Image Stained Glass using Voronoi Diagrams

Michael Gorbach
mgorbac1@cs.swarthmore.edu

Abstract

The geometrical concept of the Voronoi diagram was used to create an image filter providing a "stained glass" or mosaic effect on an image. The Voronoi diagram was calculated by exploiting its dual relationship with the Delaunay triangulation, which was in turn calculated using a randomized incremental algorithm and stored in a DCEL. Various methods were tried for selecting the points, including sampling from a distribution built using edge detection. Sampling using edge detection distributions was shown to provide results significantly better than uniform random sampling.

1 Introduction

Voronoi diagrams, when calculated on some set of N points in the 2d plane, segment the space into regions surrounding every point. The polygonal regions are such that, within a region surrounding some point p_0, the point p_0 is closer to any point p in that region than any other of the N points included in the Voronoi diagram. The mapping between points in the plane and surrounding regions is one to one.

This information has many uses, but one of the most obvious is processing an image for an artistic effect. The representation created by shading a Voronoi diagram on N points in the image plane with colors from each sample point creates a "stained glass" or mosaic version of the image. One of the key problems here is effective selection of the point set P for the Voronoi diagram.

2 Theory

2.1 Voronoi Diagrams

First, it is appropriate to examine the algorithms involved in the efficient calculation of a Voronoi diagram on a set of N points. The goal is a polygonal map of the plane consisting of a set of polygons surrounding the N points. The polygon surrounding a point p covers the area for which p is the closest of the N points.

Given two points p and p', we can create a Voronoi diagram by drawing a line perpendicular to the line pp', intersecting pp' at its midpoint. A Voronoi diagram with more points includes many such lines, meaning that each polygon has straight edges consisting of line segments which are sections of such perpendiculars.

Voronoi diagrams can be calculated directly, using for example the beach line algorithm from the work (Fortune, 1986). It is often simpler, however, to take advantage of the close relationship that exists between the structure of the Voronoi diagram, and that of the Delaunay triangulation. (Guibas et al., 1990)

2.2 Delaunay Triangulation

A triangulation of some point set P is a planar subdivision such that every polygon is a triangle (except for the unbounded face), and the vertices are points in P. A triangulation exists for every point set P, as any bounded face can be split up into triangles, and the unbounded face is simply the complement of the convex hull for P. There are, of course, many different triangulations on any one set of points P. Given two triangles bordered by a common edge, it is always possible to "flip" this edge such that it connects the remaining two points, assuming the quadrilateral in question is convex. (Berg, 2000)

In a triangulation, it is undesirable to have small (sharp) angles. There is one triangula-

tion, called the Delaunay triangulation, which maximizes the minimum angle and therefore is the "best" triangulation. One simple way to find this triangulation is to take an arbitrary triangulation and flip all illegal edges. Here, an illegal edge is defined as an edge for which flipping will improve the triangulation: the ordered set of angles after flipping will be lexicographically greater than the set before flipping. (Berg, 2000) Of course, an edge can only be flipped in the case of a convex quadralateral. An example of such an edge flip is shown on fig. 1.

It is not necessary to calculate all the angles to determine the legality of an edge. Consider two triangles *abc* and *dbc* that share an edge *cb*. Let C be the circle defined by *abc*. The edge ij is illegal if an only if the point d lines inside C. A proof can be found in (Berg, 2000).

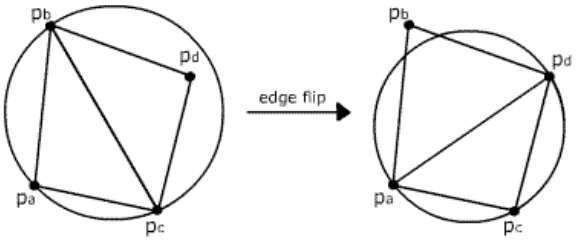

Figure 1: An edge flip during the process of creating a Delaunay triangulation. http://www.cescg.org/CESCG-2004/web/Domiter-Vid/

A Delaunay triangulation can be constructed using an incremental algorithm based on the above. (Berg, 2000) Randomizing the point set P, add the points sequentially to the triangulation. Each time a point is added, start by triangulating the face containing the new point, and then legalize edges recursively until all edges in the triangulation are legal. Thus, the algorithm maintains a correct Delaunay triangulation of the currently included points as an invariant.

