SIGRAD 2015
L. Kjelldahl and C. Peters (Editors)

Triangulation painting

Max Pihlström, Anders Hast, Anders Brun

Department of Information Technology, Uppsala University, Sweden

Abstract

In this paper a dynamic image representation is proposed by combining advantages of both raster and vector graphics in the triangulation. In order for a dynamic mesh to remain a triangulation, a method which maintains integrity of representation is devised. Together with techniques for synthesizing paint, the end result is a configurable scheme demonstrating potential as a viable alternative for digital painting and imaging in general.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation—Line and curve generation

1. Introduction

In image editing there are two main types of representation: raster graphics and vector graphics. In the field of digital imaging, what in recent years has commonly been researched are sophisticated methods for simulating traditional painting media [VLVR05, BWL04, DKI*12]. In these cases, crucially the representations are all built on top of a pixel or a vector graphics framework. For example, in an algorithm for watercolor developed at Adobe, the paint goes from "watery" polygon shapes to "dried" texture buffers but is still fundamentally a *composite* of raster and vector graphics [DKI*12]. In a different approach, Orzan et al. generate graphics using heat diffusion and vector curves but the underlying elements lack the appealing neighborhood characteristic of the pixel raster [OBB*13].

In this paper, a representation for image editing and paint synthesis will be presented which *combines* appealing characteristics of *both* raster and vector graphics in the triangulation. While the analogous nature between the pixel raster and the 2D triangulation has been recognized in earlier work [SP07], the focus of this paper will be to extend the triangulation as a representation for *dynamic* imaging. This will be done by deploying a *triangulation method* for maintaining integrity of representation during spatial transformation. With emphasis on robustness, this method solves the 2D analogue to the topological problem of handling multi-material surface tracking recently treated by Da et al. [DBG14]. Also, as part of paint synthesis, a method will be presented which can be seen as a 2D analogue to Pettersson's continuous tessellation of 3D meshes [Pet14]. Finally, techniques for contour blending and smoothing will be described, enabling for unique modes of digital imaging.

2. Edge preserving triangulation

2.1. The triangulation

A 2D *triangulation* is a straight-line graph subdividing the plane into triangles with no edge crossing another [dBvKOS00]. Importantly, the triangulation shares the basic geometric abstraction capabilities of vector graphics as well as the neighborhood characteristics of raster graphics. Any polygon can be triangulated [dBvKOS00, p. 46] and become a subset of some larger triangulation; also, it is the case that the triangulation subdivides the plane just like the pixel raster except with arbitrary triangles instead of identical squares. This combination of properties is the motivation for making the triangulation the representation structure of this discourse, with color being an attribute of the triangle faces. We define a *contour edge* as an edge where the two triangle faces adjacent to the edge have sufficiently discerning colors (such as precisely undetectable by the eye, determined by some distance measure). A *contour* is a path of contour edges.

2.2. The problem of movement

Moving a vertex of a triangulation generally violates the planar condition of no edge intersecting another. In order to maintain a triangulation there need to be *topological changes* [AWY04], primarily in the way of rearrangement of edges. In general this problem can be solved in any number

of ways, but for our purposes there are some specific requirements that need to be met: we maintain that the only important element is contour and also that *any* composition of contour should be a possible end state. These two aspects motivates the approach for a triangulation method taken here, called *edge preserving triangulation* (EPT).

We will devise an algorithm which analytically determines the topological changes needed to rectify the disrupted state caused by moving a vertex to a new position. The approach can be explained by imagining the vertex traveling along the line from the original position to the new position. On this travel line there will be points of *discrete events* [Meh04] where the planar condition is violated. Violations are resolved in the order they appear on the travel line by *"flipping"* [dBvKOS00] edges (see Figure 1), where at each flip, the color attributes are also reassigned so as to preserve contour. By this process, a triangulation is progressively ensured along the travel line until the new vertex position is reached. See Figure 1.

