directed the user to the “Downloads” webpage of the Adobe website shown in EX1025, from which AECS6, including any updates, could indeed be downloaded and installed AECS6:



EX1025 (annotations added); AEM, 517; Greenspun, ¶54 & n.8 (testifying EX1025, which he provided, shows the Wayback Machine’s capture of this “Downloads” webpage on the same day as AEM’s capture date of September 7, 2012, and thus shows the webpage that would have resulted upon clicking the aforementioned hyperlink on that date).

AEM also explains how to navigate AECS6’s workspace and panels, define “compositions” and “layers,” extract a single frame of a video, apply a “mask” or “matte” to the frame, and animate the frame 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, ¶56. Different panels contain different tools and effects for editing and animating images in AECS6. Greenspun, ¶56; *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, ¶57. "Footage items," such as images and videos, are imported into a composition. AEM, 75-76; Greenspun, ¶57; *see also* AEM, 101 ("Imported footage items appear in the Project panel."). AEM lists specific "[s]upported import formats" of such footage items, including for example "Still-image formats" such as JPEG, and "Video and animation formats" such as MOV and animated GIF. AEM, 98-101; Greenspun, ¶57. Once imported, a footage item becomes a "layer" in that composition. AEM, 75, 98; Greenspun, ¶57; *see also* AEM, 121 ("You can create a layer from any footage item in the Project panel..." by "[s]electing one or more footage items and folders in the Project panel" and "[d]rag[ging] the selected footage items to the Composition panel.").

Layers are "the elements that make up a composition" and are stacked in a "vertical arrangement" such that the uppermost layer is visible. AEM, 120, 125; Greenspun, ¶58.

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, ¶59. 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, ¶59. When ready, the user “render[s] [the] composition to create the frames of a final output movie.” AEM, 75; Greenspun, ¶59.

c. Extracting a Single Frame from a Video

While AEM teaches creating a composition using a video (AEM, 75, 98-101), AEM also teaches extracting a “single frame” from such a composition and thus the video, which “is useful for,” e.g., “exporting an image from a movie for posters or storyboards” (*id.*, 590). Greenspun, ¶60. To do so, AEM instructs to “[g]o to the frame that you want to export so that it is shown in the Composition panel.” AEM, 590; Greenspun, ¶60. Then, “choose Composition > Save Frame As > File. Adjust settings in the Render Queue panel if necessary, and then click Render.” AEM, 590; Greenspun, ¶60.

The aforementioned “Render Queue panel” allows management of various “render settings and output module settings” for the frame, including “output format.” AEM, 572; Greenspun, ¶61. “Supported output formats” include many of the same formats listed as “[s]upported import formats” for footage items discussed in Section VI.A.1.b, such as JPEG, PNG, and other “Still-image formats.” *Compare*

AEM, 573-74, *with* AEM, 98-101. Greenspun, ¶61. Indeed, AEM teaches importing the frame as a new footage item “by dragging its output module from the Render Queue panel into the Project panel.” AEM, 574-75; Greenspun, ¶61. Such provides “a convenient way to convert a footage item from one format to another,” e.g., to extract a single frame from a video as a footage item for use as a new layer of the composition. AEM, 573-75, 98-101, 590; Greenspun, ¶61.

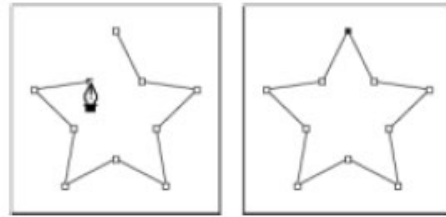
d. Modifying Transparency of Certain Pixels

AEM describes tools for modifying 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, ¶62. Masks and mattes modify the layer’s “alpha channel,” which “determines the transparency of the layer at each pixel.” AEM, 318; Greenspun, ¶62.

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, ¶63. AEM provides an example composition using two layers and a rectangular mask:



AEM, 318; Greenspun, ¶63. Masks are created 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 mask's path and vertices. AEM, 264-66; Greenspun, ¶63. AEM depicts an exemplary star-shaped mask created accordingly:



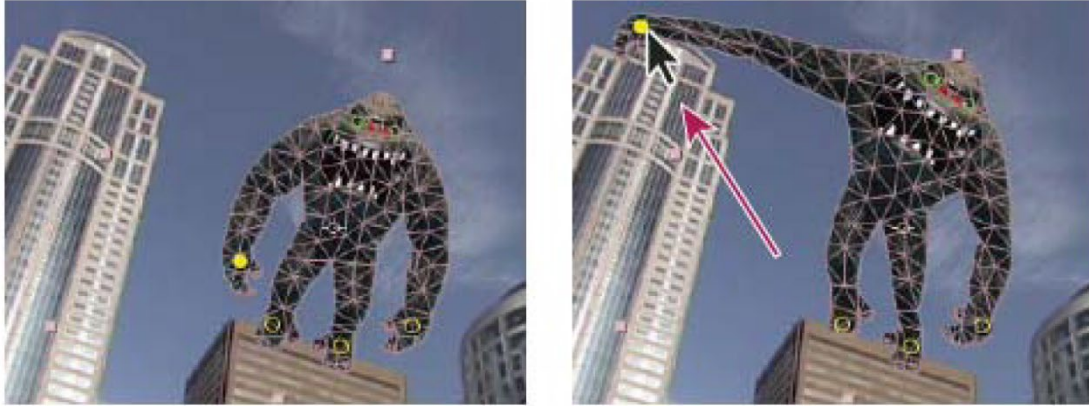
AEM, 265; Greenspun, ¶63. Masks can also be automatically created using AECS6's "Auto-trace" function, which creates a mask by "searching for edges" across a layer and tracing such edges. AEM, 262; Greenspun, ¶63. Such edges are detected based on "the alpha, red, green, blue, or luminance channel of [the] layer." AEM, 262; Greenspun, ¶63.

Mattes similarly "define[] the transparent areas" of a layer. AEM, 317; Greenspun, ¶64. Specifically, a matte isolates a layer's "foreground" from its "background," making the latter transparent. AEM, 328; Greenspun, ¶64. A matte is created 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, ¶64. AECS6 then uses "Edge Detection" to determine a "segmentation boundary" separating the drawn-ver foreground and background elements. AEM, 328-31; Greenspun, ¶64.

e. 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, ¶65. 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, ¶65. 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, ¶65. 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, ¶65. 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, ¶65. The user can add more Deform pins to move other portions of the layer and its mesh. AEM, 218; Greenspun, ¶65.

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



AEM, 218; Greenspun, ¶66. 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, ¶66. 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 "rigid." *See* AEM, 218; Greenspun, ¶66; *see also* AEM, 219.

f. 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, ¶67. To do so, AEM instructs to define a starting time point and starting position for a Deform pin, and then define an ending time point and ending position for the Deform pin. AEM, 177, 219; Greenspun, ¶67. This is known as defining starting and ending "keyframes" for the Deform pin, respectively. AEM, 177, 219; Greenspun, ¶67. AECS6 then interpolates the Deform pin's position from the starting time and position (the starting "keyframe") to the ending time and position (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, ¶67. This is known as “keyframe interpolation.” AEM, 177, 193; Greenspun, ¶67.

To define a starting “keyframe” for a Deform pin, AEM instructs 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, ¶68; *see also* AEM, 166 (describing the “current-time indicator (CTI)” in the “Timeline panel,” which allows a user 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, ¶68.

AEM then instructs 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, ¶69.

Once such keyframes 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, ¶70. AECS6 performs such interpolation for all keyframes of a Deform pin. AEM, 193, 219; Greenspun,

¶70. The user may place additional Deform pins to animate other parts of the layer in the same way. AEM, 219; Greenspun, ¶70.

Further, keyframe interpolation of a Deform pin's change in position follows a user-defined path—a “motion path.” AEM, 187, 219; Greenspun, ¶71. 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, ¶71.

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, ¶72. 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, ¶72. This loop repeats until specified otherwise. AEM, 562, 219; Greenspun, ¶72.

2. Independent Claim 1

a. [1pre]

To the extent [1pre] is limiting, AEM teaches that AECS6 is “software” that is installed on a user's computer and run on “64-bit operating systems.” AEM, 19; Greenspun, ¶101. Such installation of AECS6 on a user's computer discloses the

claimed “computer system” and remaining claim language according to Plotagraph’s infringement contentions in the Litigation. *See* EX1022, 134; Greenspun, ¶101.

Alternatively, AEM instructs to “[m]ake sure that you’ve installed the current version of [AECS6], including any available updates.” AEM, 517; Greenspun, ¶102. To view such updates, AEM instructs to “go to the Downloads section of the Adobe website,” and provides a hyperlink for doing so. AEM, 517; Greenspun, ¶102. A user following these instructions would have downloaded and installed the most up-to-date version of AECS6 (the claimed “software”) onto the user’s computer (“client computing device” with “one or more client processors”) from Adobe’s website, which would have been provided by a server (*see* Nakagawa, Abst.) (“computer system providing, to a client computing device, software” and comprising “one or more processors” and “one or more computer-readable media having stored thereon executable instructions that are transmitted to the client computing device for execution by one or more client processors”). Greenspun, ¶102.

Further, AEM describes using AECS6 to animate the movement of pixels within a single frame of a video (“automating a shifting of pixels within a video file”). Sections VI.A.2.b-VI.A.2.g; Greenspun, ¶103.

b. [1a]

AEM teaches importing video “footage items” from a “local disk drive” into a composition in AECS6—“access, from memory, a digital image file....” AEM,

75, 98-100 (listing “[v]ideo and animation formats” of footage items supported for importing, including MOV, MPEG, and animated GIF),⁵ 17; ’641 Patent, claim 3; Greenspun, ¶104; *see also* AEM, 90 (“Compositions and footage items are listed in the project panel.”), 121 (“You can create a layer from any footage item in the Project panel” by “[s]electing one or more footage items and folders in the Project panel” and “[d]rag[ging] the selected footage items to the Composition panel.”).

By way of further illustration, an annotated screenshot of AECS6 is provided below,⁶ which shows an exemplary composition where a MOV video of a smokestack has been imported as a layer:

⁵ A POSITA would have known that such video files “comprise[] information that corresponds to individual pixels within a frame” as claimed. *See* AEM, 104, 105-08; Hair, 1:15-21, 1:45-49; Greenspun, ¶104, n.12.

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, ¶105.⁷

c. [1b]

AEM teaches, after importing a video into a composition, extracting a single frame from the composition (and thus the video)—“first image frame”—to be a new layer in the composition, which “is useful for,” e.g., “exporting an image from a movie for posters or storyboards.” AEM, 590; Greenspun, ¶106. To do so, AEM instructs to “[g]o to the frame that you want to export so that it is shown in the

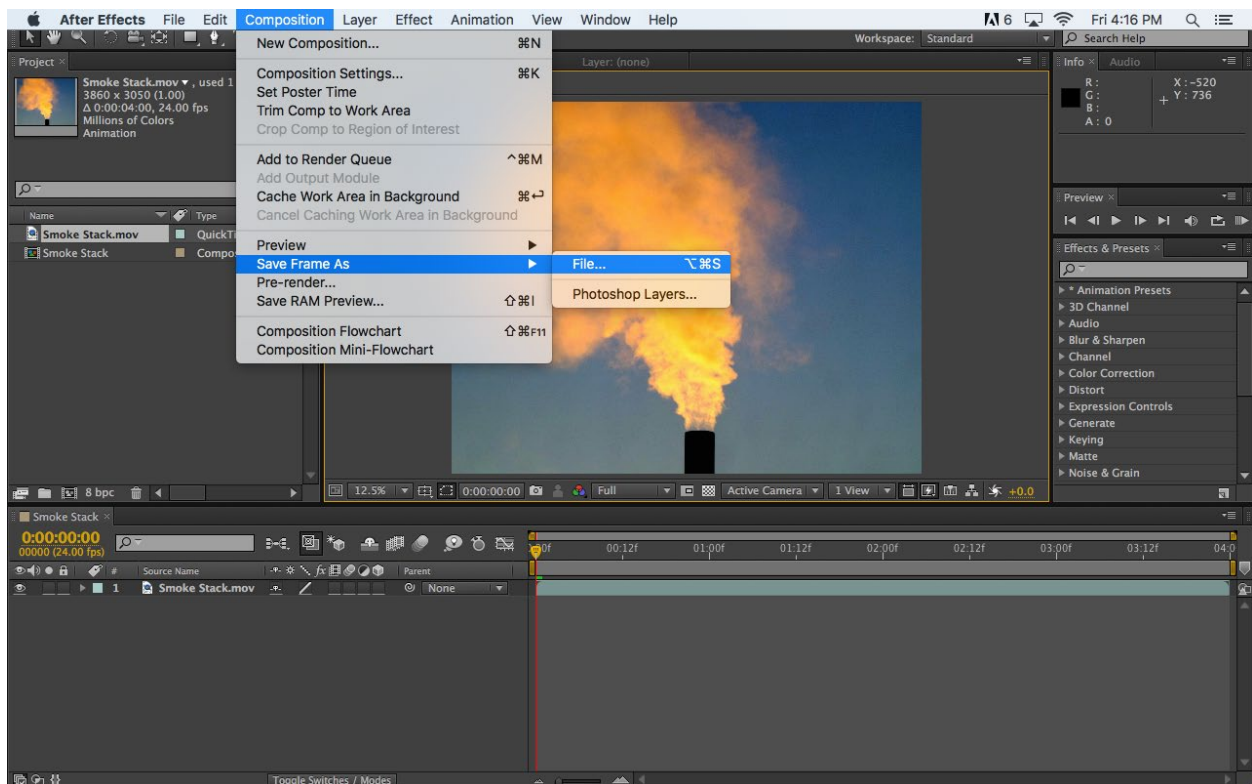
⁷ Dr. Greenspun also explains step-by-step how the screenshots here and below were created. See Greenspun, ¶¶73-100.

IPR2023-00568

U.S. Patent No. 11,182,641

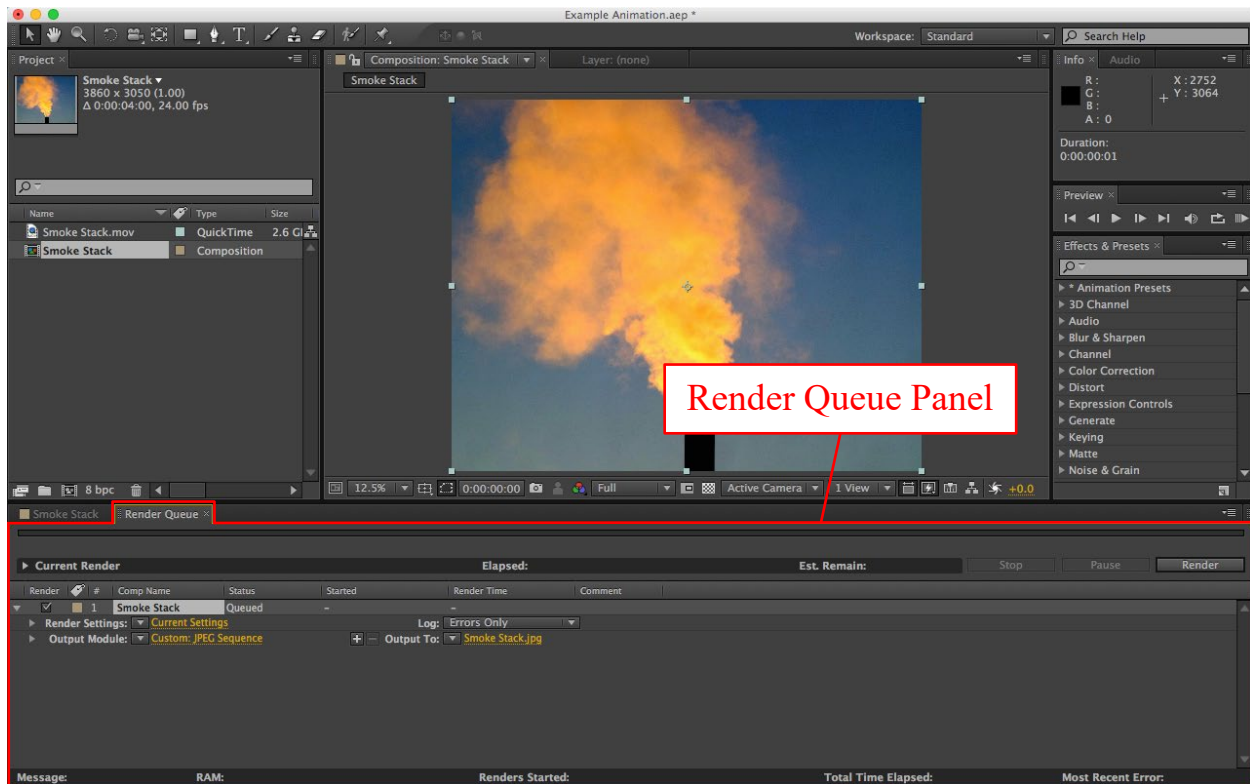
Composition panel.” AEM, 590; Greenspun, ¶106. Then, “choose Composition > Save Frame As > File.” AEM, 590; Greenspun, ¶106.

To further illustrate, the below screenshot shows the exemplary composition from Section VI.A.2.b, where the first frame of the video (at “0:00:00:00”) is now being extracted according to AEM’s discussed instructions:



Greenspun, ¶107.

