

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

LIGHTRICKS LTD.,
Petitioner,

v.

PLOTAGRAPH, INC. and SASCHA CONNELLY,
Patent Owner.

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

**PETITIONER'S REPLY TO
PATENT OWNER'S RESPONSE**

TABLE OF CONTENTS

I.	“IN A LOOP” SHOULD BE CONSTRUED ACCORDING TO THE TERM’S PLAIN AND ORDINARY MEANING.....	1
A.	The Plain and Ordinary Meaning of “in a loop” Includes Resetting and Repeating the Shifting of a Set of Pixels.	1
B.	Plotagraph’s Claim Construction Improperly Imports a Limitation from the Specification.	3
C.	The Petition is Consistent with the Specification, Unlike Plotagraph’s Construction of “in a loop.”	5
II.	AS THE PETITION SHOWED, AEM TEACHES APPLYING “LOOPOUT()” TO A DEFORM PIN AND WHAT RESULTS.....	8
III.	IMU AND AEM TEACH “SHIFTING” A “SET OF PIXELS.”	9
A.	The Specification Discloses that IMV6’s Shepard’s Distortion “shift[s]” a “set of pixels.”	10
B.	The Specification Discloses that the Triangular Mesh Algorithm of AECS6’s Puppet Effect “shift[s]” a “set of pixels.”	12
C.	The Petition Never Asserted that Okabe Teaches “shifting.”	21
IV.	THE IMU-OKABE COMBINATION SHIFTS A SET OF PIXELS “ALONG [A] NON-LINEAR PATHWAY.”	22
A.	IMU Discloses the Claim Language by Teaching Incrementally Moving Shepard’s Distortion Control Points Along a Curved Path	22
B.	The IMU-Okabe Combination Repurposes Okabe’s Strokes to Allow the Curved Path to be Defined Visually.....	26
V.	STROKES IN THE IMU-OKABE COMBINATION WOULD HAVE BEEN USER-VIEWABLE.	28
VI.	THE IMU-OKABE COMBINATION’S MATTE PERFORMS “MASK[ING]” OF PIXELS.....	29
VII.	AECS6’S MASK/MATTE PERFORMS “MASK[ING]” OF PIXELS AND “SELECTING”/“IDENTIFYING” OF A “SET OF PIXELS.”	29
A.	AEM Teaches Applying AECS6’s Mask/Matte, Which Performs the Claimed “mask[ing].”	30

B.	AEM’s Mask/Matte Disclosure Likewise Teaches the Claimed “selecting”/“identifying” of a “set of pixels.”	31
----	---	----

Petitioner's Exhibit List

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

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

		<i>Usage under Windows</i> , ImageMagick (Mar. 21, 2012), http://www.imagemagick.org/Usage/windows/ [https://web.archive.org/web/20120405151502/http://www.imagemagick.org/Usage/windows/]
1009	Okabe	Makato Okabe, et al., <i>Creating Fluid Animation from a Single Image using Video Database</i> , 30 Computer Graphics Forum 1973 (2011)
1010	Li	Yin Li, et al., Lazy Snapping, 23 ACM Transactions on Graphics 303 (2004)
1011	'017 PH	Prosecution history of U.S. Patent No. 10,346,017
1012	Archive	Declaration of Nathaniel E Frank-White of the Internet Archive
1013	Lindahl	U.S. Patent No. 9,466,127
1014	McNally	U.S. Patent No. 9,070,222
1015	Siddharth	U.S. Patent App. Pub. No. 2014/0087797
1016	Brookhart	U.S. Patent No. 9,400,942
1017	Sakiyama	U.S. Patent No. 9,692,974
1018	Forbes	Angus Graeme Forbes et al., <i>Analyzing Intrinsic Motion Textures Created from Naturalistic Video Captures</i> , 2014 International Conference on Information Visualization Theory and Applications 107 (2014)
1019	Luo	U.S. Patent No. 8,280,164
1020	Tang	U.S. Patent No. 8,913,847
1021		Notice of Docketing, <i>Plotagraph, Inc. v. Lightricks, Ltd.</i> , No. 23-1048, Dkt. No. 1 (Fed. Cir. Oct. 17, 2022)
1022		Plaintiffs' Unopposed Sworn Motion for Leave to Extend Time to Comply with P.R. 3-1 and P.R. 3-2 Dislcousre [sic] of Asserted Claims and Preliminary Infringement Contentions and Make Document Production, <i>Plotagraph, Inc. v. Lightricks Ltd.</i> , Civil Action No. 4:21-cv-03873, Dkt. No. 38 (S.D. Tex. May 5, 2022)
1023	N/A	Omitted
1024	N/A	Omitted

1025	Waiver	Waiver of the Service of Summons from <i>Plotagraph, Inc. v. Lightricks Ltd.</i> , 4:21-cv-03873 (S.D. Tex. 2021)
1026	Complaint Exhibits	Exhibits A-D of the Complaint from <i>Plotagraph, Inc. v. Lightricks Ltd.</i> , 4:21-cv-03873 (S.D. Tex. 2021)
1027	Rosenberg-Segalovitz	Declaration of Noa Rosenberg-Segalovitz
1028	Jan. 4 Service	January 4, 2022 Correspondence to Lightricks's Jerusalem, Israel Office from Israel's Courts Administration
1029	Cover Page Translation	English Translation of the Cover Page of the January 4, 2022 Correspondence to Lightricks's Jerusalem, Israel Office from Israel's Courts Administration
1030	Megidish	Declaration of Idan Megidish
1031	Matam Park	Map of Matam Park and List of Buildings in Matam Park, <i>available at</i> https://www.matam.co.il/wp-content/uploads/Map-of-Matam-Park-2021.pdf and https://www.matam.co.il/en/buildings/
1032	Waiver Correspondence	Email correspondence between Robert Greeson, Erik Janitens, and Marcellous McZeal regarding service of the complaint and waiver of service
1033	N/A	Omitted
1034	N/A	Omitted
1035	N/A	Omitted
1036	N/A	Omitted
1037	N/A	Omitted
1038	N/A	Omitted
1039	N/A	Omitted
1040	Greenspun2	Second Declaration of Philip Greenspun, Ph.D.
1041	Kernighan	Brian W. Kernighan & Dennis M. Ritchie, <i>The C Programming Language</i> (2d ed. 1988)
1042		Wayback Machine Capture dated Apr. 4, 2012 of <i>Inverse Distance Weighted Interpolation</i> , http://www.ems-i.com/smshelp/Data_Module/Interpolation/Inverse_Distance_Weighted.htm [https://web.archive.org/web/20120404084929

		/http://www.ems-i.com/smshelp/Data_Module/Interpolation/Inverse_Distance_Weighted.htm]
1043	Wampler	U.S. Patent App. Pub. No. 2017/0098326
1044	Igarashi	Takeo Igarashi et al., <i>As-Rigid-As-Possible Shape Manipulation</i> , 24 ACM Transactions on Graphics 1134 (July 1, 2005)
1045	Efros1	Wayback Machine Capture dated Sept. 9, 2015 of Alexei Efros, <i>Image Warping, 15-463: Computational Photography</i> , Carnegie Mellon University, Computer Science Department (Sept. 20, 2007), http://graphics.cs.cmu.edu/courses/15-463/2007_fall/Lectures/warping.pdf [https://web.archive.org/web/20070622221140/http://graphics.cs.cmu.edu/courses/15-463/2006_fall/www/Lectures/warping.pdf]
1046	Efros2	Wayback Machine Capture dated Sept. 10, 2015 of Alexei Efros, <i>Image Warping and Morphing, 15-463: Computational Photography</i> , Carnegie Mellon University, Computer Science Department (Sept. 25, 2007), http://graphics.cs.cmu.edu/courses/15-463/2007_fall/Lectures/morphing.pdf [https://web.archive.org/web/20150910054926/http://graphics.cs.cmu.edu/courses/15-463/2007_fall/Lectures/morphing.pdf]
1047		Wayback Machine Capture dated Sept. 10, 2015 of <i>15-463: Computational Photography</i> , Carnegie Mellon University, Computer Science Department (Fall 2007), http://graphics.cs.cmu.edu/courses/15-463/2007_fall/ [https://web.archive.org/web/20150910011432/http://graphics.cs.cmu.edu/courses/15-463/2007_fall/]
1048	IMU-Affine	Wayback Machine Capture dated May 25, 2012 of Anthony Thyssen, <i>ImageMagick v6 Examples - Affine Matrix Transforms</i> , ImageMagick (Oct. 10, 2009), http://www.imagemagick.org/Usage/distorts/affine/ [https://web.archive.org/web/20120525065408/http://www.imagemagick.org:80/Usage/distorts/affine/]
1049	Dong	European Patent No. 2,706,507 B1

IPR2023-00153

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

1050		<i>Lightricks Ltd. v. Plotagraph, Inc.</i> , IPR2023-00569, Paper 8 (PTAB Oct. 2, 2023)
------	--	---

I. “IN A LOOP” SHOULD BE CONSTRUED ACCORDING TO THE TERM’S PLAIN AND ORDINARY MEANING.

Plotagraph contends that “in a loop” should “be construed as meaning that a leading pixel in the selected set of pixels be rendered at the starting point of the shift and behind one or more trailing pixels which it previously was leading.” Paper 11 (“POR”), 21. The plain and ordinary meaning of “in a loop,” however, is broader, encompassing also the resetting and repeating of shifting a set of pixels as mapped in the Petition. Plotagraph attempts to justify its confined construction by pointing to the specification’s “examples,” but these examples fall short of redefining or disavowing the broad meaning of “in a loop.” At bottom, PO’s construction amounts to an improper importation of limitations from the specification into the claims and should therefore be denied.