Figure 1: *An example of the EPT algorithm. The flipped edge is shown as a dashed line in the old color.*

2.3. Terminology and assumptions

The algorithm assumes a *double-connected edge list* (DCEL) data structure [dBvKOS00] where the sign of the determinant of two edges of a triangle determines the orientation and direction of traversal. A triangulation contains only positive triangles. Depending on whether the determinant is strictly positive or non-negative we will say that a triangle is positive or non-negative respectively.

We will call the set of triangles incident to some vertex the *star* [Rot88] of the vertex and a triangle belonging to the star will be called a *star triangle*. The *hull* of a star of a vertex is the union of those edges of the star triangles not incident to the vertex; the *hull edge* of a star triangle is the edge of the triangle which intersects the hull; the *hull line* is the line which extends some hull edge infinitely in both directions. In the context of two star triangles we will say that their hull is *convex*, *concave* or *flat* referring to the angle between the two hull edges of the triangles.

Let T be a triangulation of the point set P. Let the input v be a vertex of T with the position p_0, and let the input p_1 be the position where v is to be moved. We assume that p_1 is in the interior of the convex hull of P. For the time being we will also assume that $p_1 \notin P$, that is, the vertex is never moved so that it overlaps another.

Let us define the line segment

$$P : p(t) = p_0 + t(p_1 - p_0), t \in [0,1]$$

where we observe that $p(0) = p_0$ and $p(1) = p_1$. Let us also associate the position of v with $p(t)$ so that t decides the state of the mesh where v is moving along P.

2.4. Finding critical points

Moving a vertex of T generally results in one or more *planar violations*, which is to say, a star triangle being non-positive. We assert without proof that moving a vertex always results in at least one positive star triangle. Since all star triangles of v are assumed to be positive at $t = 0$, we have that if some star triangle A is non-positive at $t = 1$ there must exist some $t_e \in [0,1]$ when A is zero and the planar violation occurs. Finding t_e can be reduced to the task of determining when P intersects the hull line of A. This is expressed in the equation

$$t_e = 1 - \frac{x_{[h_b - h_a]} y_{[p_1 - h_a]} - y_{[h_b - h_a]} x_{[p_1 - h_a]}}{x_{[h_b - h_a]} y_{[p_1 - p_0]} - y_{[h_b - h_a]} x_{[p_1 - p_0]}} \quad (1)$$

where h_a and h_b are the incident vertices of the hull edge of A and x and y is the scalar components of the vector denoted in the index.

Let E be the set of tuples of all non-positive star triangles of v with the corresponding value t_e. We will call E the set of *violation events*. Let E' be the subset of E containing the violation events with the smallest value t_e, denoted by t'_e. E' may contain several violation events since these may be simultaneously occurring. We have that $p(t'_e)$ represents the mesh state when all star triangles are either positive or zero.

2.5. Resolving planar violations

In the violation event of A there are three ways the hull edge of A can be located in relation to the *half-planes* [dBvKOS00] defined by the travel line: the hull edge is on the left half-plane, on the right half-plane, or both. We will call violation events corresponding to these cases, respectively, a left event, a right event, or a middle event. These are all resolved by flipping.

2.5.1. The middle event

A planar violation of a middle event, occurring at t'_e, is resolved by flipping the intersected hull edge, resulting in two new star triangles. Any of the two triangles may be non-positive at $t = 1$. We assert that the violation events associated with such a triangle must occur at some t_e larger than t'_e.

2.5.2. The left and right events

Let A' be a triangle of some left event in E' – the right event is (mostly) symmetrical. We know that the hull edge of A' and the star triangle B' left-adjacent to A' is always non-concave. If it was concave, the hull line of B' would need to have been intersected by P at some earlier t, which contradicts the minimality of t'_e. Given that the hull of A' and B'

is convex, a flip between A and B resolves the planar violation of A; importantly, the resulting non-star triangle will be positive. We assert without proof that if, as a result, the new star triangle violates the planar condition at $t = 1$, the violation occurs at some t_e greater than t'_e.