AEM also teaches managing the “render” and “output module” settings for the frame using the “Render Queue panel,” which is highlighted in the below annotated screenshot:



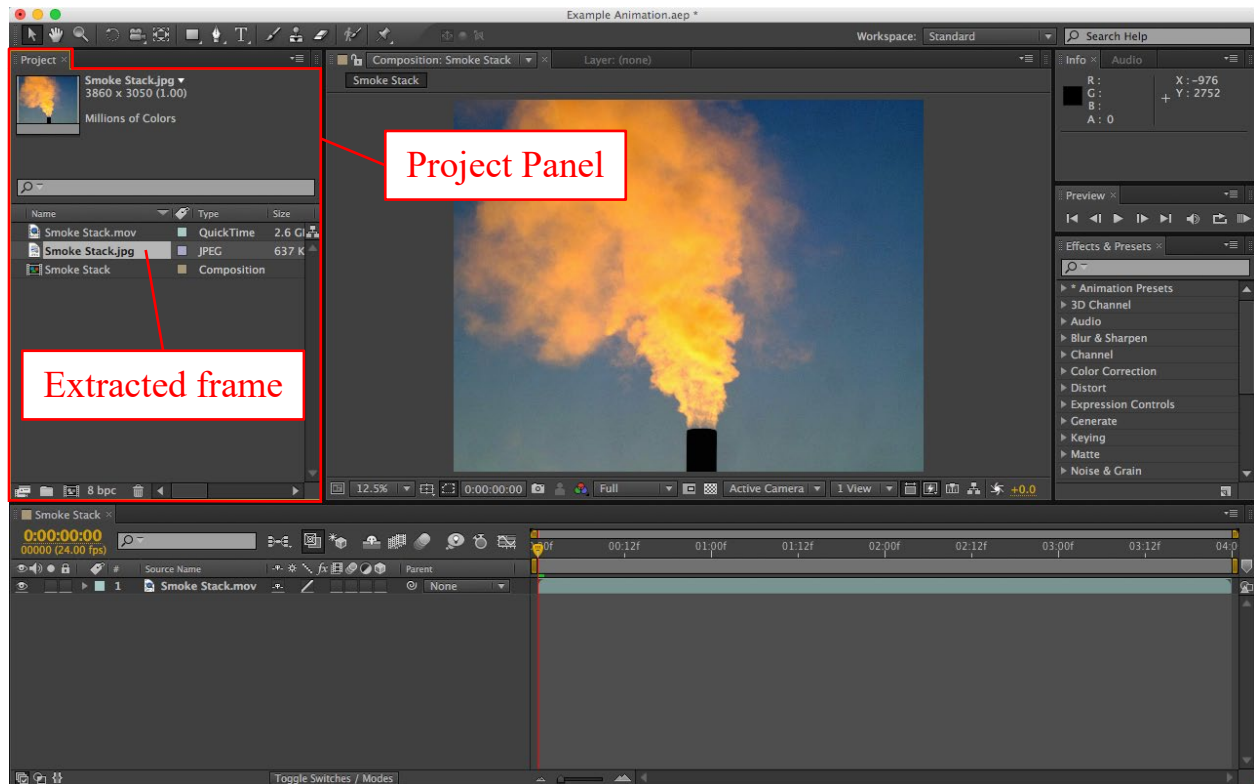
AEM, 573-74; Greenspun, ¶108. One such manageable setting in the Render Queue panel is “output format,” which may be adjusted so the extracted frame can be used later as a footage file, e.g., by setting the output format to be JPEG, which is a “[s]till-image format[]” supported as both an output format and a footage item import format. *Compare* AEM, 573-74, with *id.*, 98-101. Greenspun, ¶108. Indeed, AEM teaches importing the extracted frame as a new footage item “by dragging its output module from the Render Queue panel into the Project panel,” and then creating a layer from the frame by selecting the frame in the Project panel and “[d]rag[ging] the selected footage item[] to the Composition panel.” AEM, 574-75, 121; Greenspun, ¶108. Such provides “a convenient way to convert a footage item from

IPR2023-00568

U.S. Patent No. 11,182,641

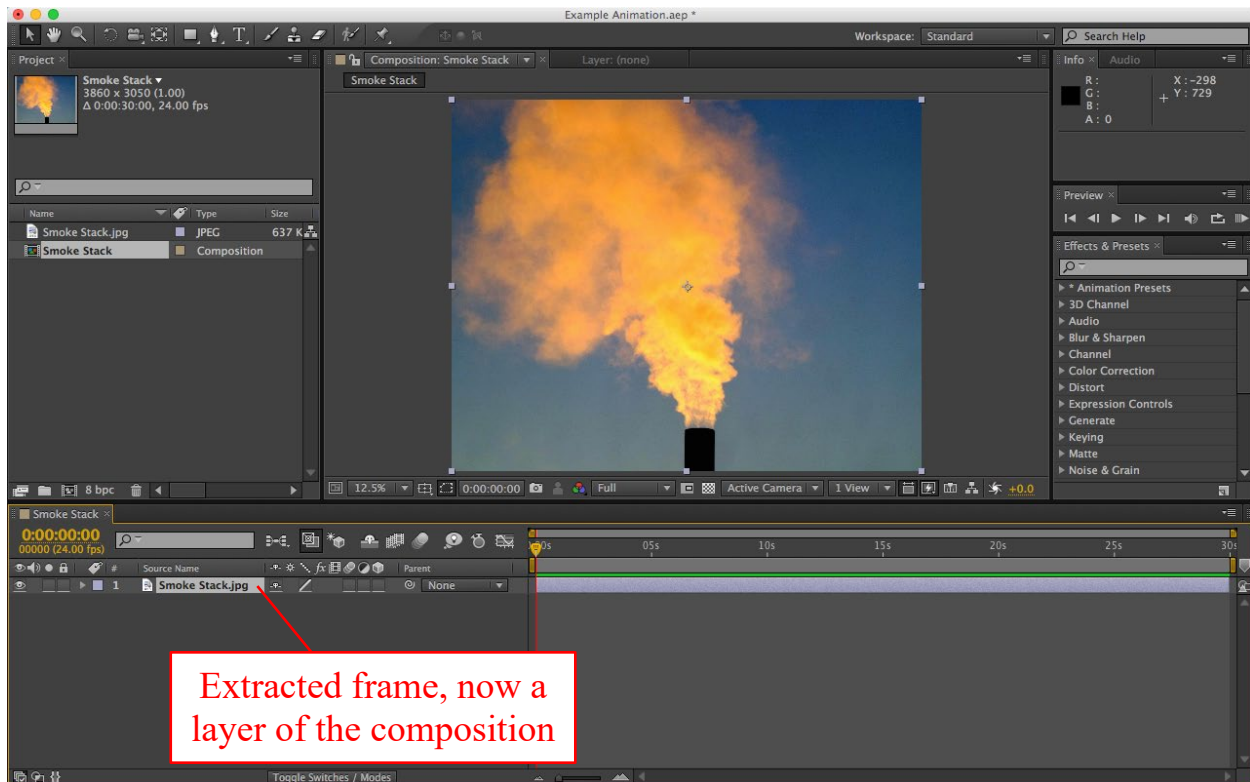
one format to another,” e.g., to extract and import a single frame from a video as a new layer in the composition. AEM, 574-75, 98-101, 590; Greenspun, ¶108.

To further illustrate, the below annotated screenshot shows the result of extracting the first frame of the above screenshot’s video as a JPEG and dragging the frame’s output module to the Project panel:



Greenspun, ¶109. The frame is then selected and dragged to the Composition panel, causing the frame to become a new layer of the composition, as shown below⁸:

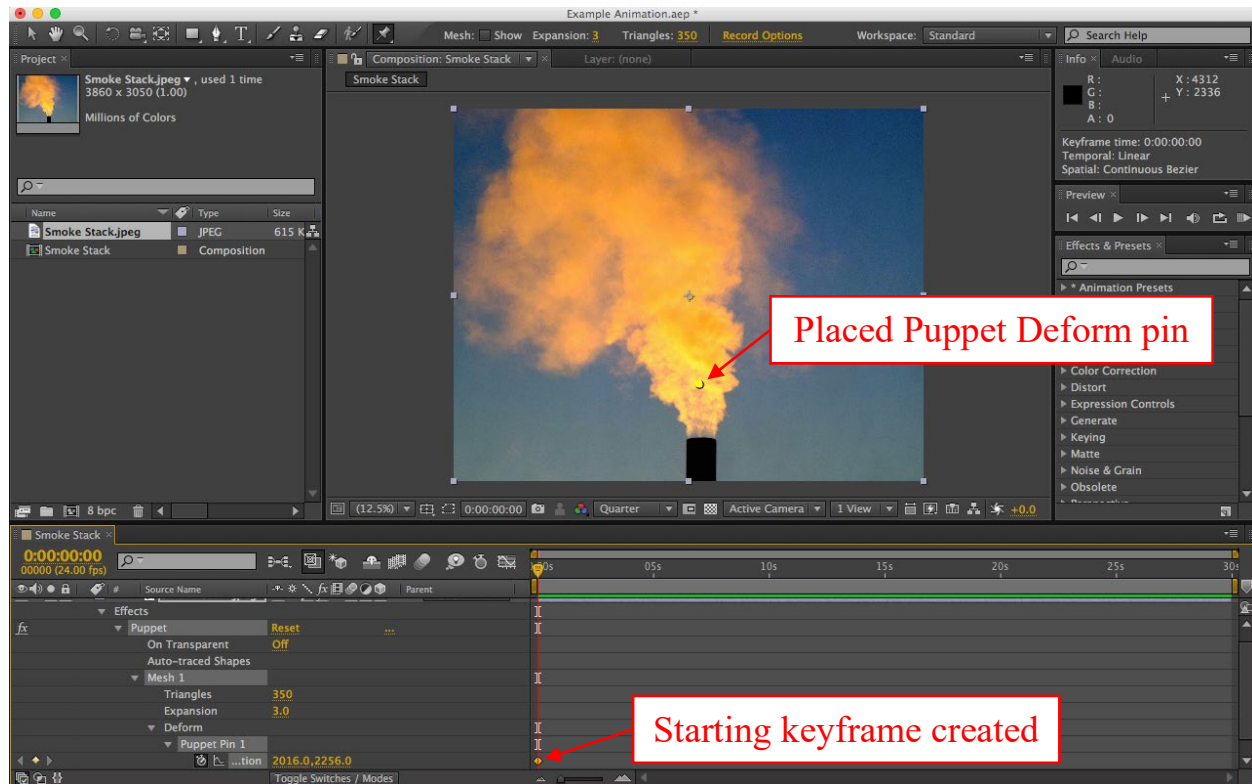
⁸ For purposes of clarity, the original video is deleted from the composition and Project panel as unnecessary. Greenspun, ¶109 n.13.



Id.

Further, AEM teaches creating a Puppet effect animation of the frame 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 animation time and the Deform pin’s position, thus disclosing the remaining elements of [1b]. AEM, 218-19; Greenspun, ¶110; *see also* ’641 Patent, 6:38-40, 6:43-45.

To further illustrate, the annotated screenshot below shows a Deform pin placed in the “Composition panel” on the extracted frame of the previous screenshots, and a starting keyframe automatically created as a result:



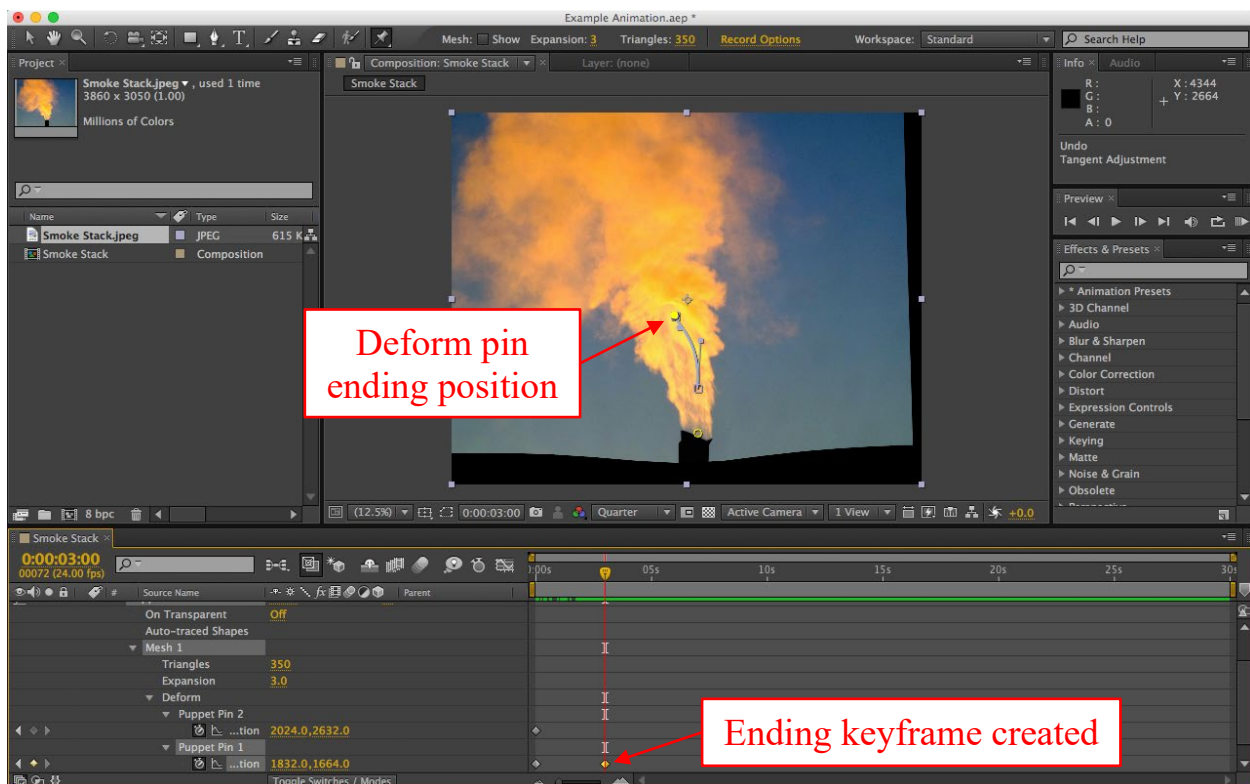
Greenspun, ¶111.

Further, should Plotagraph assert that AEM discloses creating a Puppet effect animation of only an image and not specifically the video frame extracted according to the above, a POSITA would have nevertheless been motivated to use such an extracted frame as the digital image to be animated because such a frame would have been easily accessible and, in common instances, particularly desirable for modifying with AECS6's effects, e.g., the Puppet effect. Greenspun, ¶112; *see also*, e.g., '641 Patent, 1:36-40 ("The increased ease with which video and images can be captured has led to an explosion in the amount of shared multimedia content."); AEM, 590 (stating that "export[ing] a single frame from a composition" is "useful" for, e.g., "exporting an image from a movie for posters or storyboards").

d. [1c]

Once a starting keyframe for a Deform pin has been defined, AEM instructs 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,” which discloses “receive a first ending point....” AEM, 218-19; Greenspun, ¶113; *see also* ’641 Patent, 6:48-51.

To further illustrate, the below annotated screenshot depicts creating an ending keyframe for the Deform pin previously discussed and depicted in the fifth screenshot in Section VI.A.2.c:



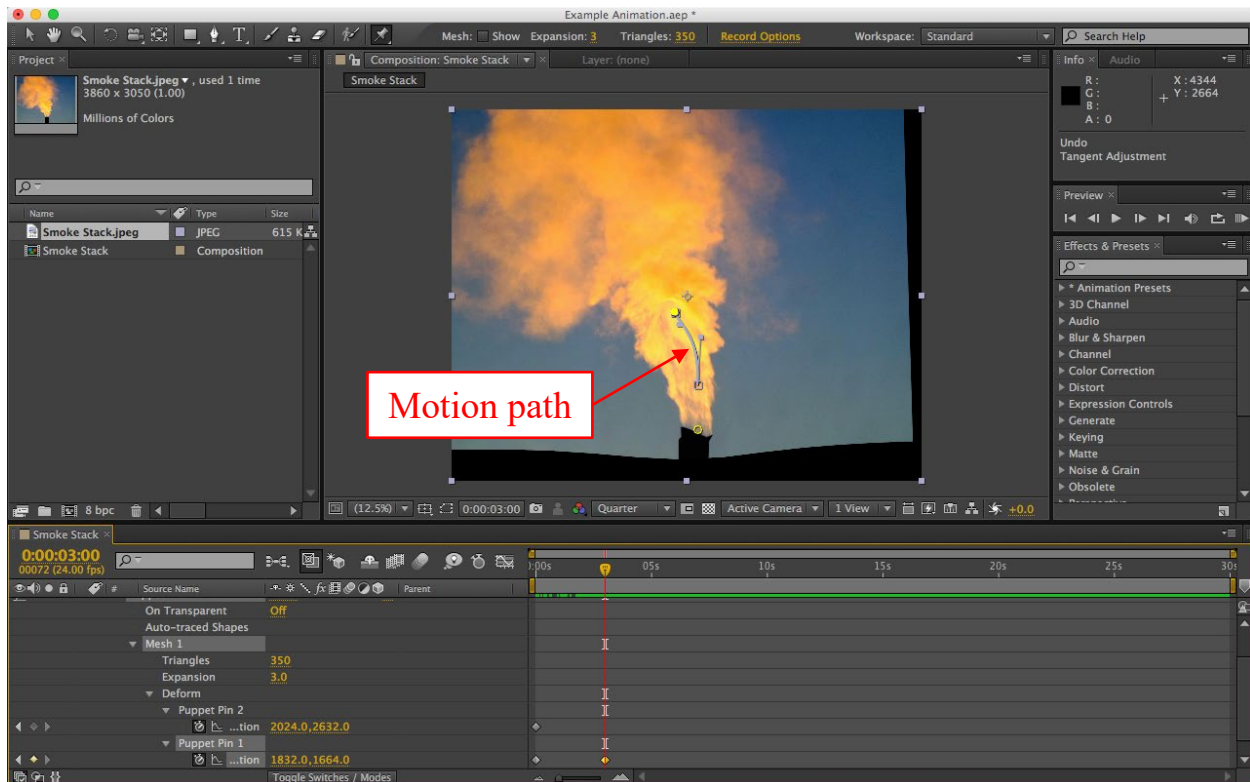
Greenspun, ¶114.

e. [1d]

AEM teaches that, when a Deform pin is initially placed and a starting keyframe is created, AECS6 automatically outlines the layer's nontransparent pixels and divides this outline into a "mesh of triangles." AEM, 219, 220-21; Greenspun, ¶115. 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, ¶115.

Further, the ending keyframe's creation prompts AECS6 to perform "keyframe interpolation," where AECS6 interpolates the Deform pin's position from the starting to 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, ¶116. The interpolation is performed according to, and visually indicated by, a "motion path," which extends from the starting to ending position/keyframe and thus discloses the claimed "create a first digital link." AEM, 187, 218-19; Greenspun, ¶116.

To further illustrate, the screenshot in Section VI.A.2.d is reproduced below, but with further annotations highlighting the motion path created between the Deform pin's starting and ending position/keyframe:



Greenspun, ¶117.

f. [1e]

Regarding “identify a first set of pixels,” the ’641 Patent states that “[t]he size of the group of pixels may be user selectable or automatically determined.” ’641 Patent, 7:28-29; Greenspun, ¶118. For example, a user can perform the “identify[ing]” of a “first set of pixels” by identifying certain pixels between the starting and ending points to be “covered by a mask” and excluded from the set of pixels to be shifted. ’641 Patent, 7:25-28, 4:24-36; Greenspun, ¶118. Alternatively, the software can perform the “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.” ’641 Patent, 7:22-28; Greenspun, ¶118.

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, ¶119.