A. The Plain and Ordinary Meaning of “in a loop” Includes Resetting and Repeating the Shifting of a Set of Pixels.

A POSITA would have recognized that the plain and ordinary meaning of “in a loop” includes other ways to “loop” beyond what Plotagraph proposes, including shifting a set of pixels and, upon completion, resetting the set back to its original position and repeating the shift. This is demonstrated by “existing systems for image processing and animation” (*see* Pet., 5-6), including those described in AEM and IMU, both of which indeed teach the latter way of looping all while using the exact word “loop” (Greenspun2, ¶¶3-5).

As the Petition explained, AEM teaches that AECS6's "**loop**Out()" expression by default causes "all keyframes [to] **loop**." AEM, 562; Pet., 16, 31; Greenspun2, ¶6. Thus, when applied to a Puppet effect Deform pin, the "loopOut()" expression causes the Deform pin and surrounding pixels in the mesh to shift along the defined motion path and to **only thereafter** reset to repeat this shift in what AEM expressly states is a "**loop**." AEM, 562, 219; Pet., 16, 31; Greenspun2, ¶6; *infra* at Section II.

Similarly, IMU teaches that IMV6's "**-loop**" operator specifies the "[n]umber of times [a] GIF animation is to cycle **th[r]ough** the image sequence before stopping." IMU-Animating, 2; Pet., 59; Greenspun2, ¶7. As IMU teaches, a GIF animation comprises frames played on a canvas where "the whole canvas is always cleared **at the end** of the animation sequence, **before it loops and repeats**." IMU-Animating, 4-6; Greenspun2, ¶7. And "**-loop**" specifies the number of such "loops"—i.e., how many times the GIF repeats **after** playing its animation sequence—where an "infinite **loop**" is the default. IMU-Animating, 2; Greenspun2, ¶7; *see also* Pet., 59. As such, when applied to a GIF animation created using, e.g., IMV6's Shepard's Distortion, "**-loop**" by default causes the Shepard's-Distortion-shifted pixels to reset **at the end** of the animation sequence, **not before**, and the shifting to repeat in expressly an "infinite **loop**." Pet., 56-59, 65; Greenspun2, ¶7.

Thus, both AEM and IMU demonstrate that, when shifting a set of pixels—whether with AECS6's Puppet effect or IMV6's Shepard's Distortion—the pixels

“loop” when, after shifting, the pixels are reset and the shifting is repeated. Greenspun2, ¶8. While other ways to loop may exist, AEM and IMU nonetheless confirm that a POSITA would have understood at least the above to be shifting “in a loop” under the term’s plain and ordinary meaning. Greenspun2, ¶8.

B. Plotagraph’s Claim Construction Improperly Imports a Limitation from the Specification.

As for Plotagraph’s construction, the claims themselves contain no language reflecting what Plotagraph proposes. While Plotagraph belabors that “in a loop” requires specific positioning of “a leading pixel” and “one or more trailing pixels” (POR, 20-28), the claims recite no such “leading pixel,” “trailing pixels,” or any such positioning, only “rendering and re-rendering in a loop the first set of pixels being shifted.” Greenspun2, ¶9.

Plotagraph nevertheless argues that, “because all the described embodiments” perform what Plotagraph construes as looping, the claim scope of “in a loop” should be confined to such looping. POR, 21-24. But such commits the exact type of limitation importation deemed improper in *Phillips v. AWH Corp.*:²

[A]lthough the specification often describes very specific embodiments of the invention, ***we have repeatedly warned***

² *Phillips*’ claim construction standard governs this proceeding. 83 Fed. Reg. 51340 (Oct. 11, 2018). Plotagraph’s construction is, however, “pursuant to the broadest reasonable interpretation” and thus rooted in the wrong standard. *See* POR, 20.

against confining the claims to those embodiments. In particular, we have expressly rejected the contention that if a patent describes only a single embodiment, the claims of the patent must be construed as being limited to that embodiment.

415 F.3d 1303, 1323 (Fed. Cir. 2005).³

Construed properly, “in a loop” instead “must be given [its] plain and ordinary meaning to one of skill in the art,” and “[t]here are only two exceptions to this general rule: 1) when a patentee sets out a definition and acts as his own lexicographer, or 2) when the patentee disavows the full scope of a claim term either in the specification or during prosecution.” *Thorner v. Sony Computer Entertainment America LLC*, 669 F.3d 1362, 1365-67 (Fed. Cir. 2012) (citing *Phillips*, 415 F.3d, 1313)). For either exception to apply, the specification must be “clear.” *Id.*, 1365-67.

Here, the specification nowhere provides any definition—much less Plotagraph’s—for “in a loop.” Nor is there any disclaimer of any other ways to perform looping. In fact, all disclosed embodiments are expressly labeled *exemplary*, not definitional or limiting. Greenspun2, ¶11. Regarding the “primary illustrated embodiment” in FIGS. 5A-5D (*see* POR, 21-23), the specification prefaces the corresponding discussion with, “[b]y way of *example*....” ’017 Patent,

³ Emphasis added throughout unless indicated otherwise.

7:20-21; Greenspun2, ¶12. And regarding specifically FIG. 5D, which Plotagraph alleges depicts a “loop around” (POR, 22), the specification again reiterates that such depicts a mere “*example*.” ’017 Patent, 7:66-8:4; Greenspun2, ¶12. Similarly, regarding the “alternative embodiment” (*see* POR, 23-24), the specification likewise reemphasizes that such is a mere “*example*” among “*various* alternative or additional embodiments.” ’017 Patent, 8:19-26; Greenspun2, ¶13.

By providing only illustrative, nonrestrictive, exemplary embodiments, the specification thus provides no indication that anything other than the plain and ordinary meaning of “in a loop” applies. Greenspun2, ¶14. Indeed, the specification expressly reiterates that “[t]he described embodiments are to be considered in all respects *only as illustrative and not restrictive*” and “[t]he present invention may be embodied in *other specific forms*,” such that “[t]he scope of the invention is... indicated by the appended claims rather than by the foregoing description.” ’017 Patent, 13:57-62; Greenspun2, ¶14; *see also Hill-Rom Servs., Inc. v. Stryker Corp.*, 755 F.3d 1367, 1373 (Fed. Cir. 2014) (declining to limit claims to disclosed embodiments based on similar boilerplate specification language).

C. The Petition is Consistent with the Specification, Unlike Plotagraph’s Construction of “in a loop.”

Plotagraph asserts that the specification “does not support Petitioner’s implicit construction for shifting ‘in a loop’” because the specification “fails to show any pixel being moved from the starting point to the ending point and then returning to

the starting point.” POR, 24-28. Plotagraph, however, misunderstands the law. As a corollary to the well-settled principle that the claims are not limited to the disclosed embodiments, “claims may embrace different subject matter than is illustrated in the specific embodiments in the specification.” *Phillips*, 415 F.3d at 1323 (citation and quotation marks omitted).

In re Power Integrations, Inc., 884 F.3d 1370 (Fed. Cir. 2018), *cited in* POR, 28, neither changes this well-settled principle nor helps Plotagraph. While the court there indeed found a construction “unsupported” by the specification, such was because the construction was “***inconsistent*** with both the specification... and the plain claim language.” *Power Integrations*, 884 F.3d at 1376-77.

Here, only Plotagraph’s overly narrow construction is “inconsistent” with the specification. Specifically, ***the specification expressly contemplates embodiments that loop after shifting a set of pixels using a “mesh algorithm”*** (e.g., that of AEM’s Puppet effect) ***or “Shepard’s distortion”*** (e.g., IMU’s “Shepard’s Distortion”) (’017 Patent, 8:28-37)—***both of which***, as Plotagraph acknowledges, ***“move[] pixels by stretching or deforming the image”*** (POR, 46; *infra* at Section III). Greenspun2, ¶16. Plotagraph, however, excludes these embodiments by arguing that these pixel “stretching” techniques are categorically “incapable of shifting a set of pixels ‘in a loop’” under Plotagraph’s construction. *E.g.*, POR, 46, 48, 2-3; Greenspun2, ¶16. By doing so, Plotagraph therefore proposes a construction

that is inconsistent with the specification and thus improper, as claim terms “normally [are] not interpret[ed]... in a way that excludes embodiments disclosed in the specification.” *Google LLC v. EcoFactor, Inc.*, No. 22-1750, slip op. 14 (Fed. Cir. Feb. 7, 2024) (citation and quotation marks omitted); Greenspun2, ¶17.

The Petition, by contrast, did not exclude these embodiments. Instead, the Petition not only relied on the discussed pixel “stretching” techniques for meeting the claimed “shifting” of a “set of pixels” (*infra* at Section III), but also explained how such shifting can be executed “in a loop”: by resetting and repeating the shift (Section I.A). Pet., 29-31, 72-73; Greenspun2, ¶18.

Further, even the specification itself suggests that such resetting and repeating qualifies “loop[ing]” as claimed. As disclosed, “[i]n various embodiments, a user is provided with an **animation duration**... option,” under which “a user may indicate that **each respective loop** in the pixel shifting only **take 2 seconds**.” ’017 Patent, 7:36-45; Greenspun2, ¶19. Such disclosure suggests that “**loop[ing]**” an entire shifting “**animation**” after a duration—e.g., 2 seconds—qualifies as shifting “in a loop.” Greenspun2, ¶19.