A degenerate case occurs when the hull of A' and B' is flat. Then B' is also associated with a left event for which the planar violation needs to be resolved before the violation of A' can be resolved. B' could in turn also have a left adjacent star triangle C' for which again the hull of B' and C' is flat, and this could go on. See Figure 2. But as previously asserted we know that a star always has at least one positive triangle, which means that in the chain of flat hulls there must exist a triangle Ω' sharing a convex hull with an adjacent left triangle, which means that the planar violation of Ω' can be resolved by a flip, whereupon the entire chain can be resolved.

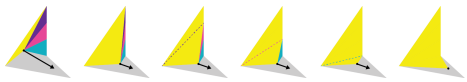

Figure 2: *Stages of a degenerate violation event.*

2.5.3. Termination

Once all planar violations of E' have been resolved, $p(t'_e)$ represents a triangulated state. We can therefore imagine setting $p_0 := p(t'_e)$ and regarding the travel line distance as a *bound function*. The algorithm terminates when E is empty.

2.6. Implementation considerations

In an implementation it cannot be assumed as we did earlier that $p_1 \notin P$. The case when $p_1 \in P$ needs to be checked for continuously and is resolved by removing and re-linking pairs of triangles in the DCEL. Vertices moved outside of the hull or on its boundary need also be taken into special consideration. Finally, the color of the triangle faces need to be reassigned as seen in Figure 1. It turns out that the reassignment logic only has to adapt for each type of violation event for it to be consistent.

2.7. Properties

The following are noteworthy properties of the EPT algorithm.

- By minimal disruption to topology, contour is well-preserved. See Figure 4.
- Any triangulation is a possible end state of the algorithm.
- Any linear motion of a vertex can be collapsed into one execution of the algorithm.
- The algorithm can be made *robust*, avoiding round-off errors potentially leading to a corrupted state of the mesh.

3. Paint synthesis

3.1. Refraction

A method for introducing geometric detail in the triangulation is the insertion of an extra vertex in the middle of contour edges that are above a certain threshold length. We will call this method *refraction*. Refraction (together with EPT) produces smooth contours, particularly during simple spatial transformations such as rotation or movements induced by strokes with a pointing device. See Figure 4.

3.2. Contour blending

To achieve contour blending we consider color exchange between two triangles adjacent to a contour edge. To avoid coloring of long-spanning triangles when blending over a contour edge a simple method is to split the triangle at one of the other two edges at some maximum depth length. See Figure 3.

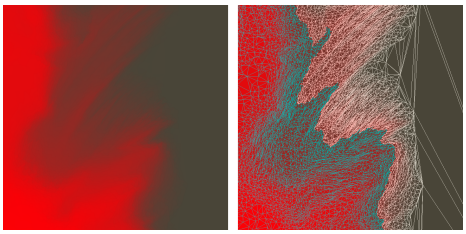

Figure 3: *Contour blending produces a mesh of triangles.*

3.3. Smoothing

It is possible to do *smoothing* on a region of the triangulation with a simple rule: for each vertex, calculate the average of the vectors of all contour edges and apply it as a translation vector for the vertex. When iteratively applying this rule, curve geometry is accentuated. See Figure 4.

Figure 4: *A graphic and a smoothed version of it.*

4. Discussion

4.1. Performance

Proof of concept software was developed to produce the results presented in this paper and was tested on a laptop

computer with a 1.86GHz Intel Core2Duo and an Intel integrated graphics card. Graphics rendering was performed on the GPU using OpenGL while other methods such as EPT were performed on the CPU in C++ code. With reasonable brush sizes, real time speed was possible with operations such as contour blending and smoothing with the smallest triangles being roughly the size of the pixels of the screen raster. Tests suggest that the EPT algorithm is the main performance bottleneck, where the computation times are proportional to the number of vertices processed and the length of their movement vectors. Since the algorithm relies heavily on vector calculations potentially a lot of performance gains can be made with parallel processing on the GPU.