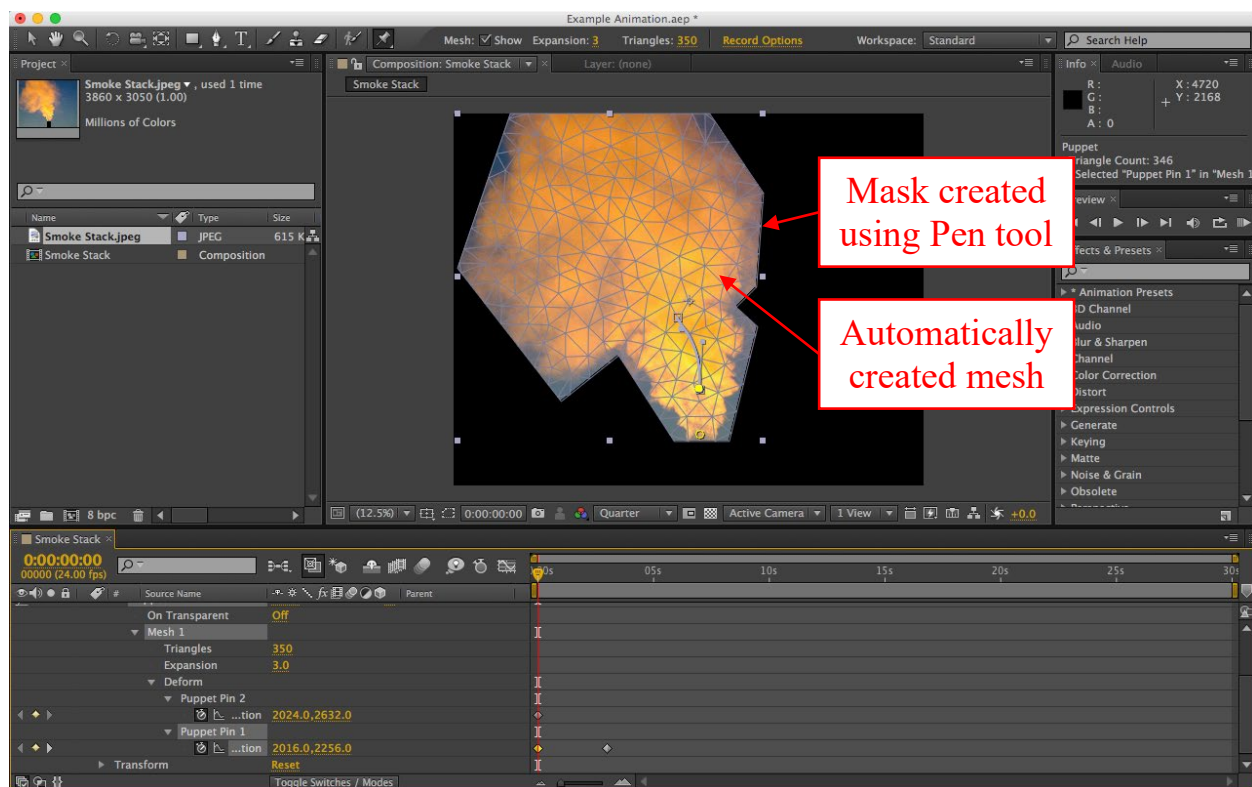
Thus, as AEM teaches, AECS6 allows a user to apply either a mask or matte to select any or all pixels of the layer to be nontransparent and thus included in the layer’s mesh to be moved and animated, including pixels that lie along the Deform pin’s motion path between the Deform pin’s starting and ending positions/keyframes, disclosing [1e]. AEM, 218-19, 315, 194-95; ’641 Patent, 7:28-29 (“The size of the group of pixels may be user selectable...”); Greenspun, ¶120. Further, AECS6 automatically creates the mesh from the layer’s nontransparent pixels, which additionally discloses [1e]. AEM, 218-19, 194-95; ’641 Patent, 7:28-29 (“The size of the group of pixels may be... automatically determined.”); Greenspun, ¶120.

Separately, a POSITA would have found obvious selecting and making nontransparent, using a mask or matte, those pixels that lie along the Deform pin’s motion path between the Deform pin’s starting and ending positions/keyframes,

IPR2023-00568

U.S. Patent No. 11,182,641

because such pixels would in common instances be pixels that the user desires to include in the mesh to be moved and animated. Greenspun, ¶121. For instance, to move and animate the smoke rising from the smokestack of the video frame 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:

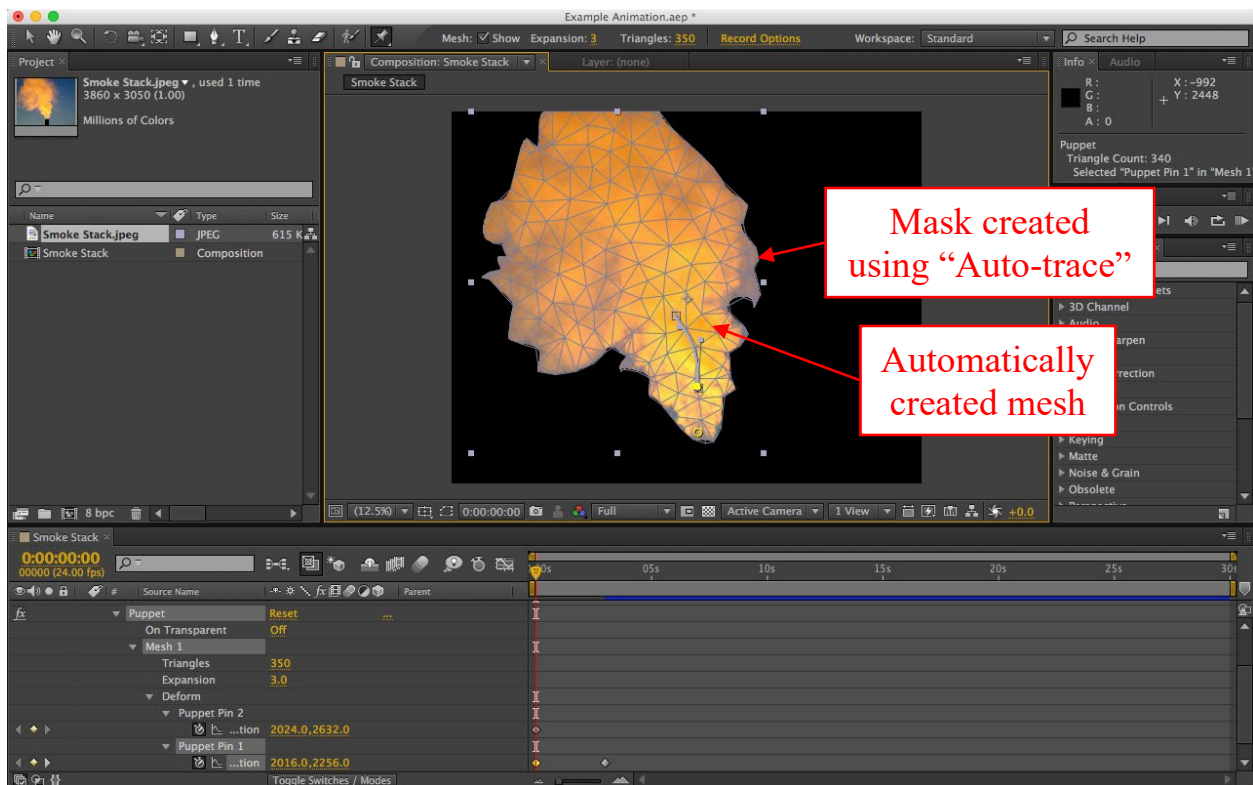


Id.; cf. Okabe, 7, 2 (depicting a similar instance in Figure 1, which depicts user-drawn “strokes” going downwards through a waterfall, and a “matte” specifying a to-be-animated “region of interest” (in white) that includes all pixels of the waterfall,

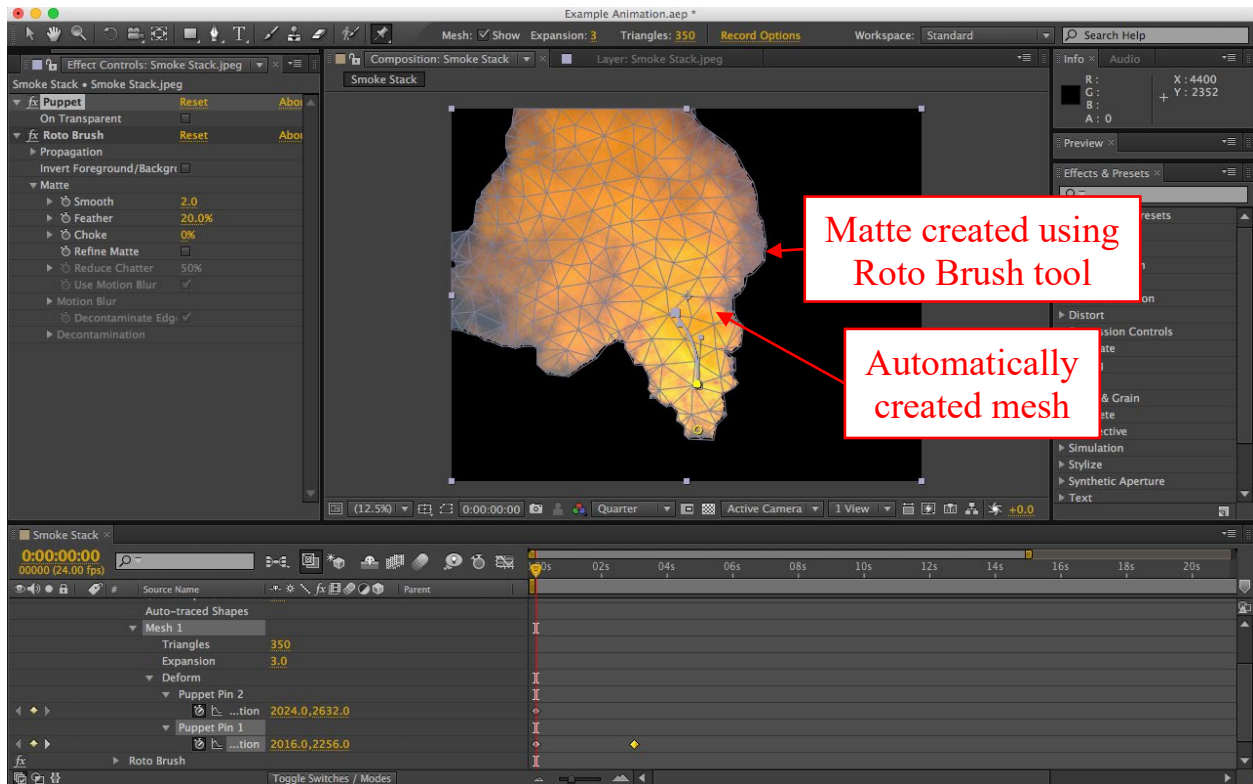
IPR2023-00568

U.S. Patent No. 11,182,641

including that lie along each stroke between each stroke's starting and ending points); Section VI.B.4.f; Greenspun, ¶121 n.14. As shown, the mask makes nontransparent, and thus the mesh includes, pixels that lie along the Deform pin's motion path between the Deform pin's starting and ending positions/keyframes. Greenspun, ¶121. 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.

g. [1f]

The '641 Patent states the claimed pixel “shift[ing]” can use a “mesh algorithm” that “triangulates the mesh using defined points and then calculates an affine transformation for every angle.” '641 Patent, 9:29-32; Greenspun, ¶122. The '641 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”:

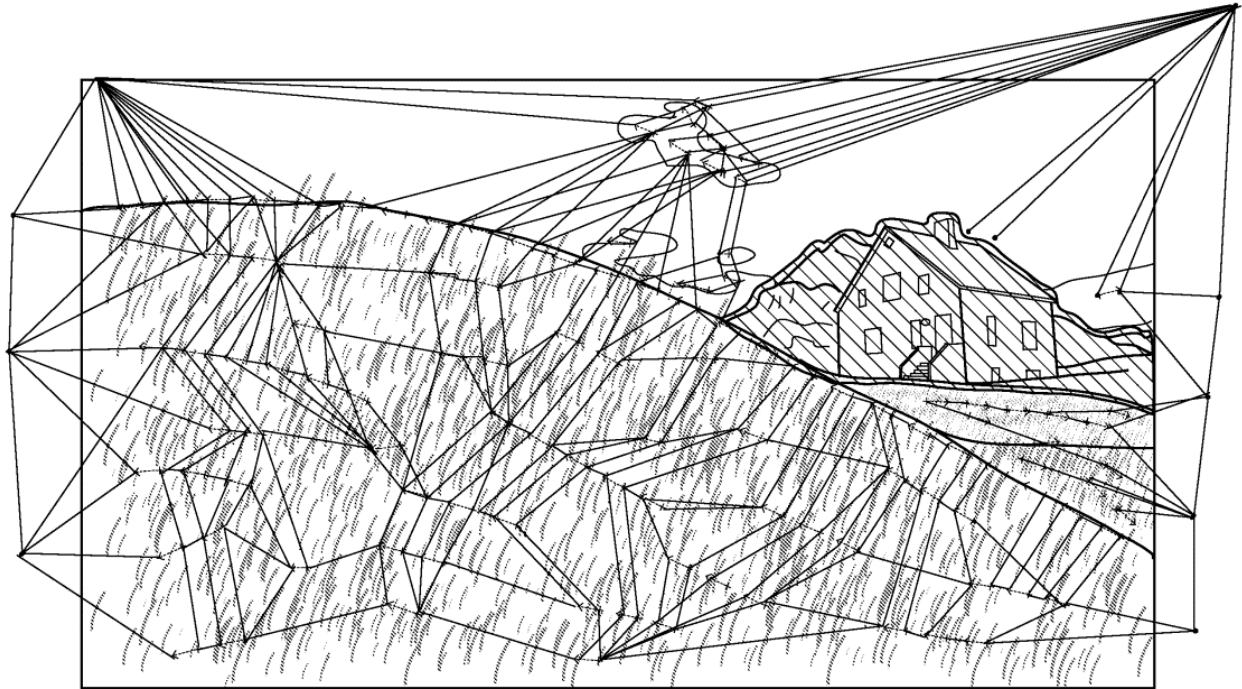
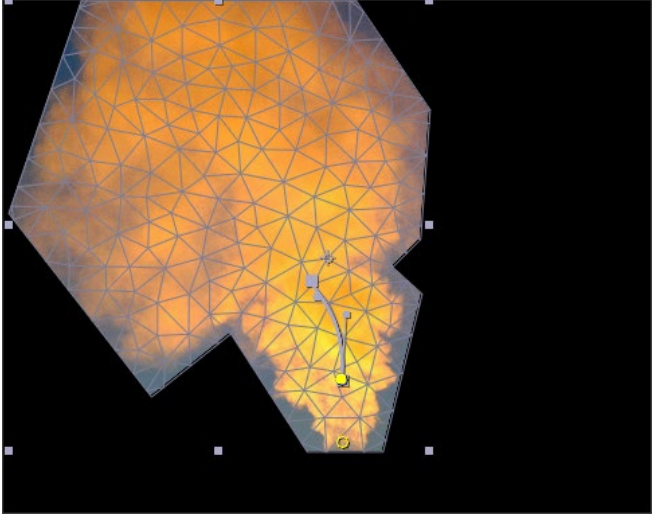
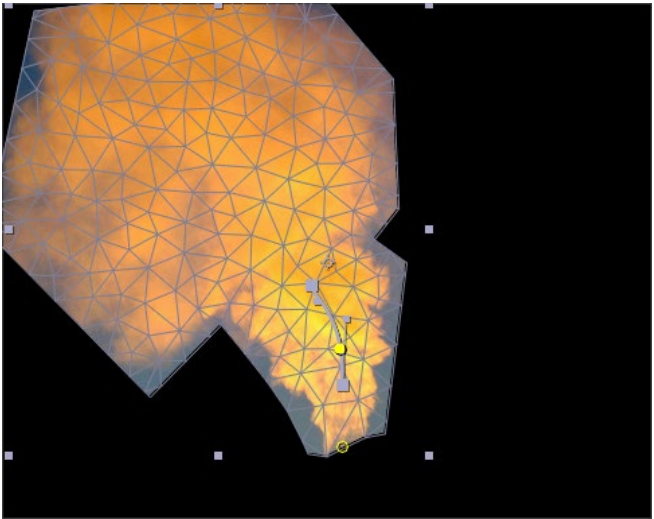


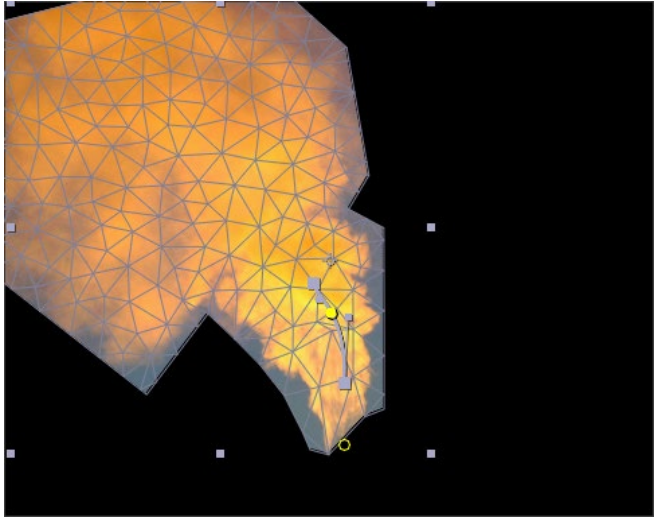
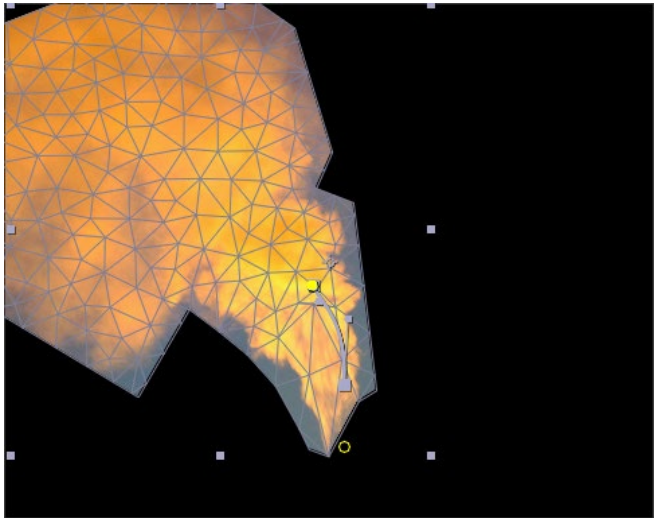
Fig. 4

'641 Patent, Fig. 4, 7:11-21; Greenspun, ¶122.

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, ¶123. And when a motion path for the Deform pin is also created, AECS6 animates the corresponding change to the shape of the layer's mesh (which includes the "first set of pixels," *see* Section VI.A.2.f), and thus the movement of the layer's nontransparent pixels accommodating the Deform pin's repositioning, disclosing [1f]. AEM, 218-19, 194-95; '641 Patent, 9:29-32; Section VI.A.2.e; Greenspun, ¶123.

To further illustrate, the following table includes a series of cropped screenshots of AECS6 showing the exemplary composition in Section VI.A.2.f's first screenshot at sequential time points to mimic the animation playing in time:

Animation Time Point	Composition Frame
(0:00:00:00)	
(0:00:01:00)	

Animation Time Point	Composition Frame
(0:00:02:00)	
(0:00:02:23)	

Greenspun, ¶124, Attachment B.

3. Claim 2

See Sections VI.A.2.c-VI.A.2.d. The ending position/keyframe for the Deform pin is created on the same video frame as that which includes the starting position/keyframe, disclosing claim 2. Greenspun, ¶125.

4. Claim 3

See Section VI.A.2.b. The discussed video footage items are “video file[s]” comprising “video frame[s]” as claimed. Greenspun, ¶126.