Thus, unlike Plotagraph’s overly confined construction, the Petition is consistent with the specification.

II. AS THE PETITION SHOWED, AEM TEACHES APPLYING “LOOPOUT()” TO A DEFORM PIN AND WHAT RESULTS.

Plotagraph asserts that the Petition failed to explain AEM’s “loopOut()” expression. POR, 45 (noting AEM’s discussion of “looping a segment of time” and “the In point and Out point of [a] layer” (citing AEM, 562)). But the Petition provided that explanation, including by quoting AEM’s discussion of how “loopOut()” works overall: “loopOut()” by “default” causes “all keyframes [to] loop.” AEM, 562, *quoted in* Pet., 16, 31; Greenspun2, ¶21. And, as the Petition explained, such a teaching resultantly also teaches that applying “loopOut()” specifically to a Deform pin would by “default” cause “all keyframes [*of the Deform pin* to] loop”—i.e., “the Deform pin’s position will be reset to the starting keyframe upon reaching the ending keyframe..., and the interpolation will be repeatedly looped....” Pet., 31; Greenspun2, ¶21. This exact result is also shown in the Petition’s screenshots of Dr. Greenspun’s animation, which thus show visually what AEM teaches in words alone. Pet., 31-34; Greenspun2, ¶21. Indeed, elsewhere in its POR, Plotagraph understands the Petition to be asserting the above. *E.g.*, POR, 13; Greenspun2, ¶21.

Plotagraph also argues that “no disclosure in AEM... suggest[s] applying a loopOut expression to a Deform pin.” POR, 45. The Petition, however, showed otherwise and did so by citing AEM, 562—which teaches that “loopOut()” is an “*expression*”—and AEM, 219—which teaches that “[y]ou can use *expressions* to

link the positions of *Deform pins* to motion tracking data, audio amplitude keyframes, or any other properties”—as well as Dr. Greenspun’s explanation of the same. AEM, 562, 219, *cited in* Pet., 31, *and quoted in* Greenspun, ¶106. Such teachings together sufficiently teach applying a “loopOut()” *expression* to a Deform pin. Greenspun2, ¶22.

III. IMU AND AEM TEACH “SHIFTING” A “SET OF PIXELS.”

Plotagraph argues that, regardless of the discussed looping, “IMU... does not teach *shifting of a set of pixels* but rather a *single* control point” (*e.g.*, POR, 38-39) and that AEM’s teachings are similarly deficient (*see, e.g., id.*, 14-16 & n.1). Although apparently admitting that a “*single*” pixel is “*shift[ed]*” by IMV6’s Shepard’s Distortion and AECS6’s Puppet effect, Plotagraph asserts that “[a]ny *other* pixels in an image are *translated* only as a result of the image being *stretched*,” not “*shift[ed]*” with the “*single*” pixel—i.e., the claimed “shifting” a “set of pixels.” POR, 38. In essence, Plotagraph’s argument is that only a *pure-translation* meets the claimed “shifting”; if a pixel is also “stretched” or otherwise deformed when translated, then it is not “*shift[ed]*.” Greenspun2, ¶23.

Plotagraph’s argument, however, directly contradicts the specification, which expressly states that “shifting” a “set of pixels” occurs even when pixels also undergo what Plotagraph characterizes as a “*stretch[ing]*.”

A. The Specification Discloses that IMV6’s Shepard’s Distortion “shift[s]” a “set of pixels.”

As the Petition explained, the specification discloses different exemplary ways to “*shift*” *not just a single pixel* but, explicitly, a “*set of pixels*” including “us[ing] a warping function, such as *Shepard’s distortion*.” ’017 Patent, 8:28-41; Pet., 72; Greenspun2, ¶25.

Perhaps recognizing that a POSITA already knew how Shepard’s distortion operates, the specification never again discusses Shepard’s distortion. Greenspun2, ¶26. And indeed, “Shepard’s distortion” was a recognized term of art. Greenspun2, ¶26. The term stems from a 1968 paper describing a method for interpolating unknown data values between and around “irregularly-spaced” apart “data *points*” based on distance. Shepard, 1-2; Greenspun2, ¶26. Shepard’s method can be used in “diverse” data contexts (Shepard, 1) and, in its general form, is commonly known as “Shepard’s interpolation” or “Shepard’s method” (e.g., EX1035). Greenspun2, ¶26.

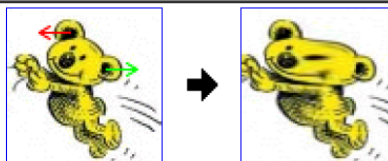
When used in image processing—especially IMV6—Shepard’s method is instead sometimes referred to as “Shepard’s distortion.” See Greenspun2, ¶27. But the underlying method remains unchanged: “Shepard’s distortion”—as it appears in both the specification and IMU—remains a “*point driven*” technique that, at a high-level, is “equivalent to a thick block of ‘taffy’ representing [a] source image[] having pins driven into it and then the pins moved around” (i.e., pins, or “control points,”

are irregularly-spaced apart). Greenspun2, ¶27. *Compare* IMU-Distorting, 62, 59, *with* Shepard, 1.

Hence, Shepard’s distortion “does not move areas, but *points*,” and “will swirl and stretch and compress the image between the control points, but it tries hard not to rotate or scale the image near the control points” (i.e., interpolates pixel values based on distance to a control point). Greenspun2, ¶28. *Compare* IMU-Distorting, 59-60, 62, *with* Shepard, 1-2.

Such Shepard’s distortion, as IMU teaches, is utilized in IMV6’s “Shepard’s Distortion” effect. IMU provides an example depicting a koala image being “pulled and stretched apart” by two IMV6 Shepard’s Distortion “control *points*” that, are placed and moved apart from one another by a specified amount of 10 pixels each:

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



IMU-Distorting, 59; Pet., 56, 67-68; Greenspun2, ¶¶29-30. As shown, only the control points move the 10-pixel amount; all other surrounding pixels are “stretched” or “compressed” based on distance to the control points. IMU-Distorting, 59; Greenspun2, ¶¶29-30.

And although IMV6's Shepard's Distortion as shown causes a pure translation of only control *points* while causing surrounding pixels to instead “stretch” and “compress” (IMU-Distorting, 59-60, 62), such Shepard's distortion nonetheless meets the claimed “*shift*,” *not just of one pixel* but, again, per the specification, a “*set of pixels*.” '017 Patent, 8:28-41; Pet., 72; Greenspun2, ¶31. Indeed, elsewhere in its POR, Plotagraph appears to agree by not only admitting that Shepard's distortion is a “*pixel-based shifting* mechanism,” but also distinguishing such from Okabe's “video patch[ing]” technique. POR, 37; *see also infra* at Section III.C.

B. The Specification Discloses that the Triangular Mesh Algorithm of AECS6's Puppet Effect “shift[s]” a “set of pixels.”

The specification also discloses that the claimed “shifting” a “set of pixels” can be performed “using a *mesh algorithm*”—particularly one that “triangulates the mesh using defined points and then calculates an affine transformation for every angle.” '017 Patent, 8:28-41; Pet., 29-30; Greenspun2, ¶32.

Like Shepard's distortion, such algorithms not only translate *but also reorient, stretch, and compress* pixels, particularly when calculating the disclosed “affine transformation[s],” which each indeed not only translate but also “scale,” “rotate,” and “shear” pixels. Greenspun2, ¶¶33-38 (citing IMU-Affine, 11, 1; Efros1, 24, 16, 21; Wampler, ¶[0003]; Igarashi, 7); *see also id.*, ¶¶39-41 (explaining how affine transformations continue to translate *and reorient, stretch, and compress* pixels when utilized in triangular mesh algorithms). Yet, even if such algorithms do

more than translate pixels, including stretching the pixels, such algorithms—again, per the specification—nonetheless “shift” a “set of pixels” as claimed. ’017 Patent, 8:28-41; Greenspun2, ¶42.

Such is also confirmed by the specification’s FIG. 4, which depicts such a triangular mesh algorithm in operation:

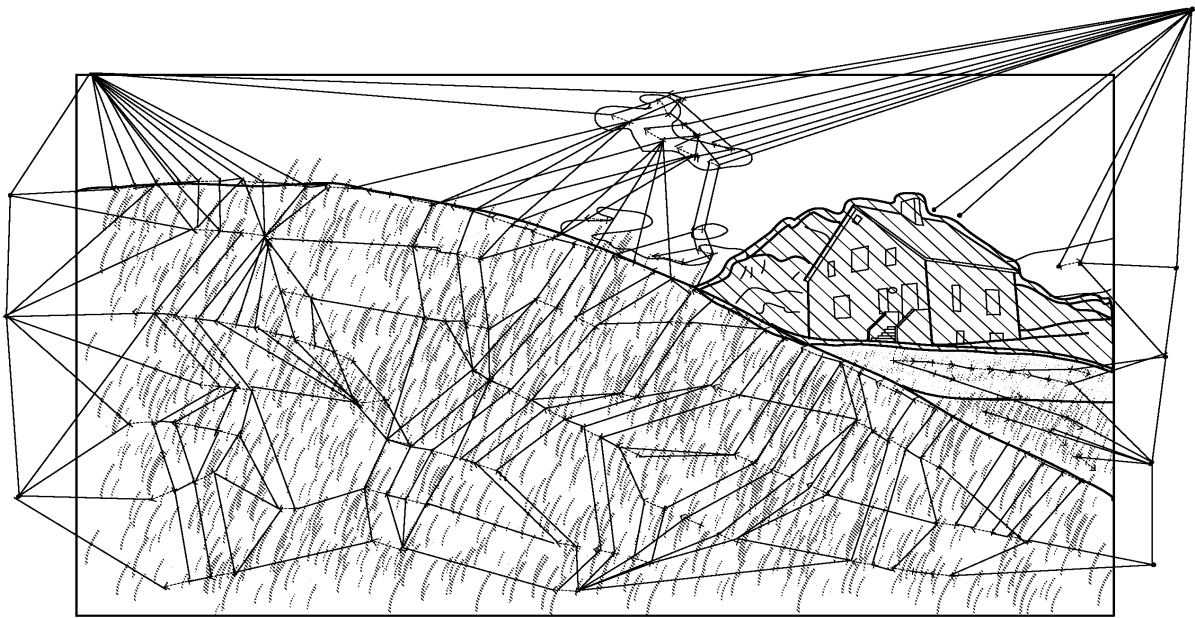


Fig. 4

See ’017 Patent, FIG. 4, 6:39-48; Greenspun2, ¶43. “FIG. 4 depicts a starting mesh and ending mesh overlaid on [a] digital image” (’017 Patent, 6:39-44), as well as “various links connecting beginning points and ending points” defined by the user (see *id.*, 10:55-11:28). Greenspun2, ¶43.