2.3 Dual Transformation

The last step in constructing a Voronoi diagram on P is to convert the Delaunay triangulation on P into a Voronoi diagram. The structures are related through duality.

A face in the Delaunay triangulation corresponds to a vertex of the Voronoi diagram, such that the location of the Voronoi vertex is the center of circumcircle for the (triangular) Delaunay face. A vertex in the Delaunay triangulation corresponds to a face in the Voronoi diagram. This Voronoi face surrounds the Delaunay vertex and represents the Voronoi cell for this vertex. An edge in the Delaunay triangulation corresponds to a perpendicular edge in the Voronoi diagram. Two Voronoi vertices are connected if an only if the corresponding Delaunay faces are adjacent. Figure 2 shows a Delaunay triangulation and the corresponding Voronoi diagram.

2.4 Point Sampling

One of the primary difficulties in using Voronoi diagrams to create stained glass effects is the selection of the point set P on which to build the diagram. Badly chosen points create a result that captures none of the features in the original image. In this project, the implementation of fully automated, intelligent point selection was a key goal. Point selection can be done, or adjusted, manually, however the need for such intervention limits to applicability of the processing, and so was not studied here.

2.4.1 Naive Approaches

The simplest method for point selection uses a random sample of N points, distributed uniformly within the boundaries of the image. Such a method is of course very simple to implement, and also has an advantage that follows from its uniformity. Because the distribution is uniform, the sizes of all the Voronoi cells will be relatively small, and thus a badly-colored Voronoi cell can have only a limited size. The obvious issue with such a method is that it fails to account for the global features of an image. Random point selection, in practice, results in significantly distorted representations, especially in high-detail regions of the image, even at large N.

A grid-based point selection approach is another simple alternative. It has the advantage of highly uniform cell size, similar to that seen in a real mosaic. Like uniform random sampling, it

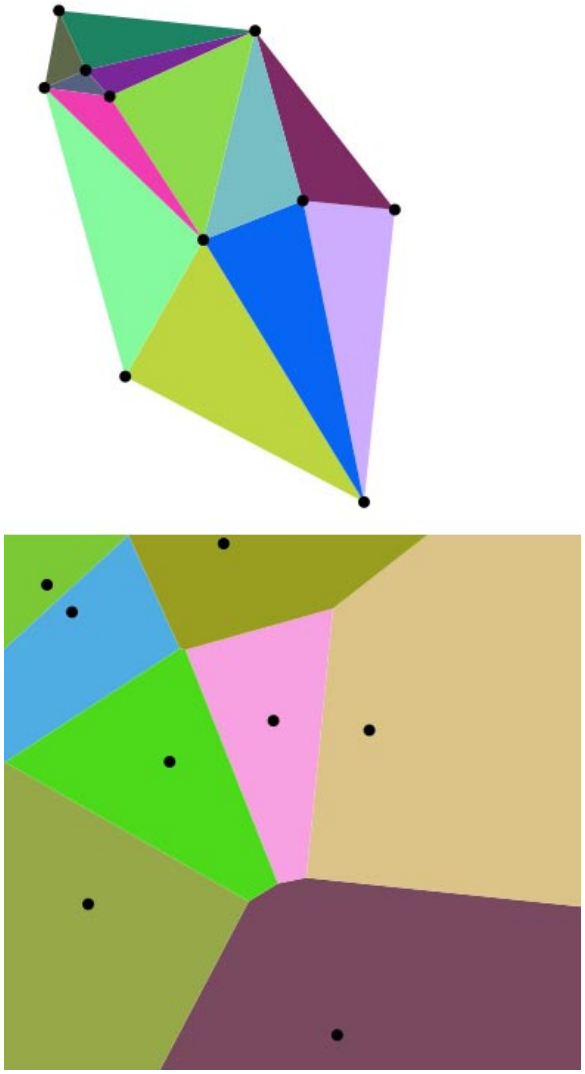

Figure 2: Example of a Delaunay triangulation (top) and corresponding Voronoi diagram (bottom). The colors have no meaning.

suffers from a failure to account for the image's important features. Good representations can only be obtained with fairly large N values.