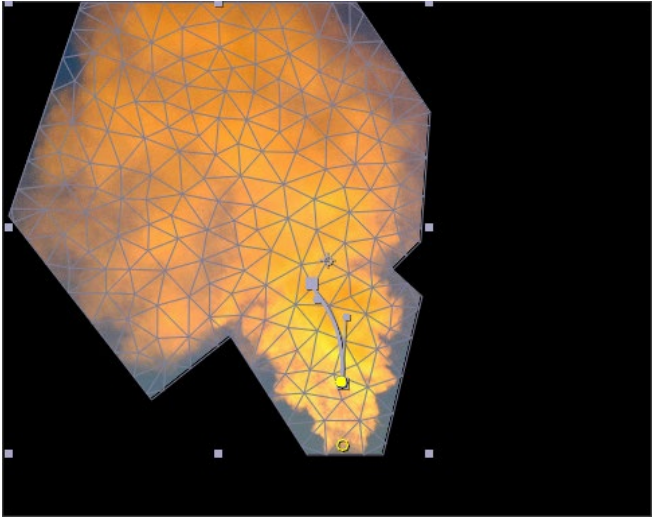
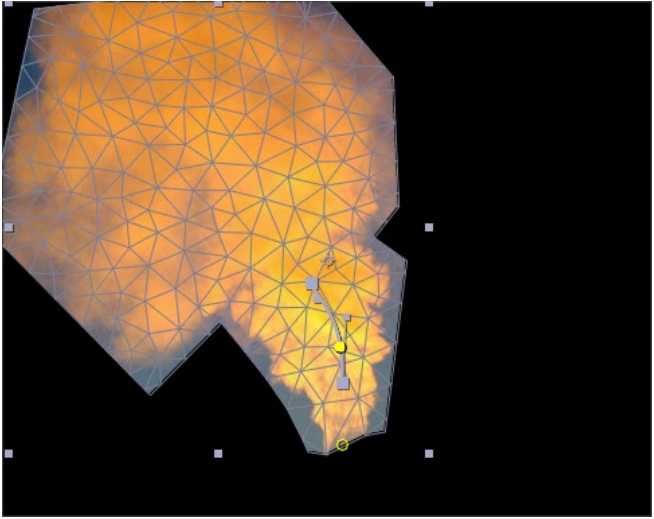
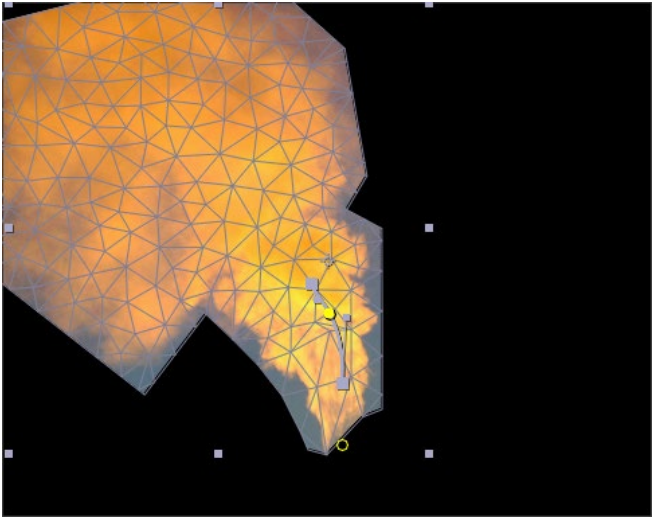
5. Claim 4

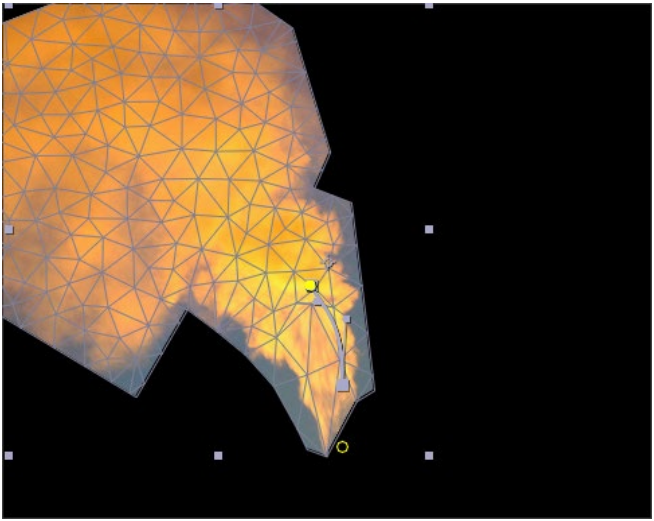
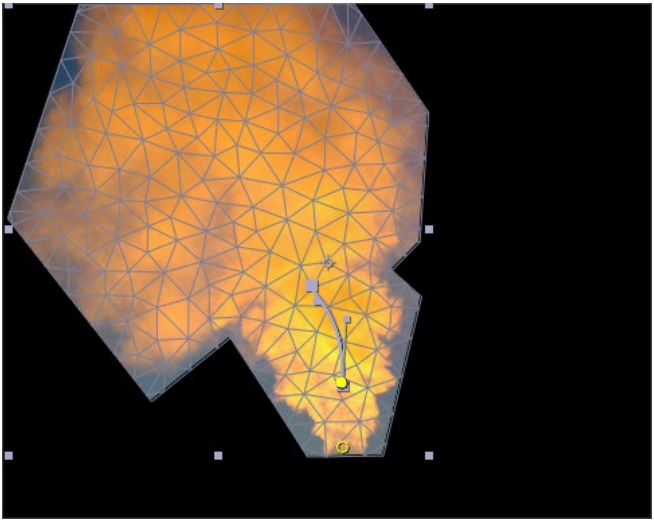
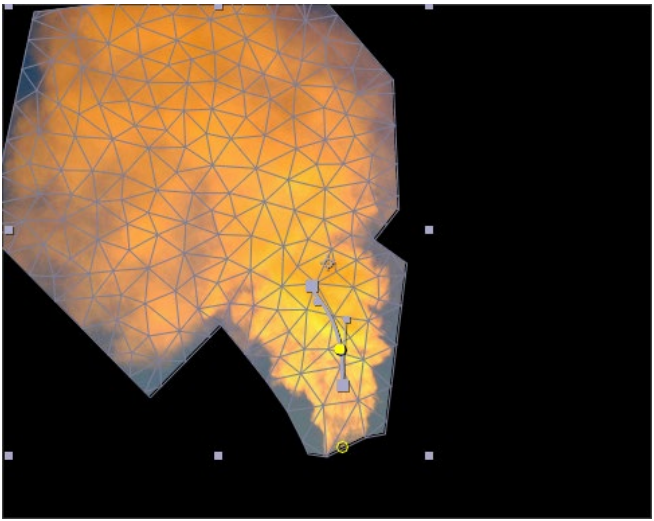
See Sections VI.A.2.c-VI.A.2.d, VI.A.3. While the starting and ending positions/keyframes for the Deform pin are created on the same video frame, such discloses claim 4 because the “first” and “second video frame[s]” may be the same frame. ’641 Patent, 6:51-54; EX1022, 137; Greenspun, ¶127.

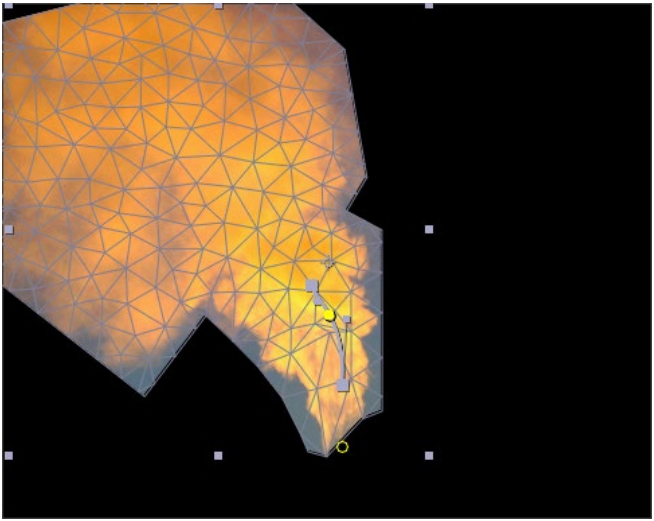
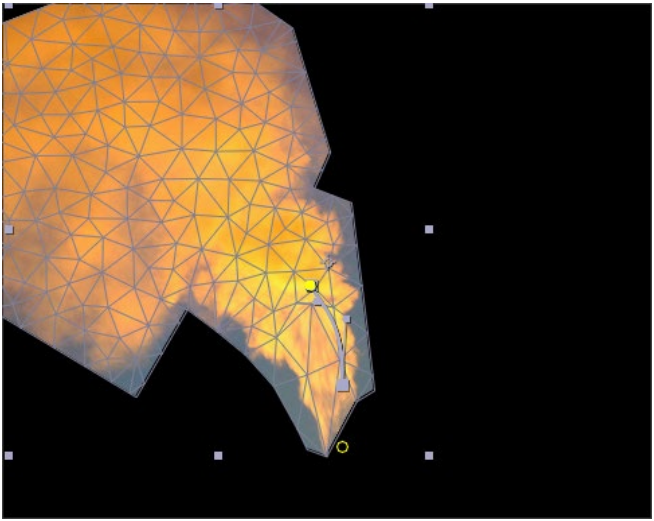
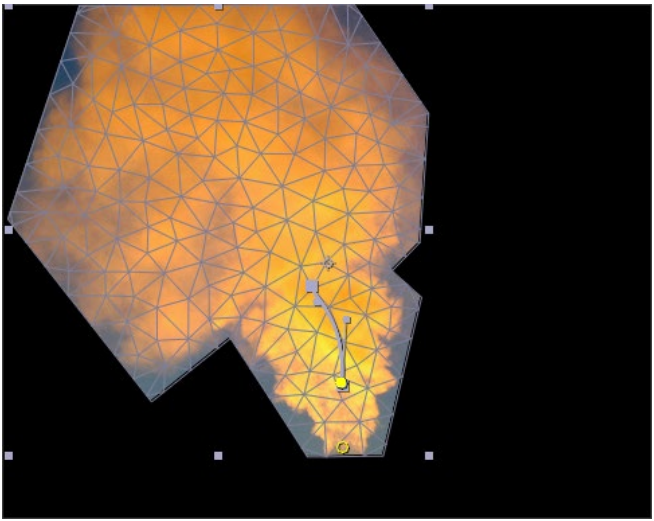
6. Claim 8

AEM teaches applying a “loopOut()” expression to a Deform pin, which, by default, causes “all keyframes [to] loop.” AEM, 562, 219; Greenspun, ¶128. Specifically, by applying a default “loopOut()” expression to a Deform pin, such as the initially placed Deform pin discussed in Sections VI.A.2.c-VI.A.2.d, 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 until otherwise specified, thus disclosing claim 8. AEM, 562, 219; Greenspun, ¶128.

To further illustrate, the table below provides a series of cropped screenshots of AECS6 like those in Section VI.A.2.g, but where a “loopOut()” expression has been applied to each Deform pin:

Animation Time Point	Composition Frame
(0:00:00:00)	 A 3D visualization of a complex, orange, translucent mesh structure against a black background. The mesh is composed of many small triangles. A yellow line with a yellow dot at its end is visible within the structure. Several small white squares are positioned around the mesh.
(0:00:01:00)	 A 3D visualization of the same orange mesh structure as in the first frame. The yellow line and dot are in a slightly different position, indicating a change in the animation. The white squares remain in the same positions.
(0:00:02:00)	 A 3D visualization of the orange mesh structure. The yellow line and dot have moved further along their path, showing a progression in the animation. The white squares are still present in the same locations.

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 the end is positioned on the right side of the map. A small yellow circle is located below the map. The background is black.
(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 the end is positioned on the right side of the map. A small yellow circle is located below the map. The background is black.
(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 the end is positioned on the right side of the map. A small yellow circle is located below the map. The background is black.

Animation Time Point	Composition Frame
(0:00:05:00)	 A composition frame showing a stylized, orange, triangular mesh map of a geographical area on a black background. A yellow line with a dot at its end indicates a path or location within the map. Several small white squares are visible around the map.
(0:00:05:23)	 A composition frame showing the same stylized, orange, triangular mesh map. The yellow line and dot have moved slightly from their position in the previous frame, indicating a change in the path or location over time.
(0:00:06:01)	 A composition frame showing the same stylized, orange, triangular mesh map. The yellow line and dot have moved further from their position in the previous frame, indicating a continued change in the path or location over time.

7. Claim 9

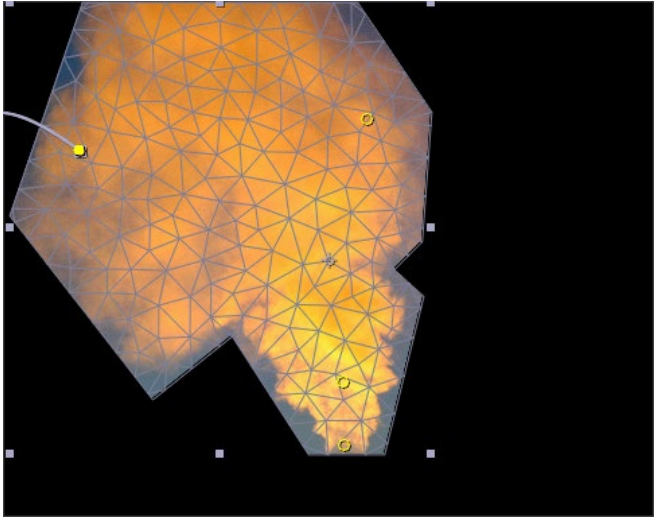
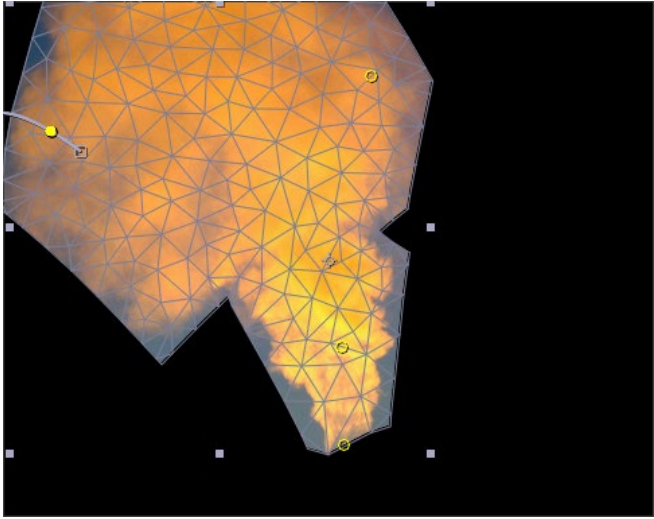
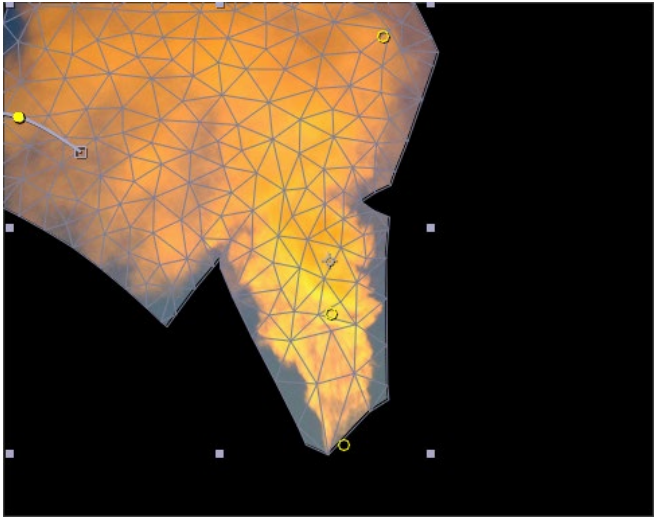
a. [9a]-[9e]

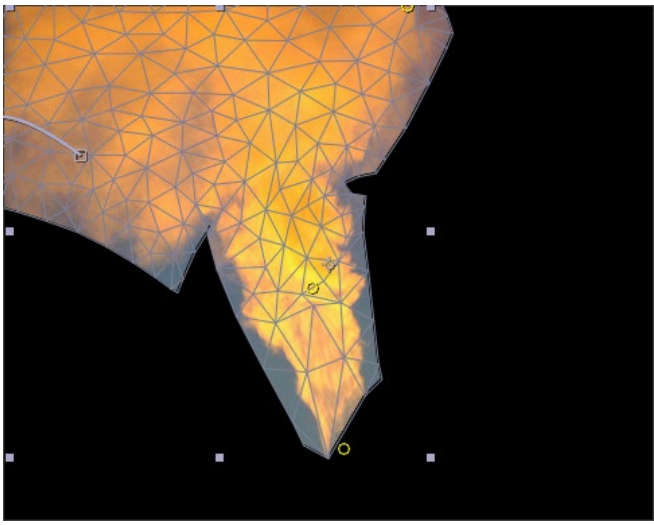
AEM teaches how to move and animate different parts of a layer through the use of multiple Deform pins. AEM, 218-19; Greenspun, ¶130. Like when placing a first Deform pin (*see* Section VI.A.2.c), a user places additional Deform pins and creates for each a starting keyframe visible in the Composition and Layer panels by simply “[c]lick[ing] in one or more places within the outline,” disclosing [9a]. AEM, 219 (emphasis added); ’641 Patent, 6:38-40; Greenspun, ¶130. The user creates ending keyframes and motion paths for these additional Deform pins in the same way as for the first Deform pin, disclosing [9b]-[9c]. AEM, 218-19; ’641 Patent, 6:48-51; Greenspun, ¶130; Sections VI.A.2.d-VI.A.2.e. Further, AEM teaches, and a POSITA would have found obvious, creating a mask or matte to select any and all pixels of the layer to be nontransparent and thus included in the layer’s mesh to be moved and animated, including pixels that lie along each Deform pin’s motion path between each Deform pin’s starting and ending positions/keyframes, disclosing [9d]. AEM, 218-19, 315, 194-95; Section VI.A.2.f; Greenspun, ¶130; *cf.* Okabe, 7, 2 (depicting, in Figure 1, not 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 that lie along the strokes between each

stroke's starting and ending points); Section VI.B.9.a; Greenspun, ¶130 n.15.

Finally, 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, disclosing [9e]. AEM, 218-19, 194-95, 562; Section VI.A.2.g; Greenspun, ¶130.

To further illustrate, 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 additional Deform pins:

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. Several small yellow circles are visible on the shape, and a white line segment is on the left side.
(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. Several small yellow circles are visible on the shape, and a white line segment is on the left side.
(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. Several small yellow circles are visible on the shape, and a white line segment is on the left side.

Animation Time Point	Composition Frame
(0:00:02:23)	

Greenspun, ¶131; Attachment C.