Below, one such **link** has been annotated in **red** alongside a **green**-annotated **edge**:

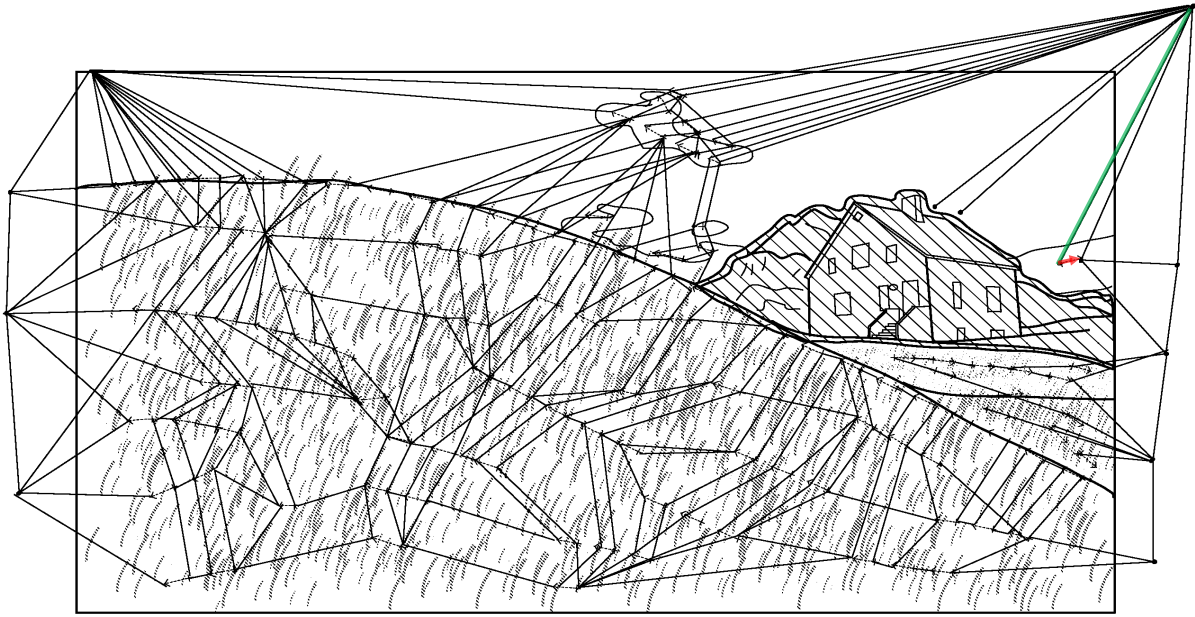


Fig. 4

Greenspun2, ¶44. As shown, the **link's starting** point is the lower vertex of the **edge**.

Greenspun2, ¶44. The **edge** thus, by association, is part of the “**starting** mesh”—
i.e., is a **starting edge**. See '017 Patent, 6:39-44; Greenspun2, ¶44.

Below, a **blue** annotation of an additional **edge** has been further added:

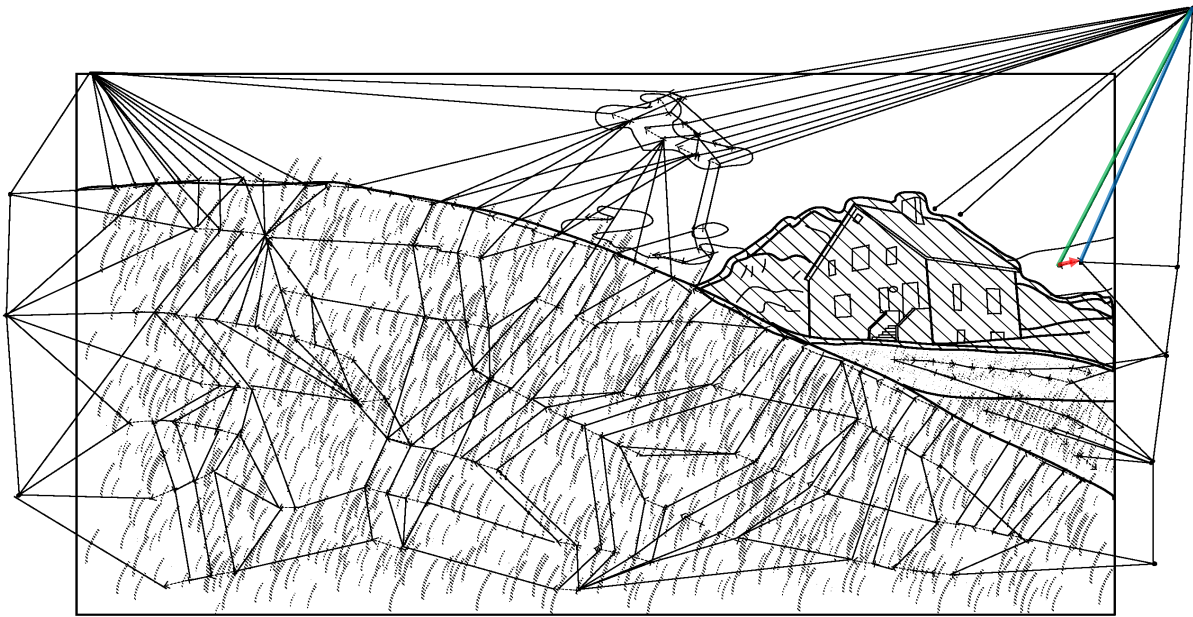


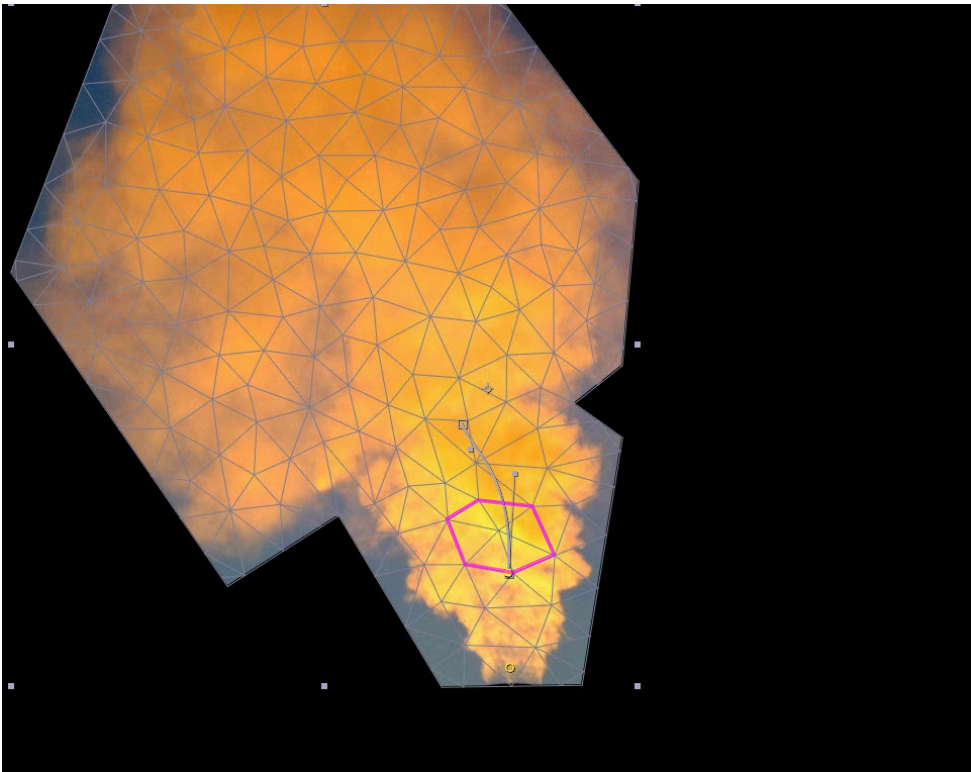
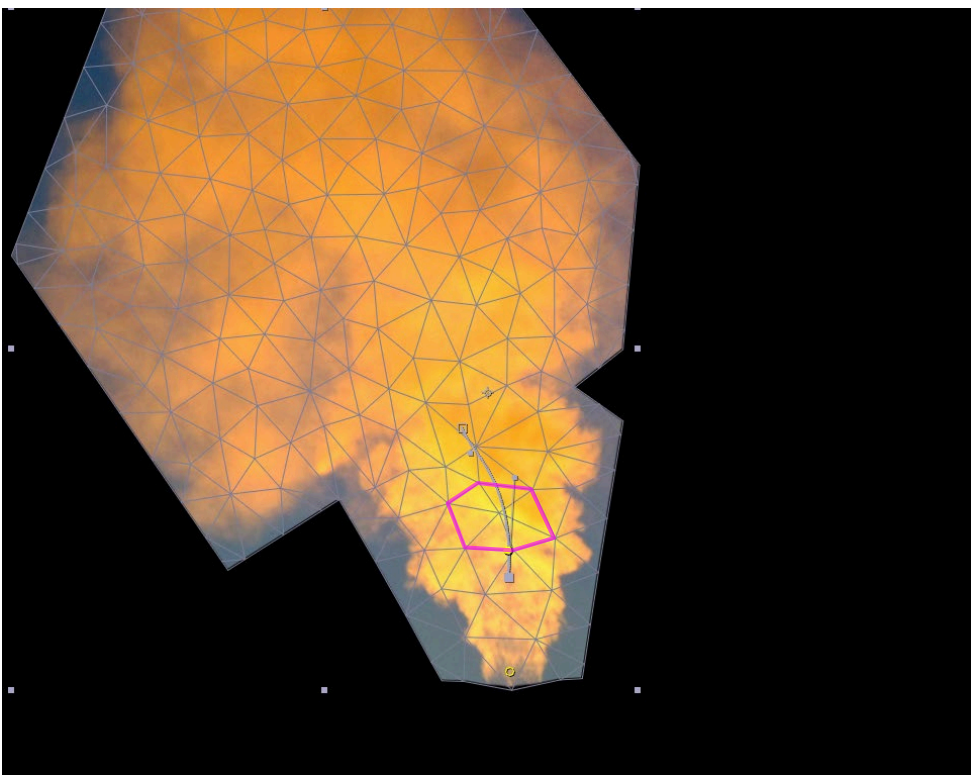
Fig. 4