4.2. Characteristics

The markedly special property of the triangulation as structure for image representation is that of contours and their spatial relationship being *integral to the structure*. This should be contrasted to raster graphics where the geometry is approximated by a grid, and vector graphics where the geometry is free but elements lack spatial relationship in terms of neighborhood. The implications of this type of integral representation is epitomized by the smoothing rule: structures can be accentuated because the total geometry is defined immediately in the triangulation with minimal compromise and redundancy. The possibilities of such rules reach beyond what has been presented here.

4.3. Future work

Currently EPT is used in research on decimating triangulation meshes in conjunction with edge-preserving filtering of images of hand-written text. The resulting meshes capture semantic content related to edge structure. See Figure 5.

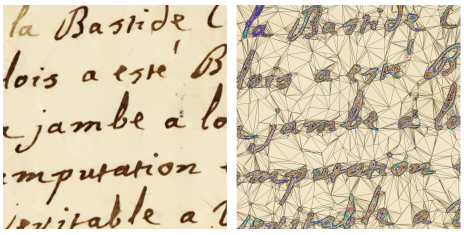

Figure 5: *EPT can be used in mesh decimation.*

5. Conclusion

The triangulation proves viable as a structure for dynamic representation by combining the idea of the neighborhood of raster graphics and the idea of geometric abstraction of vector graphics. In order to maintain a triangulation during transformations, a triangulation method is needed. The edge preserving triangulation (EPT) algorithm analytically determines required topological changes that preserve contour. On top of this, methods for producing geometric detail and blending of contour can be applied along rules such as smoothing for accentuating certain aspects of geometry.

The software performs in real time for image transformations of high detail and also shows potential for further improvements in this regard. Integral representation of total geometry shows promise in providing new types of image transformations. EPT has also recently shown potential as a tool for image processing in decimating triangulation meshes.

6. Acknowledgements

I acknowledge the financial support for preparing and presenting this research from the project Searching and datamining in Large Collections of Historical Handwritten Documents (Vetenskapsrådet, Dnr 2012-5743), which is a part of the From Quill to Bytes (q2b) effort at Uppsala University.

References

[AWY04] AGARWAL P. K., WANG Y., YU H.: A 2d kinetic triangulation with near-quadratic topological changes. In *Proceedings of the twentieth annual symposium on Computational geometry* (2004), ACM, pp. 180–189. 1

[BWL04] BAXTER W., WENDT J., LIN M. C.: Impasto: a realistic, interactive model for paint. In *Proceedings of the 3rd international symposium on Non-photorealistic animation and rendering* (2004), ACM, pp. 45–148. 1

[DBG14] DA, BATTY, GRINSPUN: Multimaterial mesh-based surface tracking. *ACM Trans. on Graphics* (2014). 1

[dBvKOS00] DE BERG M., VAN KREVELD M., OVERMARS M., SCHWARZKOPF O.: *Computational Geometry: Algorithms and Applications*, second ed. Springer-Verlag, 2000. 1, 2

[DKI*12] DIVERDI S., KRISHNASWAMY A., ITO D., ET AL.: Painting with polygons: A procedural watercolor engine. 1

[Meh04] MEHTA D. P.: *Handbook of data structures and applications*. CRC Press, 2004. 2

[OBB*13] ORZAN A., BOUSSEAU A., BARLA P., WINNEMÖLLER H., THOLLOT J., SALESIN D.: Diffusion curves: a vector representation for smooth-shaded images. *Communications of the ACM 56*, 7 (2013), 101–108. 1

[Pet14] PETTERSSON T.: Sculptris. *pixologic.com/sculptris* (2014). 1

[Rot88] ROTMAN J. J.: *An introduction to algebraic topology*, vol. 119. Springer, 1988. 2

[SP07] SWAMINARAYAN S., PRASAD L.: Ravegrid: Raster to vector graphics for image data. In *SVG Open* (2007). 1

[VLVR05] VAN LAERHOVEN T., VAN REETH F.: Real-time simulation of watery paint. *Computer Animation and Virtual Worlds 16*, 3-4 (2005), 429–439. 1