8. Claim 10

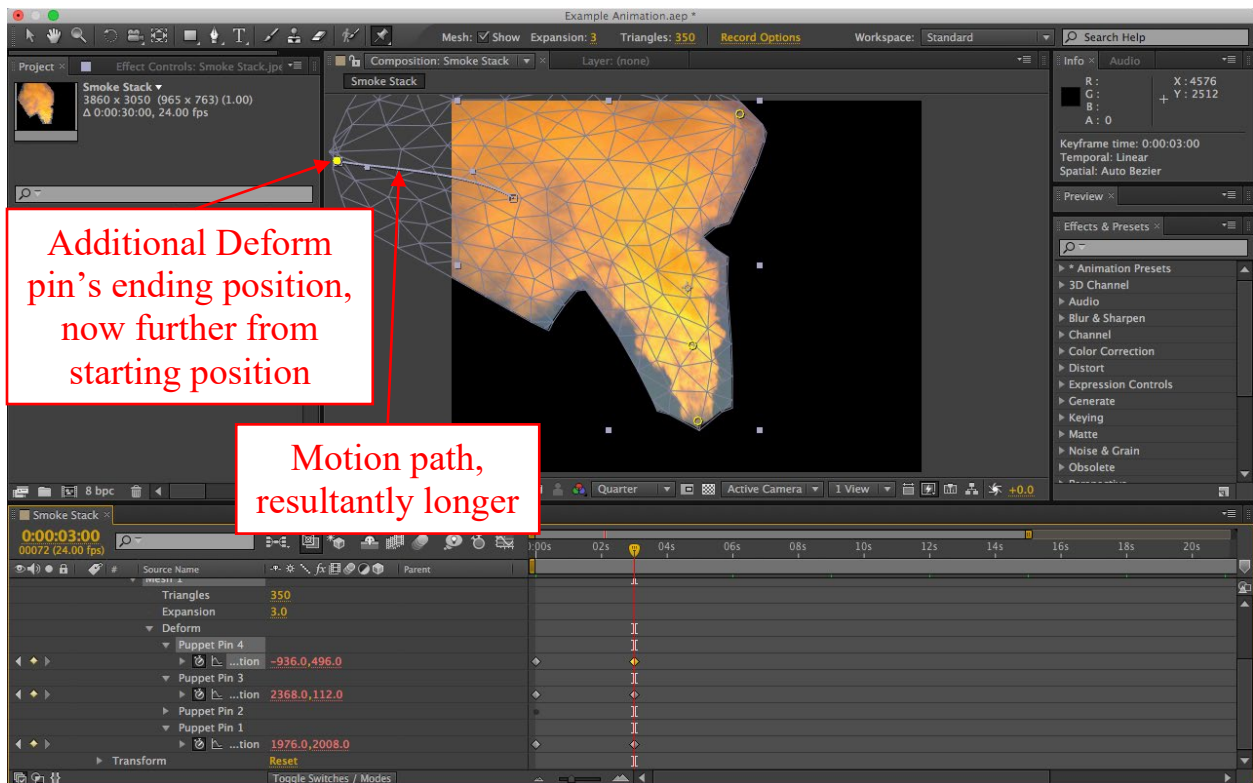
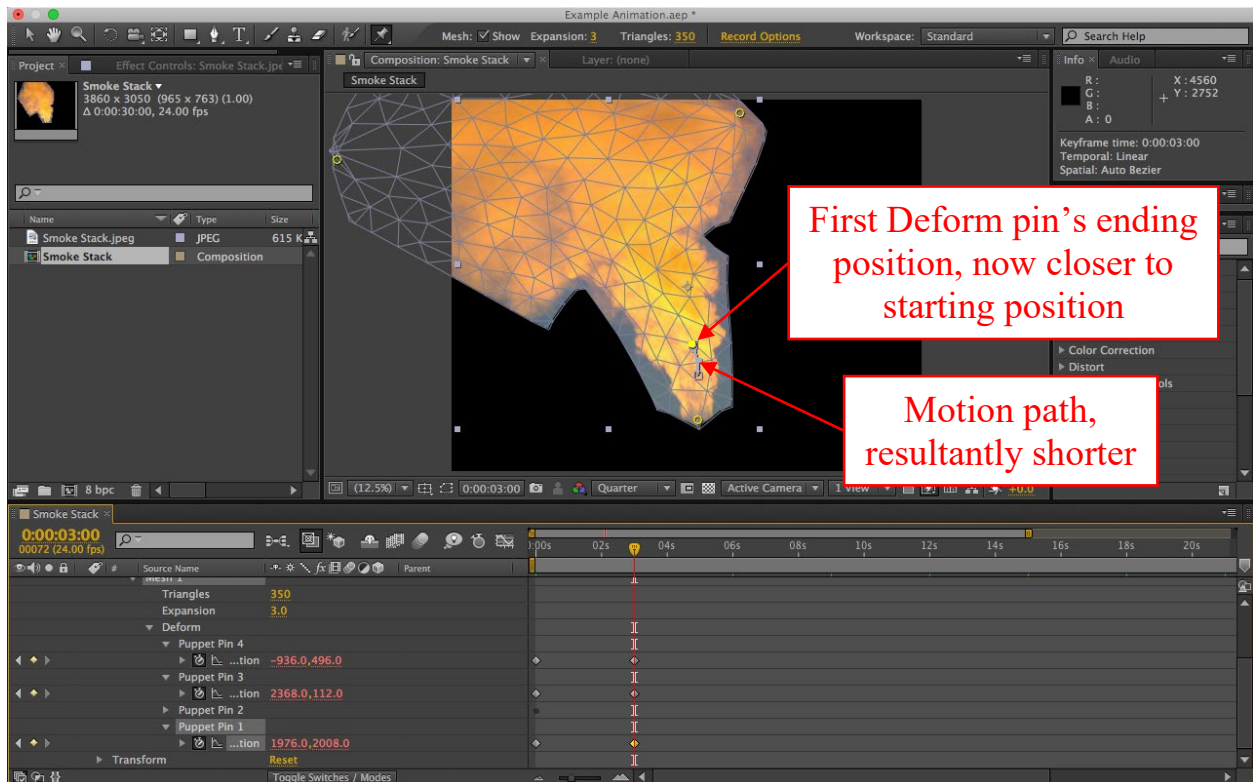
AEM teaches creating different motion paths for each Deform pin placed on a layer, disclosing claim 10. AEM, 218-19; Section VI.A.7.a; Greenspun, ¶132. Indeed, the example in Section VI.A.7.a’s table shows each Deform pin having a separate motion path that differs from the others. Greenspun, ¶132.

9. Claim 11

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.c-VI.A.2.d; Greenspun, ¶133. 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, ¶134. 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 to ending position/keyframe. AEM, 187, 218-19; Section VI.A.2.e; Greenspun, ¶134. As such, for a given span of time in the composition, if a Deform pin’s ending position/keyframe is repositioned a relatively long distance away from its starting position/keyframe, the corresponding change to the shape of the layer’s mesh—and the movement of the layer’s nontransparent pixels—will be large 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, ¶134. The opposite occurs if the Deform pin repositioned a relatively short distance. *See* AEM, 218-19; Greenspun, ¶134. Such illustrates a proportional relationship between the magnitude of pixel movement and the length of each Deform pin’s motion path, disclosing claim 11. Greenspun, ¶134.

To further illustrate, the exemplary composition from Section VI.A.7.a’s table is depicted in the screenshots below, but now the first Deform pin is repositioned a lesser amount (first screenshot) while one additional Deform pin is repositioned a larger amount (second screenshot) for a fixed span of time:



Greenspun, ¶135. 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, ¶135.

10. Independent Claim 12

a. [12pre]

To the extent [12pre] is limiting, AEM instructs to “[m]ake sure that you’ve installed the current version of [AECS6], including any available updates.” AEM, 517; Greenspun, ¶136. To view such updates, AEM instructs to “go to the Downloads section of the Adobe website,” and provides a hyperlink for doing so. AEM, 517; Greenspun, ¶136. A user following these instructions would have downloaded and installed the most up-to-date version of AECS6 from Adobe’s website—and thus a server (*see* Nakagawa, Abst.) (“computer program product comprising one or more non-transitory computer storage media having stored thereon computer-executable instructions”)—onto the user’s computer (“when transmitted to a remote computer system for execution at a processor, cause the remote computer system to perform”). Greenspun, ¶136.

Further, describes using AECS6 to animate the movement of pixels within a single frame of a video (“method for automating a shifting of pixels within an image file”). Sections VI.A.10.b-VI.A.10.f; Greenspun, ¶136.

b. [12a]

See Sections VI.A.2.b-VI.A.2.c. Receiving the Deform pin’s starting position discloses “receiving a first indication of a first starting point through a user interface.” Greenspun, ¶137.

c. [12b]

See Sections VI.A.2.d-VI.A.2.e. The discussed motion path—which results when the user creates a starting and ending keyframe for the Deform pin via the Composition or Layer panel, and which extends from the starting to ending position/keyframe—discloses [12b]. AEM, 187, 218-19; Greenspun, ¶138.

d. [12c]

See Section VI.A.2.e. The created motion path, as discussed, extends from the starting to ending position/keyframe, disclosing [12c]. AEM, 187, 218-19; Greenspun, ¶139.

e. [12d]

For similar reasons discussed in Section VI.A.2.f, AEM teaches, and a POSITA would have found obvious, applying a mask or matte in AECS to select any or all pixels of a layer to be nontransparent and thus included in the layer’s mesh to be moved and animated, including pixels that are along a Deform pin’s motion path and extend in the motion path’s direction away from the Deform pin’s starting position/keyframe, thus disclosing [12d]. Greenspun, ¶140; *see also* Section VI.A.10.d.

f. [12e]

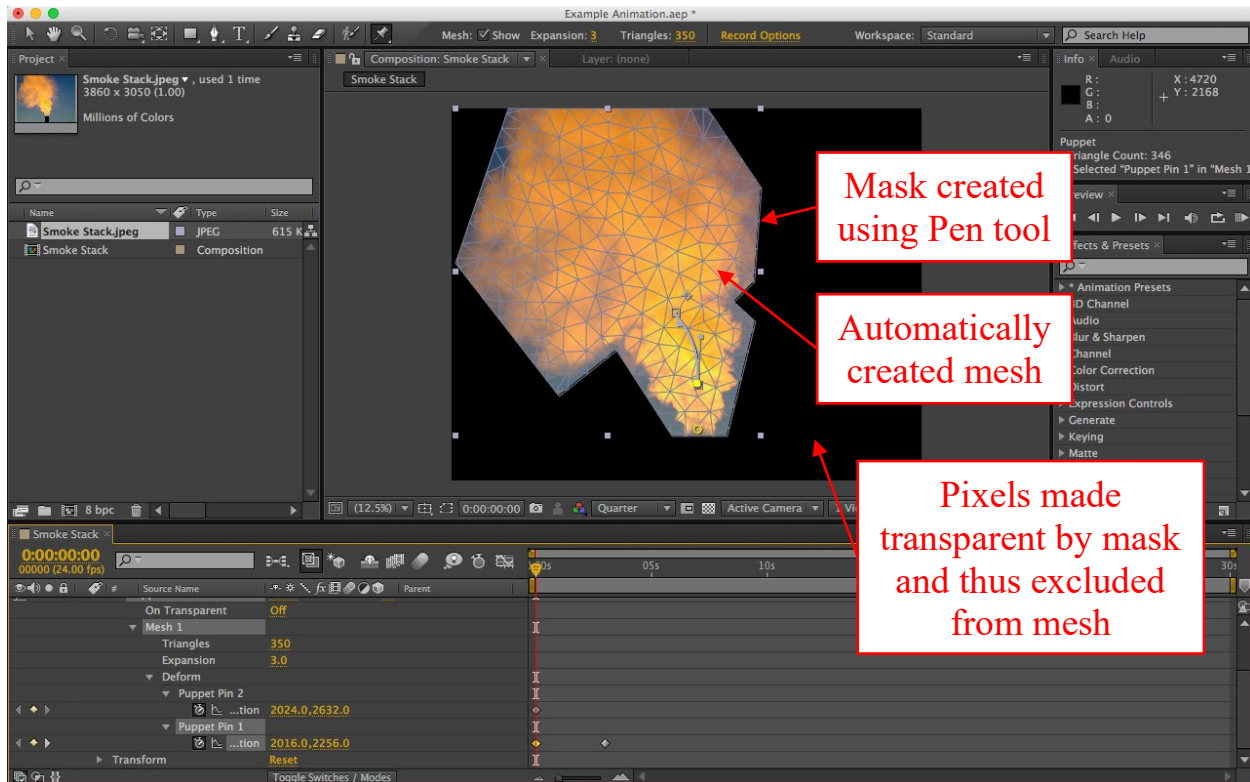
See Section VI.A.2.g.

11. Claim 13**a. [13a]-[13b]**

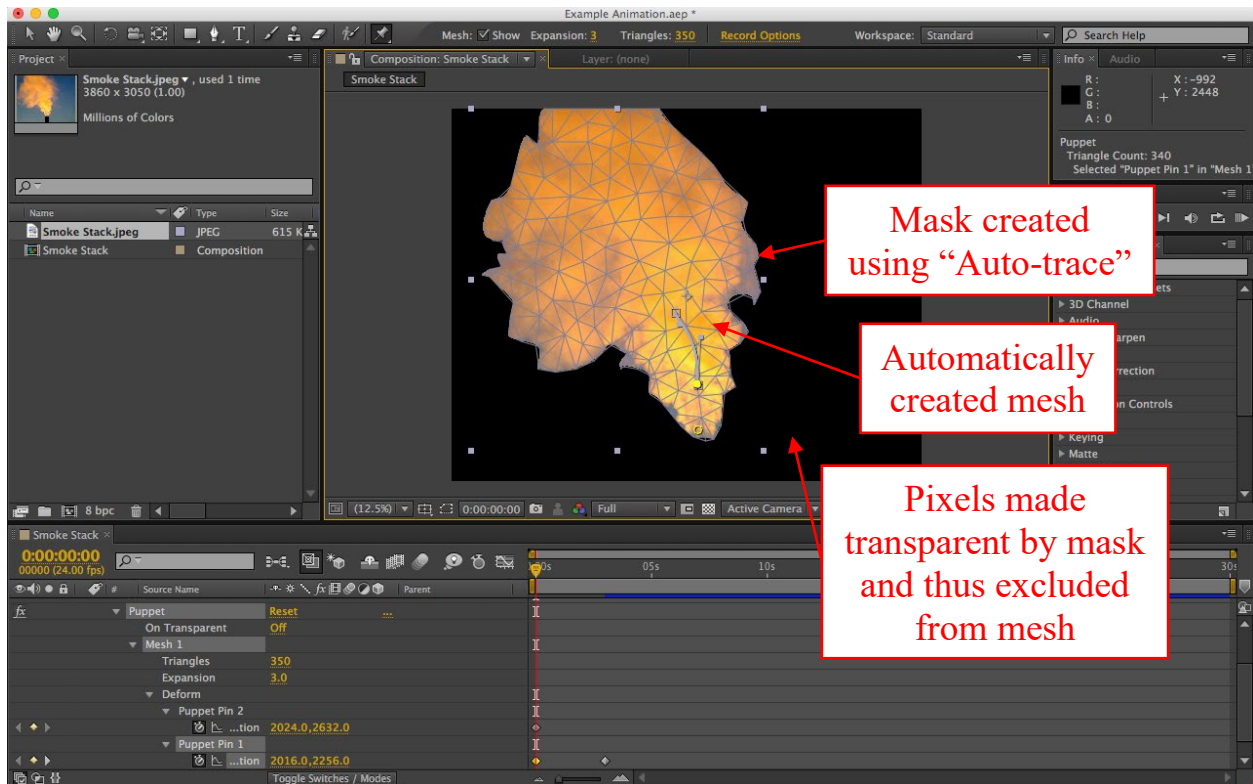
AEM teaches using a mask or matte to select pixels to be nontransparent and thus included in the layer's mesh to be moved and animated by the Puppet effect. AEM, 315, 218-19; Section VI.A.2.f; Greenspun, ¶142. Naturally, such a mask or matte is therefore also used to select one or more pixels to be transparent and thus *excluded* from the layer's mesh so as to *not* be moved and animated, and thus discloses “a first mask over a second portion of the first image frame, wherein pixels under the first mask are prevented from shifting, wherein pixels under the first mask are prevented from shifting.” Greenspun, ¶142; *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 “receiving an indication to generate a first mask.” AEM, 262, 264-65; Greenspun, ¶143. 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 “receiving....” AEM, 328; Greenspun, ¶143.

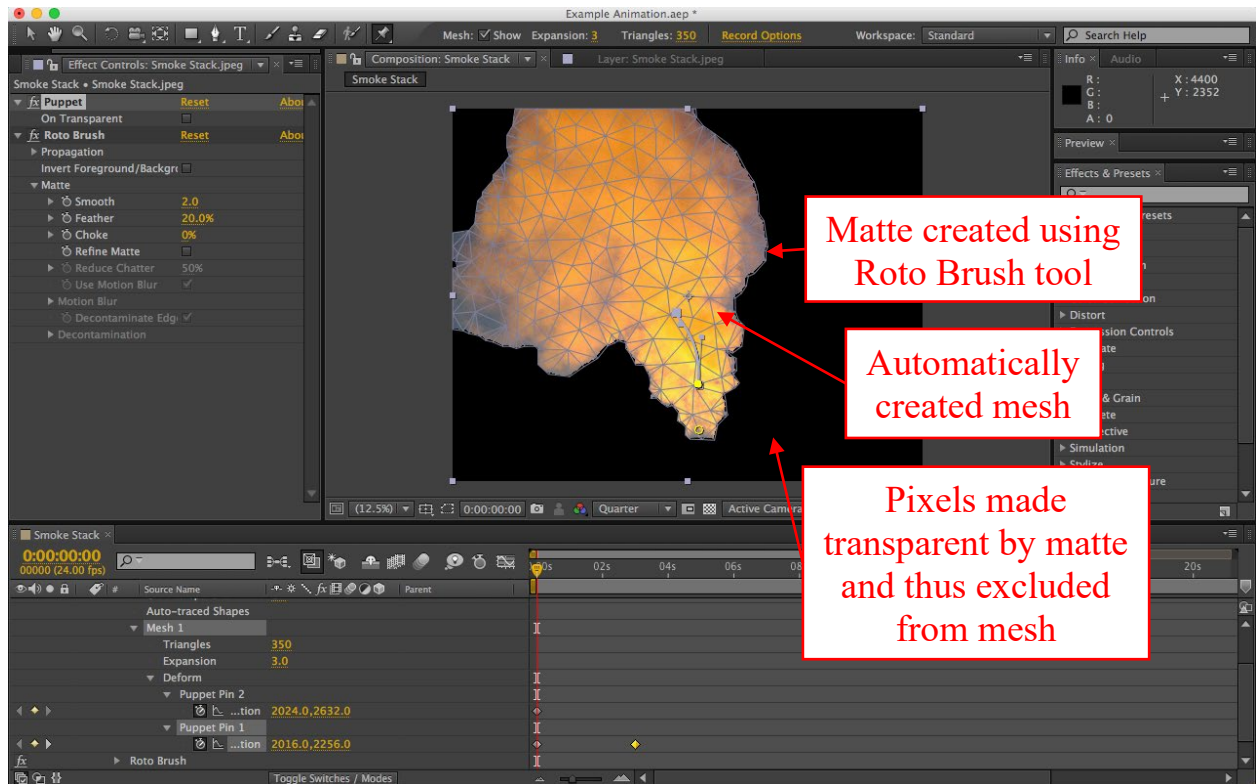
Section VI.A.2.f's first screenshot depicting a mask created using the "Pen" tool is shown below, but the annotations now indicate pixels of the layer made transparent by the mask and thus excluded from the mesh:



Greenspun, ¶144. 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.

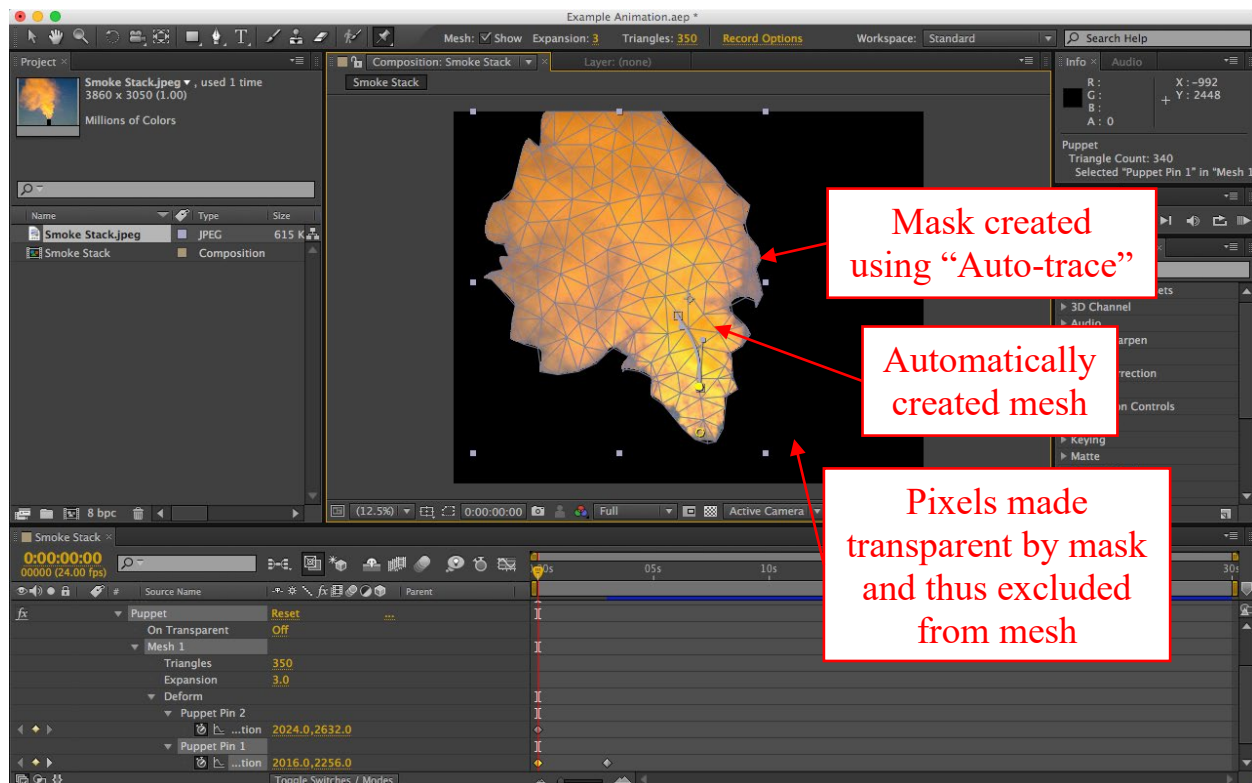
12. Claim 14

As discussed in Section VI.A.11.a, 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 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, any of which discloses claim 14. AEM, 318, 262, 264-65, 328; Greenspun, ¶145.