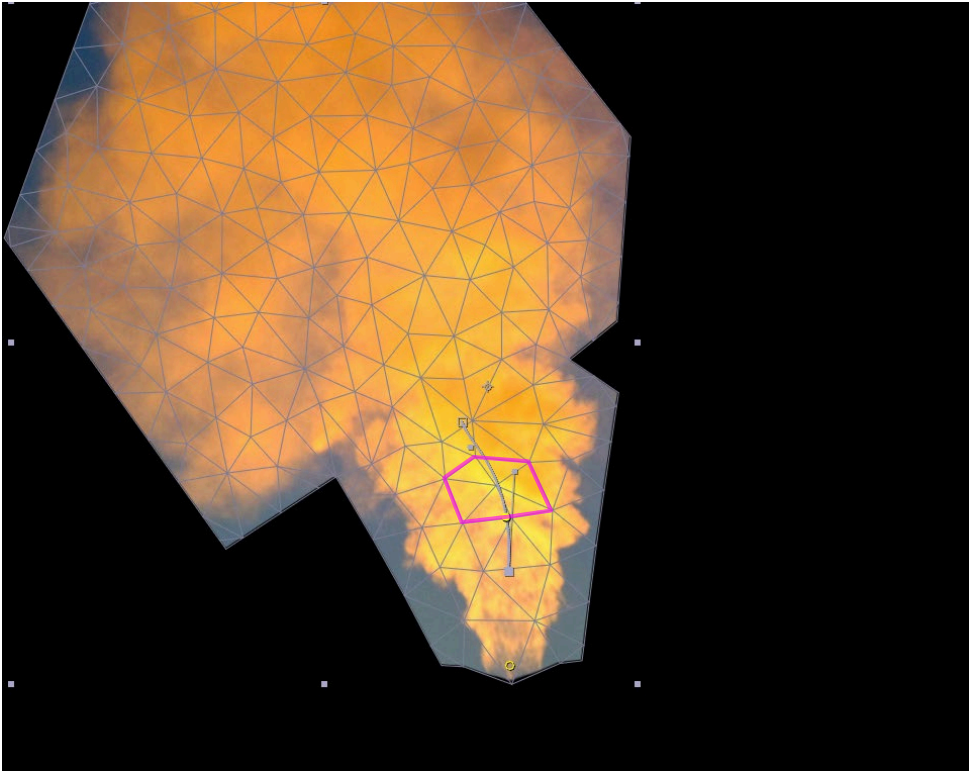
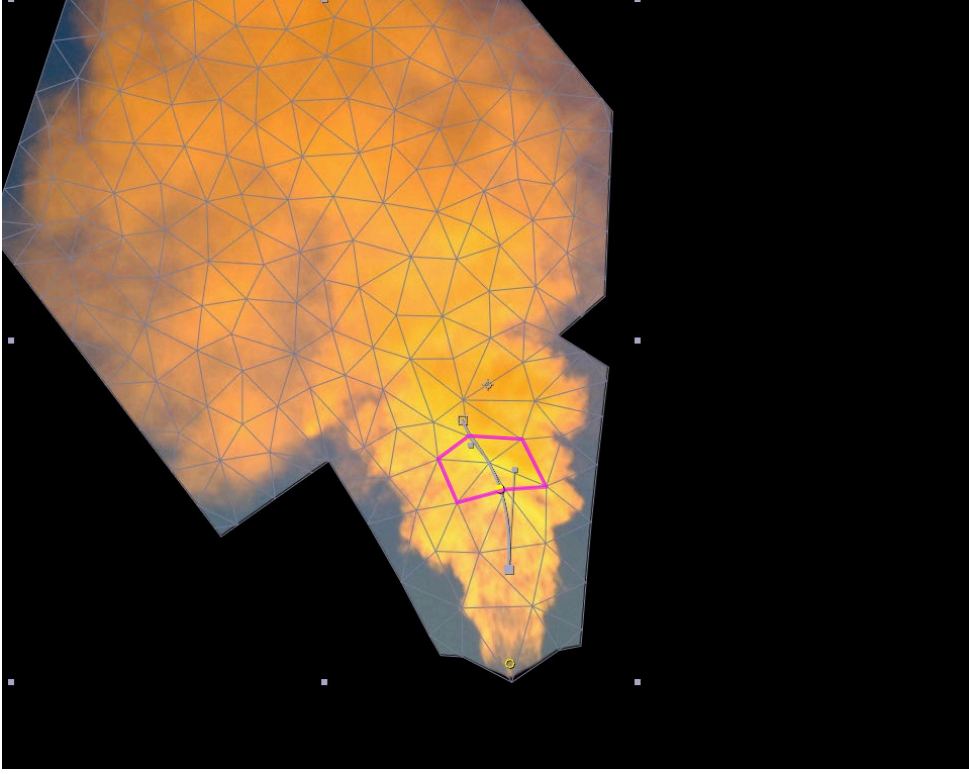
Greenspun2, ¶45. As shown, the **link's** ending point forms the lower vertex of this **edge**, which thus, again by association, is an **ending edge** within the “**ending mesh**.” See '017 Patent, 6:39-44; Greenspun2, ¶45. Further, because both share the same upper vertex, the **starting edge** and **ending edge** thus represent the **same edge** as located within the starting mesh and ending mesh, respectively. See '017 Patent, 6:39-44; Greenspun2, ¶45. I.e., the **ending edge** is located where the **starting edge** ultimately shifts to along the **link** with the starting point. See Greenspun2, ¶45.