2.4.2 Distribution Sampling

The earlier discussion of uniform sampling can be generalized to an arbitrary probability distribution on the 2d plane of the image. The question, then, is what distribution $P_s(x, y)$ on the image pixels would, when sampled from for a total of N points, produce the best representation of the image. One quantitative criterion for $P_s(x, y)$ is the error between the colored, N-point Voronoi representation with sampling from $P_s(x, y)$ and the real image. Such a value, however, does not necessarily reflect an aesthetic judgment of the colored Voronoi mosaic.

Edge detection appears as an efficient way to obtain a good $P_s(x, y)$. Good mosaics are created when Voronoi cell edges fall on edges of the image. In order for this to happen, Voronoi points must be located at equal distances from an edge line. It is undesirable for sampled points to fall on image edges themselves, as then the Voronoi cell surrounding that point is likely to be badly colored, in a way that is not expressive of key image features. The goal then, is to receive a distribution that has symmetric and significant values around edges (leading to points likely sampled there), and low values directly on edges. Here, symmetric means that values are equal at equal distances along a perpendicular to the edge.

Such a distribution can be achieved using basic edge detection and blur filters. Specifically, it is effective to use a distribution of the form $P_s(x, y) = P_{blur}(x, y) - P_{sharp}(x, y)$. Here, $P_{sharp}(x, y)$ comes from an sharp, or only very slightly blurred, black and white edge detection image. $P_{blur}(x, y)$ comes from an image processed with the same edge detection filter, followed by a significant (on the order of 5 pixels) gaussian blur. The subtracted distribution has low (dark) values immediately on the edges, and higher (lighter) values farther out from the edges. Due to the Gaussian blur, the lighter values decrease in intensity with distance from the edge. Example distributions are presented on

fig. 3. Note that the distribution along both sides of an edge is symmetric, which is good for Voronoi point selection.

2.4.3 Related Work

In addition to considering the distribution $P_s(x, y)$, it is also possible to look instead at the representation error discussed earlier. This was implemented in the work (Dobashi et al., 2002). The authors started with a simple set of points leading to a hexagonal Voronoi diagram across the image. They then adjust the locations of the Voronoi points to decrease the error (calculated as difference in colors per pixel) between the mosaic and real image representation. In the first phase, the entire set of points is moved in batch, with each point moving somewhere in its surrounding 8 pixels in such a way as to decrease the error. This process continues until changes in error are below a threshold. The second phase implements finer adjustment where each site is moved individually within its 8 pixels and the error is recalculated each time.

While this approach appears to be effective, it requires significant computational power even with approximations, and the inclusion of manual adjustments in the paper makes it difficult to judge the effectiveness of such a method for purely autoamted processing.

3 Methodology and Implementation

3.1 The Doubly-Connected Edge List

The most important component in an implementation of the above algorithms is the data structure used to represent the planar subdivision, whether it be the Delaunay triangulation or the Voronoi diagram. This data structure must support several operations in a performant way. It needs to allow fast adding of points into an existing triangulation structure, flipping of any particular edge, and traversal of a face to find its boundary edges.

The most common data structure used to meet the above requirements is called a doubly connected edge list (DCEL) (Muller and Preparata, 1977). The structure contains several types of records: faces, edges, and vertices.

Figure 3: Examples of sampling distributions created using edge detection: $P_{sharp}(x, y)$ (top), $P_{blur}(x, y)$ (middle), and $P_s(x, y)$ (bottom). Edge detection and blur were implemented using Apple Inc.'s Core Image processing filters.

Edges are represented as half edges, storing a pointer to their twin, adjacent face, and doubly linked list pointers allowing traversal of face boundaries. A face record simply contains a pointer to one half-edge along the face's outer boundary (if it exists), and a set containing one edge along every "hole" inside the face. The Delaunay triangulation was constructed in a DCEL, and then the Delaunay DCEL was transformed into its dual, representing the Voronoi diagram.

The DCEL structure described here meets all the requirements for storing a planar subdivision. However, given a point, the structure does not provide a fast way to locate the face containing a point. For this purpose, an additional DAG (Directed Acyclic Graph) data structure is layered on top of the DCEL.

3.2 DAG For Point Location

A Directed Acyclic Graph was constructed to provide fast point location during triangulation. This point location was used during the first triangulation step, where it is necessary to find the face containing the point being added. The DAG algorithm used was described in (Berg, 2000).