13. Claim 15

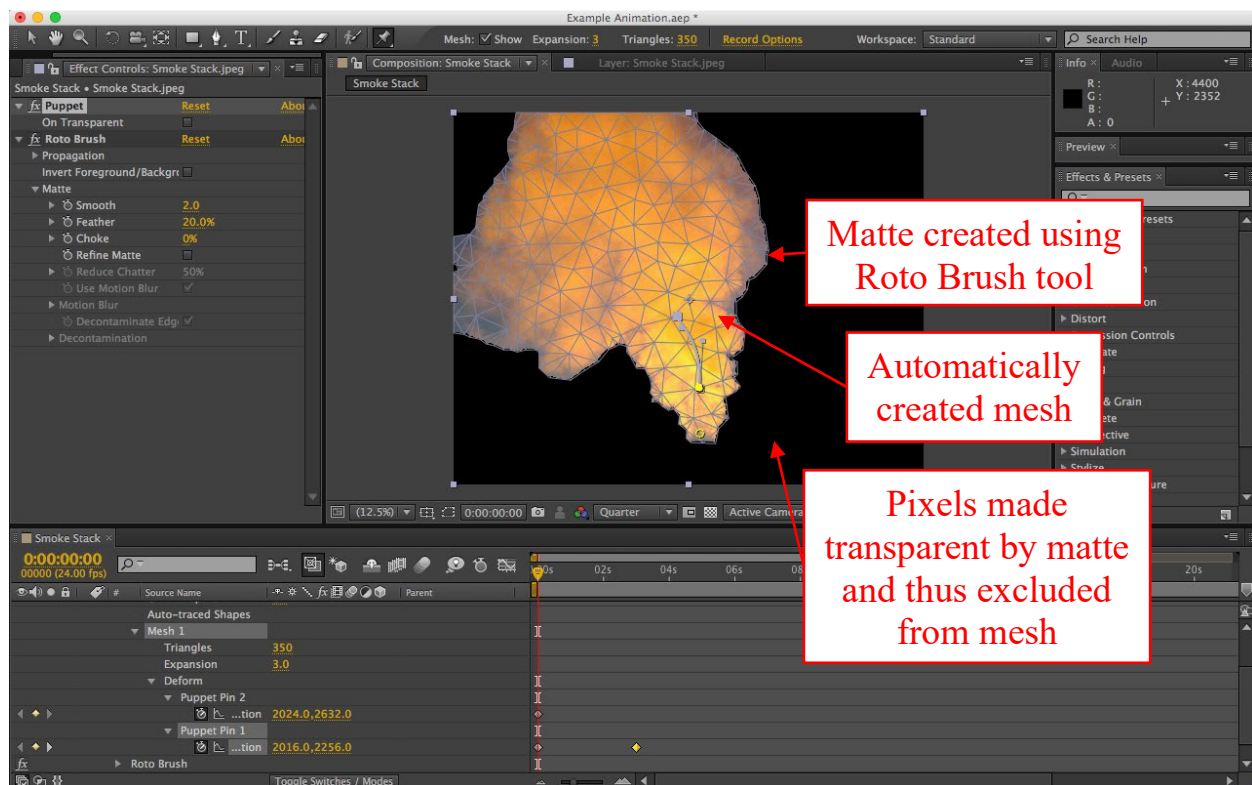
a. [15a]-[15b]

AEM teaches that a mask is automatically created for a layer when using AECS6's "Auto-trace" function. AEM, 262; Greenspun, ¶146. 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" and generates a mask that outlines such edges, thus disclosing [15a]-[15b]. AEM, 262; Greenspun, ¶146; *see also* AEM, 104, 318, 328. Indeed, Section VI.A.11.a's second screenshot shows that the mask traces the luminance edge between the smoke and sky, where only the smoke's pixels are nontransparent:



Greenspun, ¶146.

Further, AEM teaches creating a matte using the “Roto Brush” tool, which, similar to Auto-trace (*see* AEM, 262), uses “Edge Detection” to detect a “segmentation boundary” separating a layer’s foreground and background elements based on user-drawn “strokes” over “representative areas of the foreground and background elements.” AEM, 328-31; Greenspun, ¶147. This additionally discloses “identifying one or more edges....” Greenspun, ¶147; *see also* ’641 Patent, 4:46-64. Further, as AEM teaches, AECS6 applies a matte that isolates the foreground from the background, thereby making the latter transparent, and thus AEM discloses “generating the first mask....” AEM, 328, 318; Greenspun, ¶147. Indeed, Section VI.A.11.a’s third screenshot depicts creating a matte using the “Roto Brush” tool:



Greenspun, ¶147 As shown, the segmentation boundary between the foreground (smoke) and background (sky, made transparent by the matte) traces the edge between the two. *Id.*

14. Independent Claim 19

a. [19pre]

To the extent [19pre] is limiting, AEM instructs to “[m]ake sure that you’ve installed the current version of [AECS6], including any available updates.” AEM, 517; Greenspun, ¶148. To view such updates, AEM instructs to “go to the Downloads section of the Adobe website,” and provides a hyperlink for doing so. AEM, 517; Greenspun, ¶148. A user following these instructions would have downloaded and installed the most up-to-date version of AECS6 onto the user’s computer (“client computing device”) from Adobe’s website—and thus a server (*see* Nakagawa, Abst.) (“transmitting to a client computing device instructions,” and “transmitting computer executable instructions to a client computing device, the computer executable instructions configured to cause the client computing device to”). Greenspun, ¶148.

Further, AEM teaches using AECS6 to animate the movement of pixels within a single frame of a video (“for shifting pixels within a video file”). Sections VI.A.14.b-VI.A.14.g; Greenspun, ¶149.

b. [19a]

See Section VI.A.2.b.

c. [19b]

See Section VI.A.2.c.

d. [19c]

See Section VI.A.2.d.

e. [19d]

See Sections VI.A.2.e.

f. [19e]

See Sections VI.A.2.f.

g. [19f]

See Sections VI.A.2.g.

15. Claim 20

See Section VI.A.4.

B. Ground 2: IMU and Okabe Render Obvious Claims 1-4, 8-14, and 19-20

1. Summary of 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, ¶157.

IMU “strongly recommend[s] to use an up-to-date version of IM[V6]” (IMU-Windows, 3; *see also* IMU-Home), and hyperlinks ImageMagick.org’s “Downloads Page” for downloading and installing such a new version of IMV6 (IMU-Home, 1).

Greenspun, ¶158. This hyperlink, when clicked, directs to the archived “Download ImageMagick” webpage shown in EX1026, from which the “latest release of I[MV6]” could indeed be downloaded and installed. EX1026; IMU-Home, 1; Greenspun, ¶158 & n.16 (testifying EX1026, which he provided, shows the webpage that would have resulted after clicking the aforementioned hyperlink around IMU-Home’s capture date).

IMU also teaches that, once installed, IMV6 is operated in the command-line. IMU-Home, 2; Greenspun, ¶159. Thus, IMV6 “*is not a GUI image editor.*” IMU-Home, 2; Greenspun, ¶159.

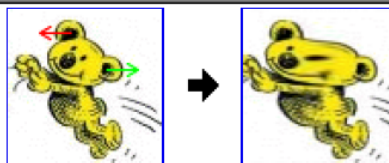
Further, IMU teaches how to manually write text commands in IMV6 to apply effects to an image, such as IMV6’s “Shepard’s Distortion” pixel shifting effect, as well as how to create infinitely looping animations from such effects. IMU also teaches how to extract individual frames from a GIF animation, which can be used to create a subsequent animation. IMU additionally teaches how to manually write commands to modify the transparency of the pixels of an image (e.g., an extracted frame of a GIF animation)—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, ¶161. One such effect is “Shepard’s Distortion,” a distortion effect that allows a user to place “control points” on pixels located at user-specified coordinates of an image and move such control points to new coordinates “to distort the image in terms of ‘local’ effects.” IMU-Distorting, 59, 19-20; Greenspun, ¶161. As an example, IMU provides an image of a koala, as well as an exemplary command that places and moves Shepard’s Distortion control points 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, ¶161. The command, as well as the 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 within the image:

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

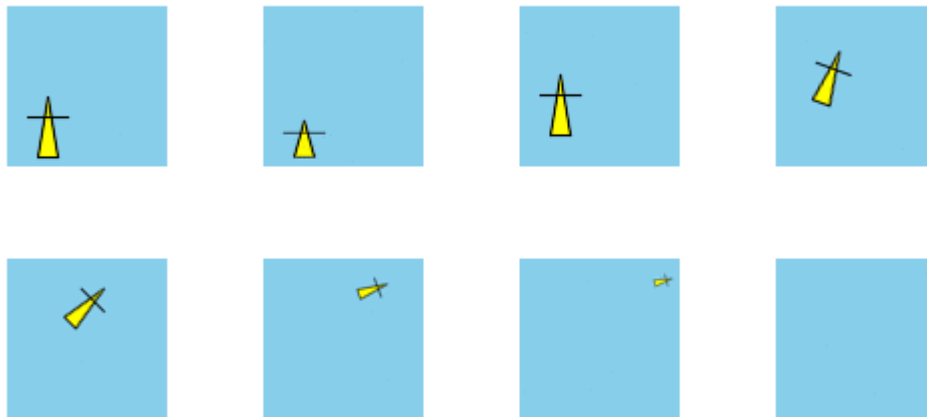


IMU-Distorting, 59; Greenspun, ¶161.

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, result in an animation. *See* IMU-Distorting, 17-18; Greenspun, ¶162. IMU provides an example command that applies 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 moving a placed SRT Distortion control point in seven increments to create frames that, when played sequentially, animate the space ship launching along a path through the sky. IMU-Distorting, 17-18; Greenspun, ¶162. The example and resulting frames 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
```



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. IMU-Animating, 1-2; IMU-Distorting, 17-18; Greenspun, ¶163. In creating such a 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, ¶163. When “-loop” is applied at its default value of zero, like in the above example, the GIF will render the animation in an “infinite loop.” IMU-Animating, 2; IMU-Distorting, 17-18; Greenspun, ¶163.

b. Extracting a Single Frame of a GIF Animation

IMU also teaches how to use IMV6 to “split[] an animation into frames”—i.e., extract and output an animation’s frames as individual images. IM-Animating, 7; Greenspun, ¶164. Particularly, IMV6’s “+adjoin” operator can be used to “read in” a “GIF animation[] and output the individual frame images in the animation sequence.” IM-Animating, 7; Greenspun, ¶164.

⁹ The full animation can be viewed at IMU-Distorting’s archived URL: <https://web.archive.org/web/20120329131929/http://www.imagemagick.org/Usage/distorts/>.

The “+adjoin” operator, however, extracts “sub-frame[s]”—certain animated GIFs are created by layering sub-frames over one another over time (*see* IMU-Animating, 16-17), and, for such GIFs, the “+adjoin” operator outputs only those sub-frames (IM-Animating, 7-8). Greenspun, ¶165. To see instead “a complete view of the animation at each point,” IMU teaches to use IMV6’s “-coalesce” operator, which works similarly to “+adjoin” but outputs each full frame image rather than sub-frames. IM-Animating, 8-9; Greenspun, ¶165.

IMU provides an example of applying “+adjoin” to an animated GIF of a script letter K being drawn to illustrate the operator’s functionality:

```
convert script_k.gif +repage +adjoin script_k_%02d.gif
```



IMU-Animating, 8; Greenspun, ¶166. IMU also provides an example of applying “-coalesce” to the same GIF to show the differences in results:

```
montage script_k.gif -coalesce \
  -tile x1 -frame 4 -geometry '+2+2' \
  -background none -bordercolor none coalesce_k_montage.gif
```



IMU-Animating, 8; Greenspun, ¶166.

Further, IMU also teaches numerous ways to use such frame images, including to “study, edit, modify and re-optimize” the original animation. IMU-

Animating, 8; Greenspun, ¶167. The user can also “use the individual frames for other projects,” such as for creating a subsequent animation. *E.g.*, IMU-Animating, 7; Greenspun, ¶167.

c. Modifying Transparency of Certain Pixels Using a Matte

IMU also teaches modifying the transparency of the pixels of an image (e.g., a frame image)—and thus the visibility of effects that are applied to the frame image’s pixels—by modifying the image’s “transparency (alpha) channel,” also referred to as a “‘matte’ channel” or simply a “matte.” IMU-Masking, 2-3; IMU-Animating, 7-8; Greenspun, ¶168. 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, ¶168.

IMU provides an example matte shaped like a crescent moon:



IMU-Masking, 2; Greenspun, ¶169. IMU further illustrates how this matte is/is not applied to correspondingly change the transparency of an image’s pixels:



IMU-Masking, 3-4; Greenspun, ¶169.

2. Summary of Okabe

Okabe describes a “method for synthesizing fluid animation from a single image.” Okabe, 1; Greenspun, ¶170. While acknowledging that “[m]any methods have been proposed for creating an animation from a single image,” Okabe asserts that its method provides, among other advantages, “markedly reduce[d]... user burden.” Okabe, 2; Greenspun, ¶170.

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, ¶171. 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 target image, “which is a simple task and takes less than 1 min[ute].” Okabe, 3, 7; Greenspun, ¶171. Okabe’s Figure 1 provides an example, where the discussed “strokes” are “shown as orange arrows”:



Okabe, 2; Greenspun, ¶171. Okabe’s algorithm then generates a flow animation in the target image corresponding to the flow direction and speed specified by the strokes, and permits “infinite repetition” of such an animation. Okabe, 1, 8; Greenspun, ¶171. According to Okabe, such simple and graphical commands allow

a user to animate images “with less effort than with previous methods.” Okabe, 1; Greenspun, ¶171.

3. The IMU-Okabe Combination

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

According to IMU, IMV6 is operated in the command-line and therefore “*is not a GUI image editor.*” IMU-Home, 2; Greenspun, ¶172. Thus, to create an animation from an image (e.g., a frame image extracted from a GIF animation using “+adjoin” or “-coalesce” in IMV6, *see* IMU-Animating, 7-8), 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, ¶¶172-73. For example, to animate a frame image using Shepard’s Distortion, a POSITA would have recognized to 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 desired direction and speed. *See* IMU-Distorting, 17-18 (creating an animation in the same way, but using IMV6’s “SRT distortion” effect instead, to animate the launch of a space ship along a non-linear path), 59, 19-20; Greenspun, ¶173.

Further, to make distortion and animation visible only in a particular region of the image, the user would write further commands applying a corresponding matte to the image such that only the particular region is visible during the distortion and

animation. IMU-Masking, 2; Greenspun, ¶174. Lastly, to save the animation as a looping animated GIF, the user would write further commands to save the animation using “-loop.” IMU-Animating, 1-2; Greenspun, ¶174; *see also, e.g.*, IMU-Distorting, 17-18.

But given this laborious, non-graphical process for extracting a frame and creating a subsequent animation of the frame image in IMV6, a POSITA would have been motivated to modify IMV6 (as described in IMU) by at least enabling a user to animate an already-extracted frame image using simpler and more intuitive graphical user commands. Greenspun, ¶175. In other words, a POSITA would have been motivated to modify IMV6 such that it *is* a “*GUI image editor*,” at least when animating images. *See* IMU-Home, 2; Greenspun, ¶175.

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, [etc]. So why should you bother to perform image processing by IM[V6]’s command line programs and scripts?” IMU-Windows, 1; Greenspun, ¶176. Indeed, AECS6 was another such “readily available” program that combined the ability to create a looping animation of effects applied to a frame image (e.g., using the Puppet effect) with simple and graphical user commands for defining such animation (e.g., by placing and moving Deform pins to define keyframes and motion

paths, 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. *See* AEM, 218-19, 187-89, 193-95, 317, 562; Section VI.A.1; Greenspun, ¶176.

Okabe teaches simple and graphical user commands that a POSITA would have found well-suited for implementing a GUI for at least animating an image (e.g., a frame image) in IMV6. Greenspun, ¶177. Specifically, Okabe teaches animating an image by first using an “alpha matte” to specify a to-be-animated “region of interest.” Okabe, 1; Greenspun, ¶177. The user also draws on the image a “sparse set of user-drawn strokes” to specify the desired direction and speed of animation, “which is a simple task and takes less than 1 min[ute].” Okabe, 7, 2, 3; Greenspun, ¶177. From these inputs, an infinitely repeating animation is automatically produced that follows the user-specified direction and speed. Okabe, 1-2, 8; Greenspun, ¶177. Okabe’s simple and graphical user commands thus enable creating looping animations “with less effort than with previous methods” and “markedly reduces the user burden.” Okabe, 1-2; Greenspun, ¶177.

Given such advantages, a POSITA would have found Okabe’s simple and graphical user commands desirable for animating an image (e.g., a frame image) in IMV6, i.e., obviating at least IMV6’s laborious and non-graphical animation procedure. Greenspun, ¶178. 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, ¶178.

Further, a POSITA would have been motivated to make such a modified version of IMV6 available for users to download on the ImageMagick.org website via the "Download Page" hyperlinked in IMU-Home. *See* IMU-Home, 1; Greenspun, ¶179. As IMU teaches, "I[MV6] is under constant development, new versions are released roughly on a monthly basis. It is strongly recommended to use an up-to-date version of IM[V6], especially when IM[V6] doesn't seem to perform a job quite as you expect it to do." IMU-Windows, 3; IMU-Home, 6; Greenspun, ¶179. And IMU provides the above hyperlink for users to download and install such new versions of IMV6. IMU-Home, 1; Greenspun, ¶179. Indeed, as a POSITA would have known, clicking the above hyperlink would have directed the user to the "Download ImageMagick" webpage of the ImageMagick.org website shown in EX1026, which provides a download link for the "latest release of I[MV6]." EX1026; IMU-Home, 1; Greenspun, ¶179. Given that modifying IMV6 in view of Okabe as discussed above would have likewise created a new version of IMV6, a POSITA would have thus found the hyperlinked "Download Page" ideal for hosting and providing such a new version of IMV6 to users. *See* IMU-Home, 1; Greenspun, ¶179.

b. Resulting Combination

So modified, the IMU-Okabe Combination allows a frame image from a GIF animation to be extracted and outputted using, e.g., IMV6's "+adjoin" or "-coalesce" operator. *See* IMU-Animating, 7-8; Greenspun, ¶180. The IMU-Okabe Combination then allows a subsequent animation of the image to be created using Okabe's simple and graphical user commands, where, rather than requiring complex commands to be written that place and move control points (e.g., for Shepard's Distortion) on the image frame-by-frame and in the desired direction and speed across the animation (*see* IMU-Distorting, 17-18, 59, 19-20), the IMU-Okabe Combination enables the same result to be achieved via user-drawn strokes 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, ¶180.

Also, similar to how IMV6 allows a user to apply a matte to make distortion and animation effects visible in only a portion of the image (IMU-Masking, 2), the IMU-Okabe Combination enables a user to apply 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, ¶181. Further, as IMV6 allows saving an animation as an infinitely looping animated GIF using "-loop" (IMU-Animating, 1-2), the IMU-Okabe Combination permits infinite repetition of the resulting animation (*see* Okabe, 1, 8). Greenspun, ¶181.

Lastly, the IMU-Okabe Combination, being a new version of IMV6, would have been made downloadable for installation onto the user's computer from ImageMagick.org's "Downloads Page" hyperlinked in IMU-Home. *See* IMU-Home, 1; Greenspun, ¶182.

4. Independent Claim 1

a. [1pre]

To the extent [1pre] is limiting, IMU teaches downloading and installing the most "up-to-date version" of IMV6 onto a user's computer from the ImageMagick.org website. IMU-Windows, 3; IMU-Home, 1, 6; Greenspun, ¶183. Okabe teaches that, when animating an image using Okabe's simple and graphical user commands, the user "uses a computer." Okabe, 1; Greenspun, ¶183. Given both IMV6 and Okabe's commands require a computer, the IMU-Okabe Combination would thus also be installed to run on a user's computer, which meets the claimed "computer system" and remaining claim language according to Plotagraph's infringement contentions. *See* EX1022, 134; Greenspun, ¶183.

Alternatively, IMU "strongly recommend[s] to use an up-to-date version of IM[V6]" (IMU-Windows, 3; IMU-Home, 6), and hyperlinks ImageMagick.org's "Downloads Page" for users to download and install such a new version of IMV6 onto their computers (IMU-Home, 1). Following IMU's recommendations, a user would have thus downloaded and installed the IMU-Okabe Combination

(“software”) onto the user’s computer (“client computing device” with “one or more client processors”) from this webpage—and thus a server (*see* Nakagawa, Abst.) (“computer system providing, to a client computing device, software...”). *See* Section VI.B.3.b; Greenspun, ¶184.

Further, the IMU-Okabe Combination enables a user to automatically animate an extracted frame of a GIF animation (“automating a shifting of pixels within a video file”). Sections VI.B.4.b-VI.B.4.g; Greenspun, ¶185.

b. [1a]

According to IMU, when using the “+adjoin” or “-coalesce” operators, IMV6—and thus the IMU-Okabe Combination—“read[s] in” a “GIF animation” from the current directory, meeting “access, from memory, a digital image file....” IMU-Animating, 7-8; IMU-Home, 4; Section VI.B.3.b; ’641 Patent, claim 3; Greenspun, ¶186; *see also* AEM, 100 (listing “Animated GIF (GIF)” as a “[v]ideo and animation format[.]”); ’641 Patent, 10:13-24 (stating “animated GIF” is one “format[.]” for viewing the video file after pixel shifting is applied); IMU-Animating, 15-19.

c. [1b]

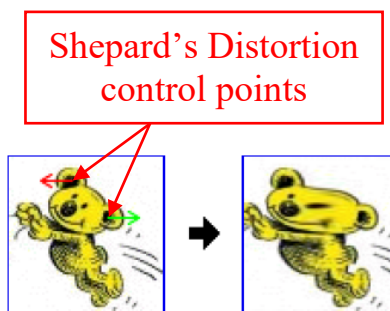
IMU teaches that “+adjoin” and “-coalesce,” as used by the IMU-Okabe Combination, “split[] an animation into frames” to allow a user to “use the individual

frames for other projects,” such as for creating subsequent animations.¹⁰ IMU-Animating, 7-8; Section VI.B.3.b; Greenspun, ¶187; *see also, e.g.*, IMU-Animating, 7. Any of such extracted frame images meets “first image frame.” Greenspun, ¶187.

Additionally, IMU teaches that IMV6 enables a user to apply effects such as distortion effects (e.g., “Shepard’s Distortion”) to convert one image (e.g., a frame image) into a sequence of frames that animate the image when played sequentially. IMU-Home, 2; IMU-Distorting, 17-18, 59; IMU-Animating, 7-8; Greenspun, ¶188. To do so, the user places a “control point” on a pixel at a user-specified coordinate of the image and then moves the control point to a new coordinate. IMU-Distorting, 19-20; Greenspun, ¶188; *see also, e.g.*, IMU-Distorting, 17-18. This is done

¹⁰ Should Plotagraph argue that IMU teaches applying Shepard’s Distortion to animate only an image and not specifically the extracted frame image, a POSITA would have nevertheless been motivated to use such an extracted frame as the image to be animated because such a frame would have been easily accessible and, in common instances, particularly desirable for modifying and animating with IMV6’s effects, e.g., Shepard’s Distortion. Greenspun, ¶188 n.18; *see also, e.g.*, ’641 Patent, 1:36-38; AEM, 590; IMU-Animating, 7-8 (teaching that extracted frames are useful not only “for other projects” but also to “study, edit, modify and re-optimize” the original GIF animation).

incrementally for each successive frame such that the control point moves across the animation at desired direction and speed. *See* IMU-Distorting, 19-20; Greenspun, ¶188; *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, ¶188. IMU also provides a figure showing the koala image before and after the Shepard's Distortion is applied:

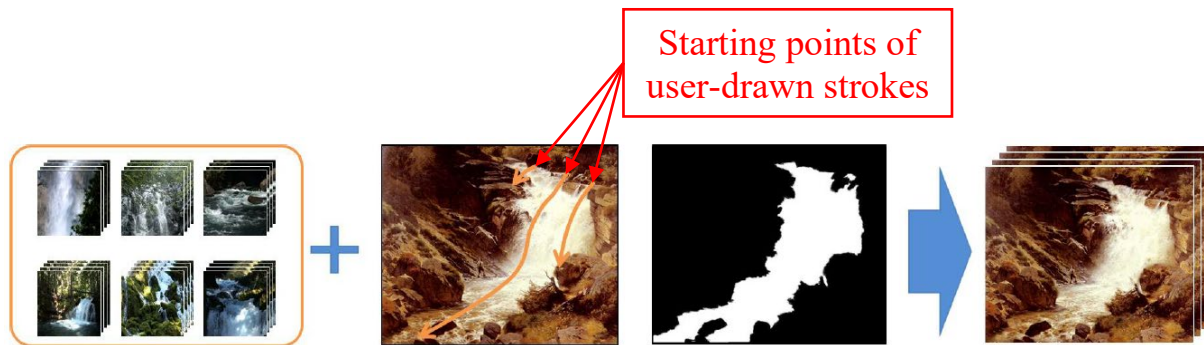


IMU-Distorting, 59 (additional annotations added); Greenspun, ¶188.

Further, in the IMU-Okabe Combination, IMU's placement and movement of a Shepard's Distortion control point corresponds to Okabe's user-drawn "stroke[]" on the frame image. *See* Okabe, 7, 3; Section VI.B.3.b; Greenspun, ¶189. That is, Okabe's user-drawn stroke as used in the IM-Okabe Combination includes an indication of a starting point, direction, and speed across the animation, like how a user in IMV6 indicates a control point's starting coordinate, 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, ¶189. Receiving the starting point of a user-drawn stroke on the frame image thus meets [1b]. *See* IMU-Distorting, 59, 19-20; Okabe, 7, 3; Greenspun, ¶189.

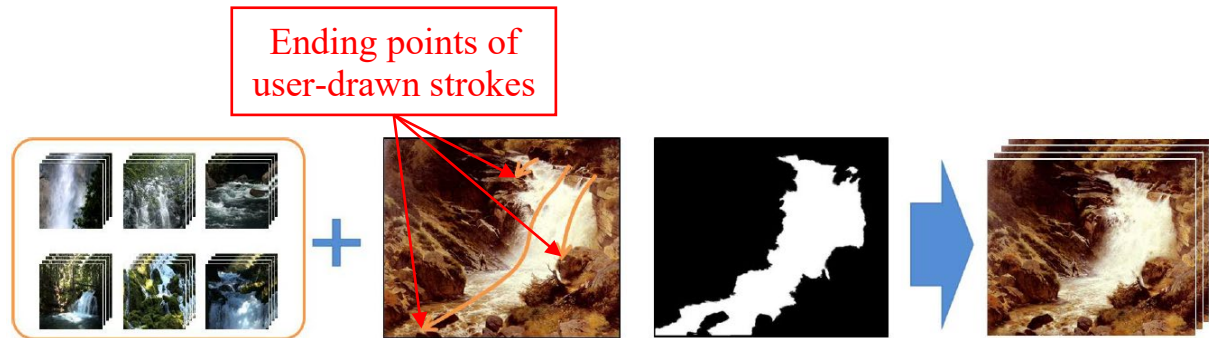
Okabe's Figure 1 illustrates the relevant disclosure by depicting several user-drawn strokes (shown in orange) on an image of a waterfall:



Okabe, 2 (Figure 1, annotations added); Greenspun, ¶190.

d. [1c]

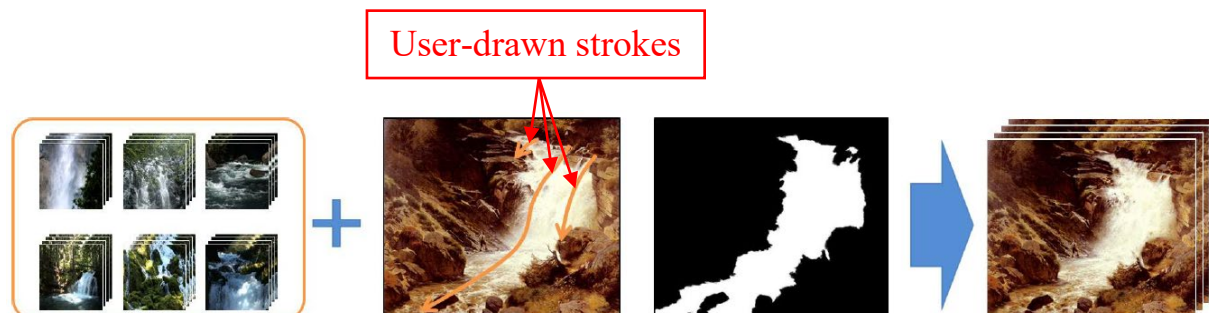
In the IMU-Okabe Combination, a user defines a Shepard's Distortion control point's direction and speed across the animation of a frame image by drawing a "stroke[]" on the image. *See* Okabe, 7, 2-3; IMU-Distorting, 59; Section VI.B.3.b; Greenspun, ¶191. Receiving this user-drawn stroke on the frame image, specifically the stroke's ending point, meets [1c]. Greenspun, ¶191; *see also* Okabe, 2 (Figure 1, reproduced below with annotations added).



e. [1d]

IMU teaches animating an image (e.g., a frame image) in IMV6 using distortion effects (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 desired direction and speed. *See* IMU-Distorting, 17-18, 59, 19-20; IMU-Animating, 7-8; Greenspun, ¶192.

Further, in the IMU-Okabe Combination, a user draws a “stroke[]” on the frame image to specify such direction and speed of a Shepard’s Distortion control point. *See* Okabe, 7, 2-3; Section VI.B.3.b; Greenspun, ¶193. Drawing this stroke thus meets “create a first digital link” comprising a “first direction” and “first length.” Greenspun, ¶193. Indeed, Okabe’s Figure 1 illustrates three user-drawn strokes, each comprising a direction (indicated by the arrowhead) and length:

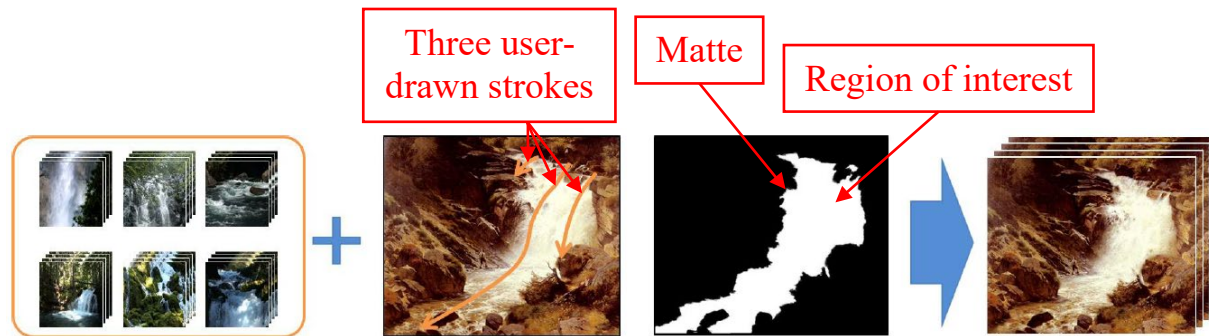


Okabe, 2 (Figure 1, annotations added); Greenspun, ¶193; *accord* Okabe, 6 (discussing Figure 8-a).

f. [1e]

IMU teaches modifying the transparency of pixels of an image (e.g., an extracted frame) in IMV6—and thus the visibility of effects applied to the frame 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; IMU-Animating, 7-8; Greenspun, ¶194.

The IMU-Okabe Combination similarly allows a user to apply a matte specifying a to-be-animated region of interest (*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, ¶195. Thus, a user would have been allowed to select any or all pixels of the image to be included in this region of interest, including pixels that lie along the stroke between the stroke’s starting and ending points, by using a matte, meeting [1e]. *See* Okabe, 1; Greenspun, ¶195. Okabe’s Figure 1 confirms this by depicting a matte specifying a to-be-animated region of interest (in white) that includes pixels that lie along three user-drawn strokes between each stroke’s starting and ending points:



Okabe, 2 (Figure 1, annotations added); Greenspun, ¶195.

g. [1f]

The '641 Patent states the claimed pixel “shift[ing]” can be performed using “a warping function, such as Shepard’s distortion.” ’641 Patent, 9:27-29; Greenspun, ¶196. IMU explicitly teaches that IMV6 performs “Shepard’s Distortion” and therefore the claimed pixel shifting. IMU-Distorting, 59; Greenspun, ¶196. That is, IMV6’s user shifts pixels as claimed by using Shepard’s Distortion to place and incrementally move a “control point” on a frame image in the user-specified direction and speed. *See* IMU-Distorting, 17-18, 59, 19-20; IMU-Animating, 7-8; Greenspun, ¶196.