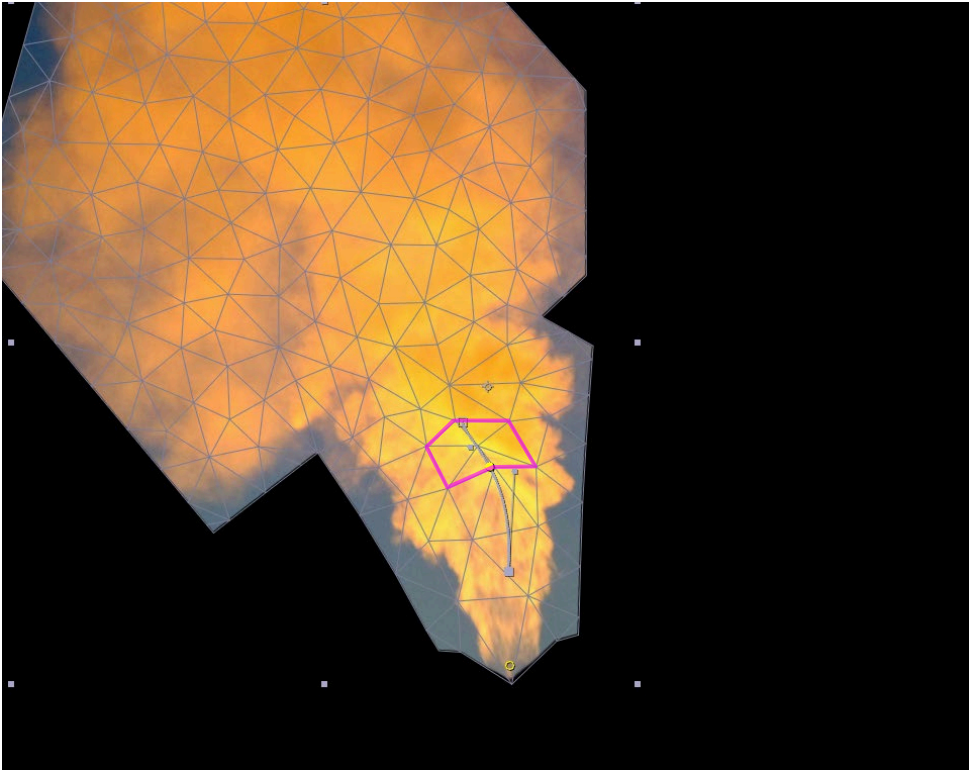
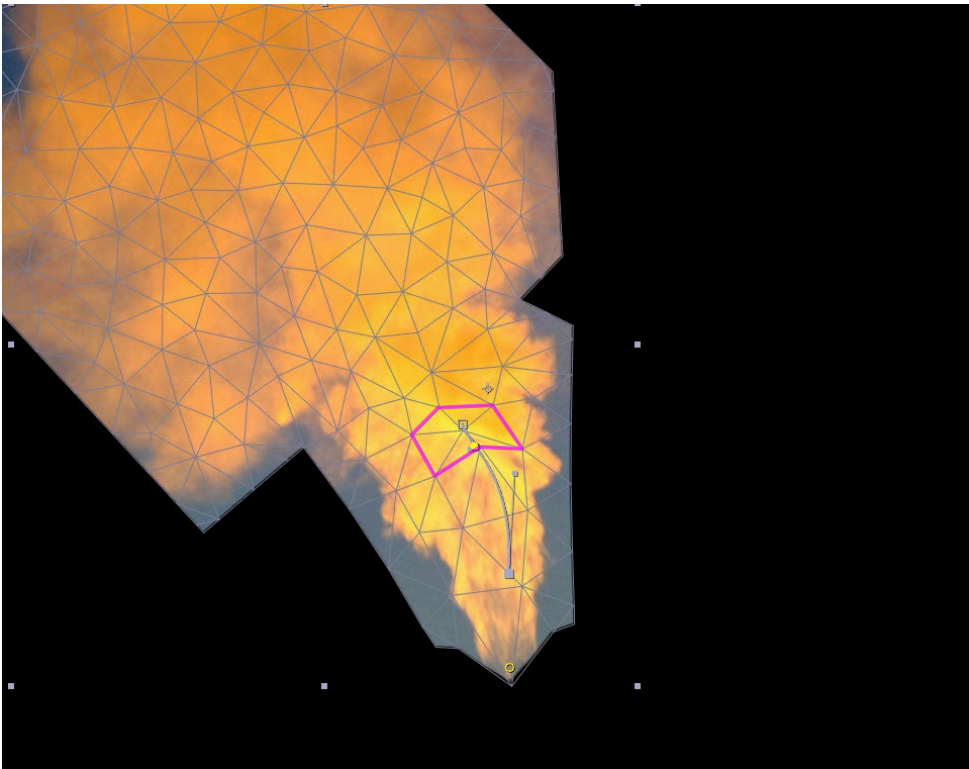
As illustrated, the **ending edge** is shorter and steeper-angled than the **starting edge**, and thus all pixels along the **starting edge**—some of which form part of the **link's** associated set of pixels (i.e., pixels parallel to, between the ends of, and within a threshold distance from the **link**) (see '017 Patent, 11:18-28)—are compressed and

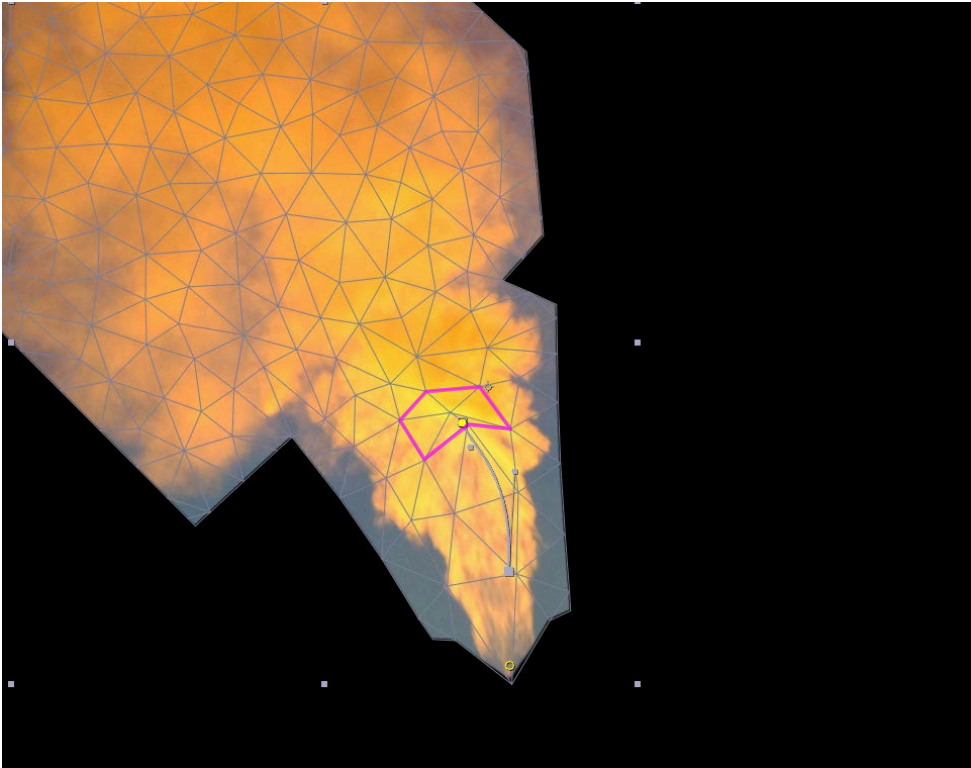
reoriented when shifted along the [link](#). Greenspun2, ¶46. Such disclosure illustrates that, even if a link's set of pixels is compressed, reoriented, or otherwise stretched when translated by a mesh algorithm, the set of pixels is still nevertheless "shifted" as claimed. Greenspun2, ¶46.

Turning next to AECS6's Puppet effect, AEM teaches that AECS6's Puppet effect similarly uses a mesh algorithm that "divides" a layer's nontransparent pixels into a "mesh of triangles" and causes "the pixels [to] move with the mesh" to "accommodate" a Deform pin's movement along a motion path. AEM, 218-221; Pet., 12-13, 21-22; Greenspun2, ¶47. Further, as can be seen in screenshots of Dr. Greenspun's animation below (remade in higher resolution and taken at half-second intervals to further aid visualization), the mesh's pixels indeed stretch, compress, and reorient as the Deform pin moves along the motion path:

Animation Time Point	Composition Frame
(0:00:00:00)	 A composition frame showing a large, irregularly shaped orange and yellow textured area on a black background. The area is overlaid with a grid of thin grey lines. A pink pentagon is drawn on the lower right portion of the textured area. A small black square is located near the center of the pink pentagon. A small yellow circle is located near the bottom right corner of the textured area. A small black square is located near the top right corner of the textured area. A small black square is located near the bottom left corner of the textured area. A small black square is located near the top left corner of the textured area.
(0:00:00:12)	 A composition frame showing a large, irregularly shaped orange and yellow textured area on a black background. The area is overlaid with a grid of thin grey lines. A pink pentagon is drawn on the lower right portion of the textured area. A small black square is located near the center of the pink pentagon. A small yellow circle is located near the bottom right corner of the textured area. A small black square is located near the top right corner of the textured area. A small black square is located near the bottom left corner of the textured area. A small black square is located near the top left corner of the textured area.

Animation Time Point	Composition Frame
(0:00:01:00)	 A composition frame showing a large, irregularly shaped orange and yellow textured area on a black background. The area is overlaid with a white wireframe grid. A pink pentagon is drawn on the lower right portion of the textured area. A small blue square is located just below the pink pentagon. A yellow circle is at the bottom right of the textured area. Several small white squares are scattered around the textured area.
(0:00:01:12)	 A composition frame showing a large, irregularly shaped orange and yellow textured area on a black background. The area is overlaid with a white wireframe grid. A pink pentagon is drawn on the lower right portion of the textured area. A small blue square is located just below the pink pentagon. A yellow circle is at the bottom right of the textured area. Several small white squares are scattered around the textured area.

Animation Time Point	Composition Frame
(0:00:02:00)	 A composition frame showing a large, irregularly shaped orange and yellow textured area on a black background. The texture resembles a wireframe mesh or a stylized flame. A pink polygonal shape is overlaid on the lower right portion of the orange area. A small blue square is positioned below the pink shape, and a yellow circle is at the bottom right of the orange area. A thin line connects the blue square to the yellow circle.
(0:00:02:12)	 A composition frame showing the same scene as the previous one, but with a slight change in the position of the pink polygonal shape and the blue square. The orange and yellow textured area remains the same, and the yellow circle is still at the bottom right. The thin line connecting the blue square to the yellow circle is still present.

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

Greenspun2, ¶48; *see also* Pet., 30-34; Greenspun2, ¶50 (providing the resulting animated GIF). As shown, pink annotations have been added outlining a set of six triangles—the pixels of which are parallel to the motion path—moving across the animation with the Deform pin. Greenspun2, ¶49. As can be seen, this initially hexagonal outline deforms as the Deform pin moves along the motion path—indeed, at (0:00:02:23) the outline has rotated and its bottom has inverted, indicating that the triangles’ pixels have reoriented and compressed together as well. Greenspun2, ¶49. Plotagraph also appears to recognize as much occurs in the screenshots provided previously in the Petition. *See* POR, 48, 14-16 & n.1; Greenspun2, ¶49.

Yet, this reorienting and compressing by a mesh algorithm—even if *more* than a pure translation of a set of pixels—is no different than what is depicted in the specification’s FIG. 4 discussed above, which, again, expressly meets the claimed “shifting” a “set of pixels.” *See* ’017 Patent, 8:28-41, 6:49-63; Greenspun2, ¶51. As such, by teaching using a mesh algorithm accordingly (*see* AEM, 218-221; Pet., 12-13, 21-22), AEM therefore teaches the claimed “shifting” of a “set of pixels.” Greenspun2, ¶51.

C. The Petition Never Asserted that Okabe Teaches “shifting.”

Plotagraph asserts that Okabe uses a “video patch[ing]” animation technique that does not “shift” pixels, and that accordingly “[o]ne reading Okabe would not look to a pixel-based shifting mechanism such as Shepard’s distortion.” POR, 36-40. Plotagraph’s argument, however, neglects the combination. The Petition never relied on Okabe as a primary reference, nor as modified by IMU. Rather, the Petition relied on the reverse: *IMU as modified by Okabe*. *E.g.*, Pet., 64; Greenspun2, ¶53.

Further, the Petition never relied upon Okabe’s alleged “video patch[ing]”-based animation. In fact, the Petition never asserted that Okabe teaches the claimed “shifting” of pixels at all. Rather, the Petition solely relied on IMU’s teachings regarding IMV6’s “Shepard’s Distortion” for this claim element. *E.g.*, Pet., 73 (“[T]he IMU-Okabe Combination... shift[s] the selected pixels in the stroke’s direction *using IMV6’s Shepard’s Distortion*.”); Greenspun2, ¶54. At most, the

Petition asserted regarding Okabe that, when conducting such *Shepard's-distortion-driven* “shifting” in the IMU-Okabe Combination—i.e., the movement of *IMV6 Shepard's Distortion control points* to specific positions over time (see Section III.A)—specifying of control point positions would have been performed visually “by drawing a corresponding ‘stroke’ from Okabe,” rather than manually. Pet., 72; Greenspun2, ¶54.

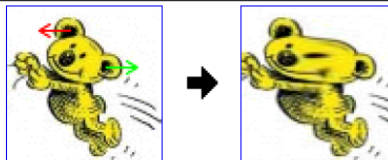
IV. THE IMU-OKABE COMBINATION SHIFTS A SET OF PIXELS “ALONG [A] NON-LINEAR PATHWAY.”

The Board “question[ed] whether the [IMU-Okabe] combination teaches the claimed ‘automatically shifting the first set of pixels along [a] non-linear pathway.’” Paper 9, 26-27; see also EX1050, 23-27. These findings are addressed below.

A. IMU Discloses the Claim Language by Teaching Incrementally Moving Shepard's Distortion Control Points Along a Curved Path

As the Petition explained, IMU teaches using IMV6's Shepard's Distortion on a koala image to generate a new, correspondingly distorted image:

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



IMU-Distorting, 59 (annotations added); Pet., 56-57, 67-68; Greenspun2, ¶56.

Specifically, as explained, IMU teaches how to place Shepard's Distortion *control*

points on specific *starting coordinates* in an image—e.g., one at (30,11) and another at (20,11) above—and how to then move the control points to *new coordinates*—e.g., (48,29) and (58,29) above, respectively—to generate a correspondingly distorted image. IMU-Distorting, 59; Pet., 56-57, 67-68; Greenspun2, ¶56.

The Petition further explained that, in view of IMU’s other teachings, a POSITA would have understood IMU’s koala example as depicting *one* increment within a potentially *multi*-increment Shepard’s Distortion animation. See Pet., 67-68 (explaining that, “[t]o animate with distortion effects,” control point movement “is done incrementally for each successive frame,” where IMU’s koala example “provides *an example of one such increment...*”); Greenspun2, ¶57.

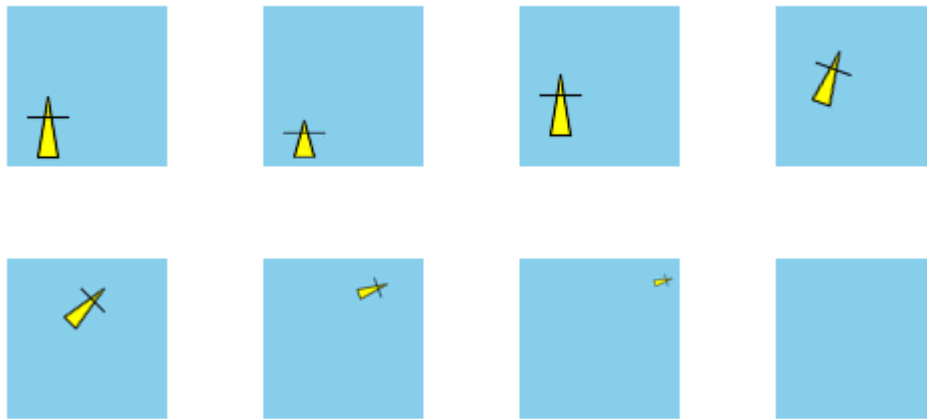
Specifically, regarding such a *multi*-increment, *control-point-based* animation, the Petition explained that IMU indeed expressly teaches the general concept of performing multi-increment, control-point-based animation in IMV6, albeit through an example using a different *but nevertheless control-point-based* distortion, “Scale-Rotate-Translate (SRT) Distortion”:

```

convert -size 80x80 xc:skyblue -fill yellow -stroke black \
-draw 'path "M 15,75 20,45 25,75 Z M 10,55 30,55"' \
spaceship.gif
convert spaceship.gif \
\(-clone 0 -distort SRT '20,75 1.0,0.6 0' \) \
\(-clone 0 -distort SRT '20,60 1 0 20,49' \) \

\(-clone 0 -distort SRT '20,60 0.9 20 27,35' \) \
\(-clone 0 -distort SRT '20,60 0.8 45 40,23' \) \
\(-clone 0 -distort SRT '20,60 0.5 70 55,15' \) \
\(-clone 0 -distort SRT '20,60 0.3 75 72,11' \) \
\(-clone 0 -distort SRT '20,60 0.1 80 100,8' \) \
-set delay 50 -loop 0 spaceship_launch.gif

```



IMU-Distorting, 16-18 (annotations added); Pet., 57 (“IMU further teaches how to incrementally apply IMV6’s effects to an image to produce a sequence of frames.... IMU provides an example of applying IMV6’s ‘SRT Distortion’ effect—*another ‘distortion’ effect that, like Shepard’s Distortion, uses ‘control points’....*”); Greenspun2, ¶58. As IMU teaches, the above example depicts a space ship being created (frame 1), a “hunker-down” step being performed (frame 2), and a “handle” point being placed on the space ship at (20,60) and incrementally moved six times to coordinates (20,49), (27,35), (40,23), (55,15), (72,11), and (100,8), which together

define a speed and direction for the control point (and thus space ship) to move across the animation—i.e., a “curved path through the sky” (frames 3-8). IMU-Distorting, 17-18; Pet., 57, 62; Greenspun2, ¶58; *see also* IMU-Distorting, 16 (explaining that the green-annotated coordinates represent the SRT transformation’s “center,” while the blue-annotated coordinates “translate the ‘center’ to these coordinates”). And as IMU explains, “[t]hese points represent *control points by which the object can then be animated* in simple terms.” IMU-Distorting, 17; Greenspun2, ¶58.

As the Petition explained, the above teaches generally creating a multi-increment, *control-point-based* animation, which—when viewed with IMU’s above-discussed teachings regarding a single Shepard’s Distortion increment⁴—equally teaches specifically creating a multi-increment, *Shepard’s-Distortion-control-point-based* animation, where a Shepard’s Distortion control point is likewise incrementally moved to multiple specific coordinates in a speed and direction defining a curved path. Pet., 61-62 (“[A] POSITA would have recognized

⁴ The Petition thus did not assert that IMU’s SRT Distortion example alone “discloses identifying a first set of pixels that lie along the non-linear pathway and shifting the first set of pixels along the non-linear pathway.” *See* EX1050, 25; Greenspun2, ¶59 n.8.

that, to animate an image in IMV6 using, e.g., Shepard’s Distortion, a user would write commands that place and incrementally move Shepard’s Distortion control points within the image to new coordinates for each successive frame, such that each control point moves across the animation at the user’s desired—and potentially non-linear—direction and speed.”); Greenspun2, ¶59; *see also* Greenspun2, ¶¶60-61 (creating one such multi-increment, Shepard’s-Distortion-control-point-based animation, and providing the resulting animated GIF and its frames).

Finally, given that Shepard’s distortion is a “*point driven*” distortion that nonetheless “shift[s]” a “set of pixels” (*see* Section III.A), the above-described incremental movement of an IMV6 Shepard’s Distortion *control point along a curved path* is thus one way—if not the only way (when using control *points*)—to perform this shifting specifically “along [a] non-linear pathway” as claimed. *See* Pet., 61-62, 80; Greenspun2, ¶62.

B. The IMU-Okabe Combination Repurposes Okabe’s Strokes to Allow the Curved Path to be Defined Visually

The Petition further explained that the IMU-Okabe Combination still achieves the “*same result*”: an IMV6 Shepard’s Distortion control point is moved to multiple, specific coordinates in a speed and direction defining a curved path. Pet., 64-65; Greenspun2, ¶63. These coordinates, however, are specified visually using Okabe’s strokes. Pet., 64-65; Greenspun2, ¶63.