Further, in the IMU-Okabe Combination, the user specifies a Shepard’s Distortion control point’s direction and speed by drawing a corresponding “stroke[]” on the image. *See* Okabe, 7, 3, 2; Sections VI.B.3.b, VI.B.4.e; Greenspun, ¶197. The IMU-Okabe Combination also allows a user to apply a matte to select pixels that lie along the stroke between the stroke’s starting and ending points—“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, ¶197. The IMU-Okabe Combination then automatically generates animation frames by shifting the selected pixels in the stroke's direction using IMV6's Shepard's Distortion, meeting [1f]. *See* Okabe, 1-2; Section VI.B.3.b; Greenspun, ¶197.

5. Claim 2

See Sections VI.B.4.c-VI.B.4.d. The stroke is drawn on a single frame image, and thus the stroke's ending point is on the same frame image as the stroke's starting point, meeting claim 2. Greenspun, ¶198.

6. Claim 3

See Section VI.B.4.b. The animated GIF discussed is a “video file” as claimed, and the extracted frame image is “a first video frame of the video file.” Greenspun, ¶199.

7. Claim 4

See Sections VI.B.4.c-VI.B.4.d, VI.B.6. While the stroke is drawn on a single frame image, and thus the stroke's starting and ending points are on the same frame image, such meets claim 4 because the “first” and “second video frame[s]” may be the same frame. '641 Patent, 6:51-54; EX1022, 137; Greenspun, ¶200.

8. Claim 8

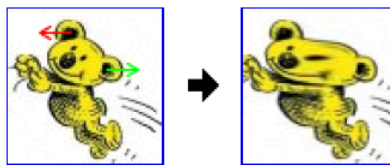
IMU teaches saving an animation of an image (e.g., a frame image) in IMV6 as an infinitely looping animated GIF using IMV6's “-loop” operator. IMU-Animating, 1-2, 7-8; Greenspun, ¶201. Similarly, Okabe describes automatically

producing an infinitely repeating animation that follows a direction and speed based on a user-drawn stroke. Okabe, 1-2, 8; Greenspun, ¶201. The IMU-Okabe Combination thus likewise permits infinite repetition of an animation of the frame image by, e.g., allowing a user to apply “-loop” to an animation created by Shepard’s Distortion and the user’s stroke, meeting claim 8. *See* IMU-Animating, 1-2; Okabe, 1, 8; Section VI.B.3.b; Greenspun, ¶201.

9. Claim 9

a. [9a]-[9e]

IMU teaches applying multiple Shepard’s Distortion control points at different coordinates of an image (e.g., a frame image) in IMV6 and incrementally moving the control points to animate the image. *See* IMU-Distorting, 17-18, 59, 19-20; IMU-Animating, 7-8; Greenspun, ¶202. IMU provides an example of one such increment using two Shepard’s Distortion control points on an image of a koala:

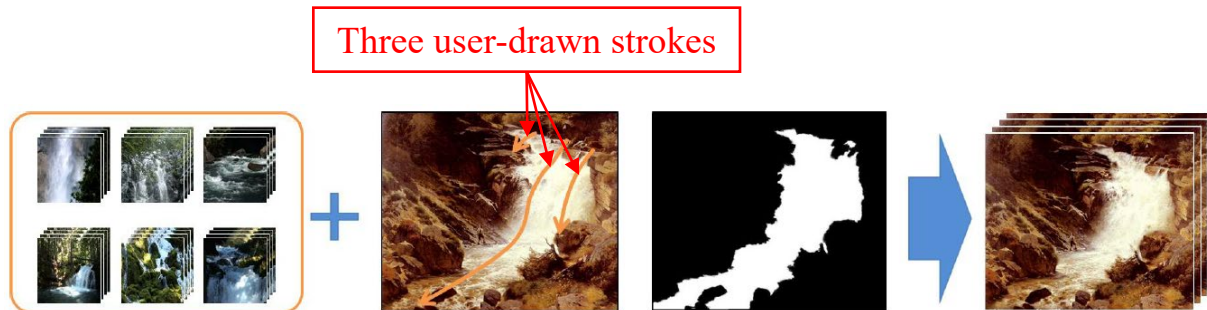


IMU-Distorting, 59; Greenspun, ¶202.

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 frame image, each specifying a direction and speed of

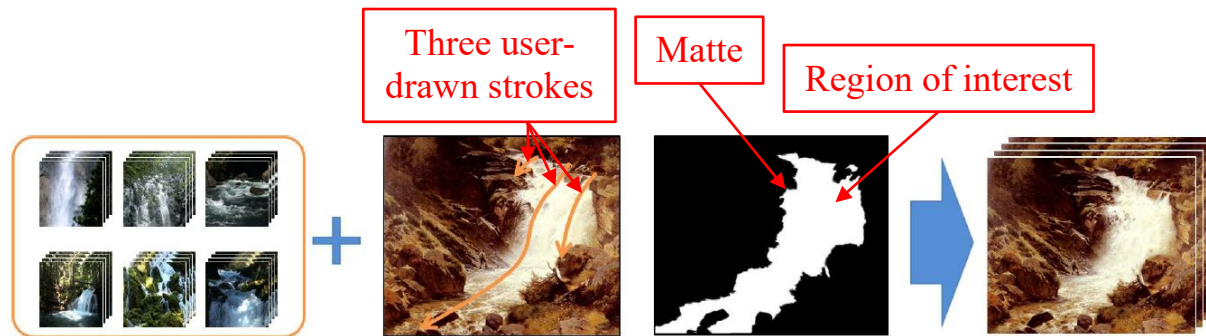
a different control point. *See* Okabe, 7, 2-3; Section VI.B.3.b; Greenspun, ¶203.

Indeed, Okabe's Figure 1 depicts three user-drawn strokes on an image:



Okabe, 2 (Figure 1, annotations added); Greenspun, ¶203; *accord* Okabe, 6 (discussing Figure 8-a). Receiving multiple user-drawn strokes—including each stroke's starting and ending points, direction, and speed—meets [9a]-[9c]. Greenspun, ¶203; *see also* Sections VI.B.4.c-VI.B.4.e.

The IMU-Okabe Combination also allows a user to apply a matte specifying a to-be-animated region of interest, including pixels that lie along each stroke between each stroke's starting and ending points, meeting [9d]. *See* Okabe, 1; IMU-Masking, 2; Section VI.B.3.b; Greenspun, ¶204; *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 that lie along three strokes between each stroke's starting and ending points:

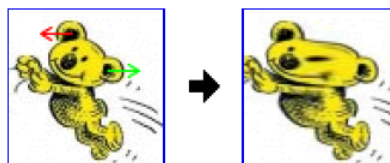


Okabe, 2 (Figure 1, annotations added); Greenspun, ¶204.

After receiving the strokes and matte, the IMU-Okabe Combination automatically generates animation frames using Shepard's Distortion according to the strokes, meeting [9e]. *See* Okabe, 1-2; IMU-Animating, 1-2; Section VI.B.3.b; Greenspun, ¶205; *see also* Section VI.B.4.g,

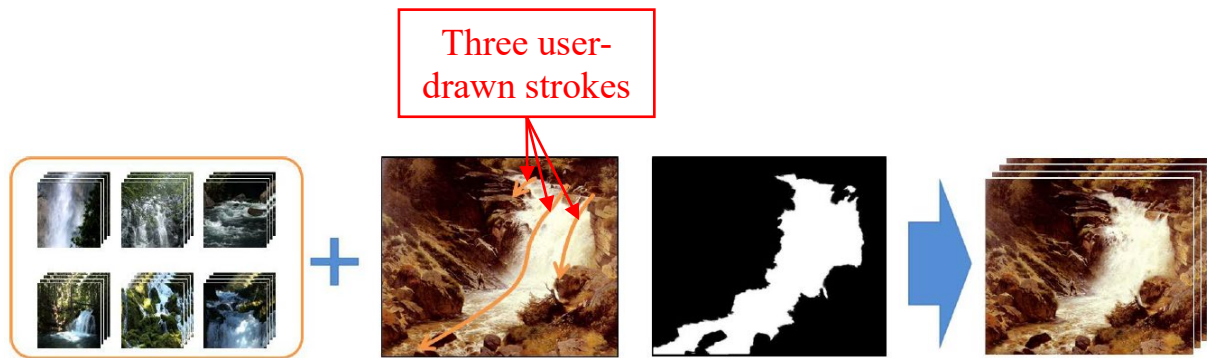
10. Claim 10

IMU teaches applying and incrementally moving control points, such as for Shepard's Distortion, to different coordinates on an image (e.g., a frame image) to generate an animation. *See* IMU-Distorting, 17-18, 59, 19-20; IMU-Animating, 7-8; Greenspun, ¶206. 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, ¶206.

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 frame image, which specify different directions and speeds of corresponding Shepard's Distortion control points across the animation, meeting claim 10. *See* Okabe, 7, 3; Section VI.B.3.b; Greenspun, ¶207. In fact, Okabe's Figure 1 depicts three user-drawn strokes with different directions:



Okabe, 2 (Figure 1, annotations added); Greenspun, ¶207.

11. Claim 11

IMU teaches that distortions using control points in IMV6, 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, ¶208. Specifically, "the 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, ¶208. Thus, when using Shepard's Distortion to animate, 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, ¶208.

In the IMU-Okabe Combination, the placement and movement of multiple Shepard’s Distortion control points is determined by a user-drawn “stroke[]” on the image that specifies 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, ¶209. 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, ¶209. Such establishes the claimed “proportional[]” relationships. Greenspun, ¶209.

12. Independent Claim 12

a. [12pre]

To the extent [12pre] is limiting, IMU “strongly recommend[s] to use an up-to-date version of IM[V6]” (IMU-Windows, 3; IMU-Home, 6), and hyperlinks ImageMagick.org’s “Downloads Page” for users to download and install such a new version of IMV6 onto their computers (IMU-Home, 1). Greenspun, ¶210. Following IMU’s recommendations, a user would have thus downloaded and installed the IMU-Okabe Combination from this webpage—and thus a server (*see* Nakagawa, Abst.) (“computer program product...”)—onto the user’s computer

(“when transmitted to a remote computer system for execution at a processor, cause the remote computer system to perform”). Section VI.B.3.b; Greenspun, ¶210.

Further, using the IMU-Okabe Combination meets the claimed “method for automating a shifting of pixels within an image file.” Sections VI.B.12.b-VI.B.12.f; Greenspun, ¶211.

b. [12a]

See Sections VI.B.4.b-VI.B.4.c. Receiving the stroke, including the stroke’s starting point, meets “receiving a first indication of a first starting point through a user interface.” Greenspun, ¶212.

c. [12b]

See Sections VI.B.4.d-VI.B.4.e. Receiving the discussed user-drawn stroke, which indicates the Shepard’s Distortion control point’s starting and ending points as well as direction and speed across the animation, meets [12b]. *See* IMU-Distorting, 17-18, 59, 19-20; Okabe, 7, 2-3; Section VI.B.3.b; Greenspun, ¶213.

d. [12c]

See Section VI.B.4.e. The discussed user-drawn stroke indicates the Shepard’s Distortion control point’s starting and ending coordinates as well as direction and speed across the animation, meeting [12c]. *See* IMU-Distorting, 17-18, 59, 19-20; Okabe, 7, 2-3; Section VI.B.3.b; Greenspun, ¶214.

e. [12d]

As discussed similarly in Section VI.B.4.f, the IMU-Okabe Combination's user selects any or all pixels of the image to be included in a to-be-animated region of interest, including pixels that are along the stroke and extend in the stroke's direction away from the stroke's starting point, by using a matte, disclosing [12d]. Greenspun, ¶215. Indeed, Okabe's Figure 1 depicts a matte specifying a to-be-animated region of interest (in white) that includes pixels that are along three user-drawn strokes and extend in each stroke's direction away from each stroke's starting point. Okabe, 2; Greenspun, ¶215.

f. [12e]

See Section VI.B.4.g.

13. Claim 13**a. [13a]-[13b]**

Similar to how IMV6's user applies a matte to modify the transparency of the pixels of an image (e.g., a frame image) and therefore the visibility of effects applied to the pixels (IMU-Masking, 2; IMU-Animating, 7-8), the IMU-Okabe Combination's user applies a matte specifying a to-be-animated region of interest (*see* Okabe, 1). Section VI.B.3.b; Greenspun, ¶217. Naturally, the matte also specifies a region that includes pixels the user does not desire to animate. Greenspun, ¶217. Indeed, Okabe's Figure 1 depicts a matte that includes white and black regions for pixels to be or not to be animated:



Okabe, 2 (Figure 1, annotations added); Greenspun, ¶217. Therefore, the matte meets “a first mask...” Greenspun, ¶217. Receiving and applying such a matte specified by the user meets the user meets “receiving...” Greenspun, ¶217.

14. Claim 14

As discussed in Section VI.B.13.a, the IMU-Okabe Combination’s user applies a matte specifying a region of an image (in black) covering pixels the user does not desire to animate. *See* Okabe, 1-2; Greenspun, ¶218. Receiving such a region meets claim 14. Greenspun, ¶218.

15. Independent Claim 19

a. [19pre]

To the extent [19pre] is limiting, IMU “strongly recommend[s] to use an up-to-date version of IM[V6]” (IMU-Windows, 3; IMU-Home, 6), and hyperlinks ImageMagick.org’s “Downloads Page” for users to download and install such a new version of IMV6 onto their computers (IMU-Home, 1). Greenspun, ¶219. Following IMU’s recommendations, a user would have thus downloaded and installed the IMU-Okabe Combination onto the user’s computer (“client computing

device”) from this webpage—and thus a server (*see* Nakagawa, Abst.) (“transmitting to a client computing device instructions...”). *See* Section VI.B.3.b; Greenspun, ¶219.

Further, the IMU-Okabe Combination is used “for shifting pixels within a video file.” Sections VI.B.15.b-VI.B.15.g; Greenspun, ¶220.

b. [19a]

See Section VI.B.4.b.

c. [19b]

See Section VI.B.4.c.

d. [19c]

See Section VI.B.4.d.

e. [19d]

See Section VI.B.4.e.

f. [19e]

See Section VI.B.4.f.

g. [19f]

See Section VI.B.4.g.

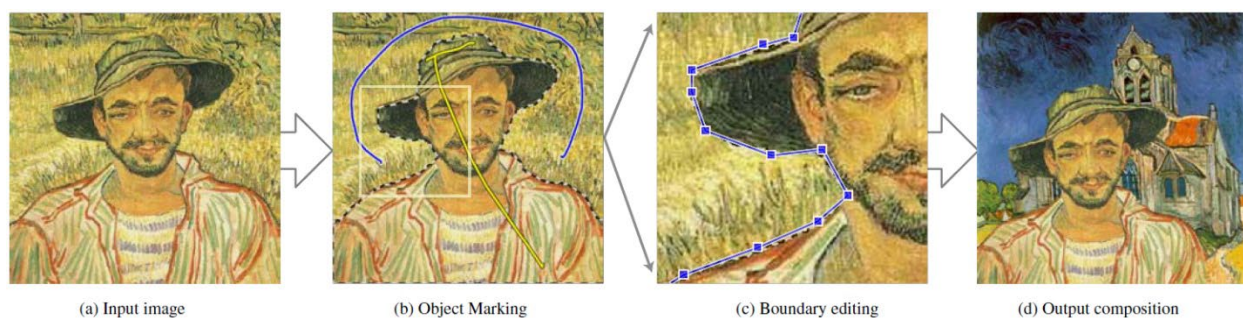
16. Claim 20

See Section VI.B.6.

C. Ground 3: IMU, Okabe, and Li Render Obvious Claims 13-15**1. Summary of Li**

Li describes “a novel coarse-to-fine UI design for image cutout” named “Lazy Snapping.” Li, 2; Greenspun, ¶228. “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, ¶228.

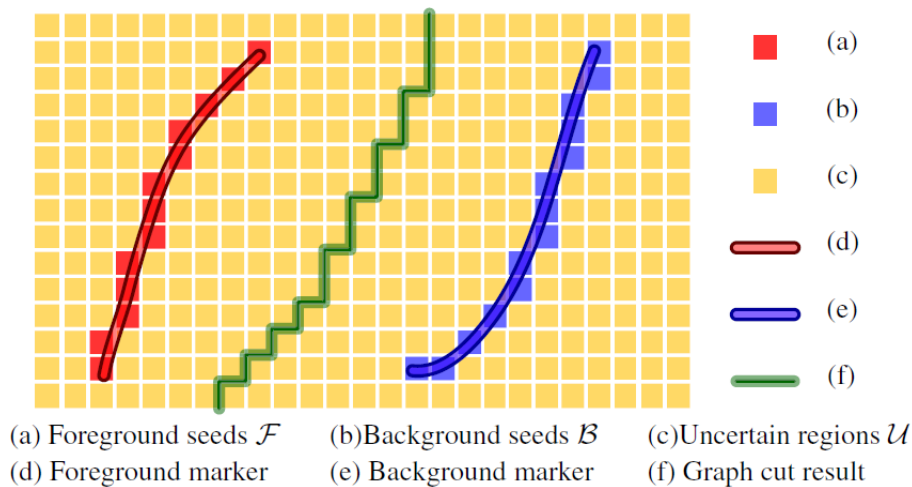
Li teaches that performing image cutout on a computer historically required a user to specify the foreground by “marking every pixel individually”—a “tedious” and “particularly frustrating task for users.” Li, 1; Greenspun, ¶229. “Lazy Snapping,” however, allows a user to easily perform image cutout in two steps: “a quick *object marking* step (b) and a simple *boundary editing* step (c).” Li, 2; Greenspun, ¶229. Li’s Figure 1 illustrates how image cutout is performed in “Lazy Snapping” accordingly:



Li, 1; Greenspun, ¶229.

The first “object marking” step “works at a coarse scale,” where the user begins by “mark[ing] 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, ¶230. After each marker line is drawn, a “segmentation process” is triggered that uses a “novel interactive graph cut algorithm” to detect a “cutout boundary” between the foreground and background lines. Li, 2; Greenspun, ¶230. Li’s Figure 2 depicts an example of detecting such a boundary (green) between a foreground line (red) and a background line (blue):



Li, 2-3; Greenspun, ¶230.

The second “boundary editing” step allows further refinement of the boundary “at a finer scale” by converting the boundary into “editable polygons” and allowing manual refinement of the boundary “by simply clicking and dragging polygon vertices.” Li, 3, 2; Greenspun, ¶231.

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; Greenspun, ¶233. Similarly, in the IMU-Okabe Combination, a user applies a matte that specifies a to-be-animated region of interest in an extracted frame image (shown in white), as well as a region not desired to be animated (shown in black). Okabe, 1-2; Sections VI.B.3.b, VI.B.13.a; Greenspun, ¶233.

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, ¶234. As such, Okabe explicitly teaches using "Lazy Snapping" to create a matte in Okabe and the IMU-Okabe Combination, and thus explicitly teaches combining the IMU-Okabe Combination with Li. Okabe, 7; Greenspun, ¶234.

Separately, Li teaches that "Lazy Snapping" 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, ¶235. Compared to 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, ¶235.

Given such benefits, a POSITA would have been motivated to utilize “Lazy Snapping” 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 transparent and the latter visible. *See* IMU-Masking, 2; Greenspun, ¶236. Further, given that a user of the IMU-Okabe Combination applies a matte that, as taught in Okabe, specifies regions in an image (e.g., an extracted frame image) to be or not to be animated (*see* Okabe, 1; IMU-Animating, 7-8; 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, ¶236. 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, ¶236.

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.a), but also functionality similar to “Lazy Snapping.” Greenspun, ¶237. Particularly, similar to Li’s “object marking” step (Li, 2-3), AECS6’s “Roto Brush” tool 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, ¶237. 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, ¶237.

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, e.g., an extracted frame image as discussed in Section VI.B.3.b, to indicate the image’s foreground (the to-be-animated region of interest) and background (the region of the image not desired to be animate). *See* Li, 2; Okabe, 1, 7; Greenspun, ¶238. The IMU-Okabe-Li Combination utilizes Li’s “novel image segmentation

algorithm” to detect the boundary between the frame image’s foreground and background based on these marker lines, as well as Li’s “boundary editing” step for manual refinement of the boundary. *See* Li, 1-2; Okabe, 7; Greenspun, ¶238. The resulting matte is then used in the remaining steps for animating the frame image discussed in Section VI.B.3.b. *See* Okabe, 1, 7; Greenspun, ¶238.

3. Claim 13

a. [13a]-[13b]

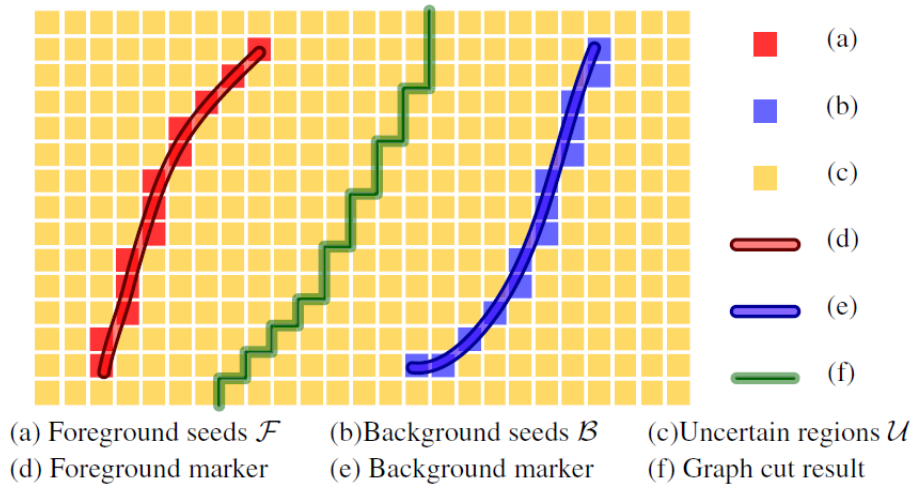
The IMU-Okabe-Li Combination allows a user to create a matte that specifies regions of an extracted frame image to animate (foreground) and not to animate (background). *See* Li, 1-2; Section VI.C.2.b; Greenspun, ¶239. Such a matte thus meets “a first mask....” Greenspun, ¶239.

To create this matte, the IMU-Okabe Combination’s user performs Li’s first “object marking” step of drawing marker lines on the image indicating the foreground and background. *See* Li, 2; Okabe, 1, 7; Section VI.C.2.b; Greenspun, ¶240. Receiving such marker lines, from which the matte is created, meets the claimed “receiving....” Greenspun, ¶240.

4. Claim 14

Li teaches that the “object marking” step of drawing marker lines on the image comprises drawing marker lines over the pixels that form the foreground and background. Li, 2; Greenspun, ¶241. Specifically, Li teaches, “[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, ¶241. Such foreground and background “pixels” or “seeds” are then used in Li’s segmentation algorithm, as utilized by the IM-Okabe-Li Combination, to detect the boundary between the foreground and background. Li, 2-3; Greenspun, ¶241; Section VI.C.2.b. Receiving the “pixels” or “seeds” of the marker lines thus meets claim 14. *See* Li, 2-3; Greenspun, ¶241.

5. Claim 15

a. [15a]-[15b]

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; Greenspun, ¶242. Such “pixels” or “seeds” are used by the IM-Okabe-Li Combination to detect the boundary between the foreground and background for

IPR2023-00568

U.S. Patent No. 11,182,641

generating the matte, meeting [15a]-[15b]. Li, 2-3; Section VI.C.4; Greenspun, ¶242; *see also* '641 Patent, 4:46-64.

VII. DISCRETIONARY DENIAL IS UNWARRANTED

In the Litigation, the '641 Patent was found to be patent-ineligible under 35 U.S.C. §101. EX1021, 17-35. Plotagraph has appealed this determination, which is now pending. *Id.*, 1. Such a pending Appeal that reviews “only the §101 issues” does not present sufficient concerns to warrant discretionary denial of this 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: April 11, 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 13,923 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 23, 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 84111
UNITED STATES

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