Okabe does not teach that its strokes move control points—indeed, the Petition never asserted that Okabe teaches “shifting” pixels at all. *See* Section III.C; Greenspun2, ¶64. But Okabe nonetheless teaches that its strokes are a visual, less-burdensome way of defining an animation’s direction and speed. Pet., 63-64; Okabe, 1-2; Greenspun2, ¶64. And in view of these known advantages, as well as the known disadvantages of IMV6’s lack of a GUI, a POSITA would have thus been motivated to use Okabe’s stroke, not IMV6’s written commands, to visually specify a IMV6 Shepard’s Distortion control point’s speed and direction. Pet., 61-65; Greenspun2, ¶64.

Although such amounts to a repurposing of Okabe’s stroke, such would not have deterred a POSITA nor diminished any reasonable expectation of success. Greenspun2, ¶65. Rather, as the Petition explained, AEM evidences that a POSITA in fact was both motivated and successful in making such a combination, as AEM teaches specifying the speed and direction of an AECS6 Deform *pin* (like an IMV6 Shepard’s Distortion control *point*), but visually using a *motion path* (like Okabe’s *stroke*). AEM, 218-19, 187-89, 193-95; Pet., 63; Greenspun2, ¶65. And even if other implementations existed, AEM demonstrates that a POSITA would have at least found suitable implementing Okabe’s strokes in the IMU-Okabe Combination to function like AECS6’s motion paths, which alone establishes *prima facie* obviousness. *Intel Corp. v. Qualcomm Inc.*, 21 F.4th 784, 800 (Fed. Cir. 2021) (“It

is not necessary to show that a combination is ‘the best option, only that it be a suitable option.’”); Greenspun2, ¶65.

V. STROKES IN THE IMU-OKABE COMBINATION WOULD HAVE BEEN USER-VIEWABLE.

Contrary to Plotagraph’s assertions (*see* POR, 41-42), the Petition provided sufficient evidence showing that Okabe’s strokes in the IMU-Okabe Combination would have been made user-viewable, not only by citing the express illustrations of strokes in Okabe (Pet., 64-65 (citing Okabe, 7, 2-3)), but also quoting a teaching from AEM that PO conveniently omits: that “[m]otion paths are simply an alternative *visual, spatial* way of *viewing* and working with spatial properties and their keyframes...” (Pet., 65 n.11 (quoting AEM, 187)). Greenspun2, ¶66.

Given a POSITA would have implemented Okabe’s strokes in the IMU-Okabe Combination to function like AECS6’s motion paths (*see* Section IV.B), Okabe’s strokes would have thus been implemented similarly as a “visual, spatial way of viewing” a Shepards Distortion control point’s pathway (Pet., 65 n.11; AEM, 187). Greenspun2, ¶67. And indeed, as the Petition explained, implementing such a “visual, spatial way of viewing” the control point’s pathway furthers the IMU-Okabe Combination’s exact purpose: to allow visualization and simplification of the animation process. Greenspun2, ¶67. *Compare* Pet., 64 (“[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.”), with *id.*, 65 n.11 (“[A] POSITA would have nevertheless been motivated to make Okabe’s strokes user-viewable to aid users in visualizing the direction and speed of their animations, *thereby making the animation process simpler, graphical, and more intuitive.*” (citing AEM, 187)).

VI. THE IMU-OKABE COMBINATION’S MATTE PERFORMS “MASK[ING]” OF PIXELS.

The Petition explained that, in the IMU-Okabe Combination, a “matte” specifies—like in Okabe—“a region of interest in the image the user desires to animate.” Pet., 76 (citing Okabe, 1), 65 (same); Okabe, 1; Greenspun2, ¶68. By confining animation to a specified region accordingly, this matte as explained thus also specifies a reciprocal region where animation does not occur (and is thus prevented from occurring)—a “mask[ed]” region as claimed. Pet., 76 (“Inherently, the matte also specifies a region that includes pixels the user does not desire to animate.”); Greenspun2, ¶68. *Contra* POR, 50. Indeed, Plotagraph elsewhere in its POR admits as much. Greenspun2, ¶68. *Compare* POR, 50, with *id.*, 38 (noting “*movement of pixels in [Okabe’s] patches is confined to be within the discrete patches themselves*”).

VII. AECS6’S MASK/MATTE PERFORMS “MASK[ING]” OF PIXELS AND “SELECTING”/“IDENTIFYING” OF A “SET OF PIXELS.”

The Petition explained that AEM teaches applying AECS6’s “mask”/“matte” to make certain pixels of a layer transparent (and thus excluded from the Puppet

effect’s mesh) but not others, which teaches not only the claimed “mask[ing]” of the certain pixels but also, per the specification, the claimed “selecting”/“identifying” of the other pixels. *See* Pet., 39-43, 23-29. And in doing so, the Petition provided sufficient evidence showing how these other pixels would, in common instances, meet the claimed “set of pixels.”

A. AEM Teaches Applying AECS6’s Mask/Matte, Which Performs the Claimed “mask[ing].”

As the Petition explained, AEM teaches that AECS6’s Puppet effect “outlin[es],” “mesh[es],” and “move[s]” the “*nontransparent*” pixels of a layer, which is determined by the layer’s “alpha channel.” Pet., 12-13 (citing AEM, 219, 220-21); Greenspun2, ¶70. As also explained, AEM teaches how to modify a layer’s alpha channel—i.e., make certain pixels transparent and others nontransparent—by applying AECS6’s “mask”/“matte.” Pet., 10 (citing AEM, 315, 318⁵), 24 (same); Greenspun2, ¶70.

⁵ In a related IPR, the Board believed that this cited disclosure from AEM solely “pertains to ‘compositing.’” EX1050, 12. Such overlooked not only that AEM, 315, discusses *both* “[c]ompositing *and transparency*,” but also overlooked that AEM, 318, discusses only “[a]lpha channels, masks, and mattes,” not compositing. Greenspun2, ¶70 n.9.

By teaching applying a mask/matte to modify a layer's alpha channel, while also teaching that the alpha channel determines which pixels of the layer form the Puppet effect's mesh, AEM thus teaches—at least by syllogism—applying a mask/matte to modify which pixels of the layer form the Puppet effect's mesh. Greenspun2, ¶71. And when applying a mask/matte accordingly, the certain pixels made *transparent*—and thus *excluded* from (and prevented from moving with (*contra* POR, 49)) the mesh—are “mask[ed]” as claimed. Pet., 39-43; Greenspun2, ¶71.

B. AEM's Mask/Matte Disclosure Likewise Teaches the Claimed “selecting”/“identifying” of a “set of pixels.”

The Petition also explained that, when “mask[ing]” certain pixels, the opposite act of *not* “mask[ing]” other pixels qualifies, per the specification, as a way of instead “selecting”/“identifying” those other pixels as claimed. '017 Patent, 6:49-55 (“[A] first of pixels may comprise a group of pixels that intersect with a link between a starting point and an ending point *and that are not covered by a mask.*”), *quoted in* Pet., 23-24; Greenspun2, ¶72. And indeed, the Petition explained that, when applying AECS6's mask/matte to make certain pixels *transparent*, the other pixels that are oppositely made *nontransparent*—and thus included in (and moved with) the mesh—are “select[ed]”/“identif[ied]” as claimed. Greenspun2, ¶72. *Compare* Pet., 39-43, *and* Section VII.A, *with* Pet., 23-29. *Contra* EX1050, 12-13 (misunderstanding Petitioner's similar argument to be that “every pixel is ‘selected’

under Petitioner’s usage of the term ‘selecting,’” rather than only the *nontransparent* pixels (emphasis in original)).

The Petition further explained that, when applying a mask/matte as such to make only certain pixels *transparent*, the other pixels made *nontransparent* would “in common instances” meet the claimed “set of pixels.” Pet., 25. The Petition did not assert this *ipse dixit*, nor based on “happenstance” or “circular logic” (*contra* EX1050, 14); rather, the Petition provided sufficient supporting evidence, not only by providing Dr. Greenspun’s expert opinion on how a POSITA would have animated an image of “smoke rising from [a] smokestack,” but by also comparing such to Okabe, which *expressly* depicts animating a “waterfall” image *in an equivalent way*. Pet., 25-28 (comparing Greenspun, ¶102, with Okabe, 7, 2); Greenspun2, ¶73. As explained, both instances show applying a mask/matte that excludes certain (non-smoke/non-waterfall) pixels from animation, but ensures that the animation still includes—*all the while drawing motion paths/strokes through*—the other (smoke/waterfall) pixels, exemplifying what a POSITA would have recognized to be “common instances” where “select[ed]”/“identif[ied]” pixels meet the claimed “set of pixels.” Pet., 25-28; Greenspun2, ¶73.

Dated: Feb. 16, 2024

/Robert Greeson/

Robert Greeson (Reg. No. 52,966)

IPR2023-00153

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

CERTIFICATE OF WORD COUNT

Pursuant to 37 C.F.R. §42.24(c)(1)(i), the undersigned certifies that this Paper—exclusive of the table of contents, certificate of service, this certificate of word count, and appendix of exhibits—includes 5,514 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 Feb. 16, 2024, a copy of Petitioner's Reply to Patent Owner's Response was served on Patent Owner via email (by consent) to:

Lead Counsel: Shawn Hunter (Reg. No. 36,168)
shunter@schneiderwallace.com

Backup Counsel: David Walker (Reg. No. 52,334)
dwalker@schneiderwallace.com

Raymond R. Ferrera (Reg. No. 47,559)
ray.ferrera@arlaw.com

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