The leaf nodes of the DAG correspond to the current triangulation. The other nodes correspond to previous triangles that existed earlier during the incremental triangulation process. When a point p is added, causing a split of the face f, the leaf node representing f receives 3 children for the newly created triangles. The DAG is also updated on edge flips, leading to situations where a leaf node has more than 1 pointer leading to it.

Using such a DAG structure, a point can be located by starting from the root and navigating down the children, checking for containment in the process.

4 Results and Discussion

The figures below present the results of constructing mosaic representations for a test image.

The butterfly image was chosen as it is similar to example images used in (Dobashi et al., 2002). It is extremely difficult to constructed automated mosaic representations on images with many human faces.

Looking at the images on figs 4 and 5, we can see that the representation unsurprisingly improves with increasing N. It is clear that the images generated using edge-detection distribution based sampling retain significantly more of the key features, and also have a far cleaner appearance. This is due to points being sampled from a distribution symmetric around the edges, leading to Voronoi cell edges matching up with image edges. The improvement using edge-detection distribution based sampling is particularly evident with lower N values.

Significant artifacts still exist with edge-detection based sampling under lower point counts. This is because the correct distribution does not guarantee a good set of sampled points with low N, meaning that there are or empty areas in the points. One approach to remedy this would be to adjust the distribution during the sampling process, subtracting discreet, 2d Gaussians from the distribution around each point as it is sampled. This would help prevent clusters of points and empty areas, improving uniformity with low N.

Ideally, it would be effective to add equidistant points in pairs, one on each side of an edge. Doing this, however, requires knowledge of the edges as vector paths instead of as lighter pixels in an image. Such an approach would not require sampling, and would probably work significantly better than a distribution-based approach. It does, however, require a very different kind of processing.

5 Conclusion

A stained glass / mosaic filter was successfully implemented based on Voronoi diagrams. Several solutions to the problem of sampling points were compared. While the solution by (Dobashi et al., 2002) provides good results, it requires both significant processing power and manual adjustment. A simple, fully automated sampling method was proposed based on subtraction of blurred distributions obtained using edge de-

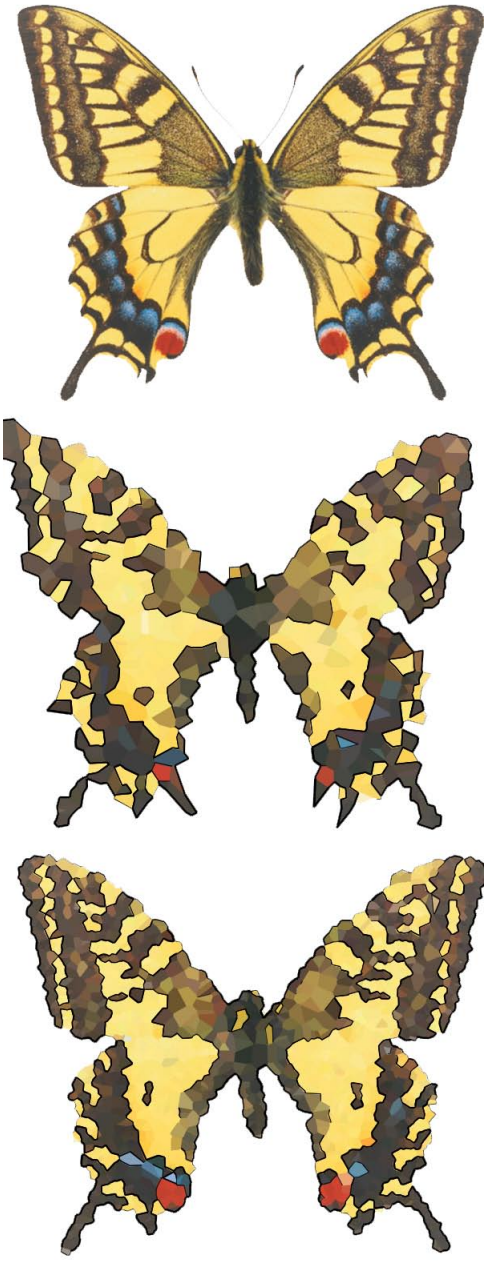

Figure 4: An example image (top), rendered using distribution-based point selection. The distribution was created by subtracting two blurred edge detection distributions, with a blur of 5.0 and a blur of 1.0. The middle image has $N = 1000$ points sampled, and the lower image has $N = 5000$.

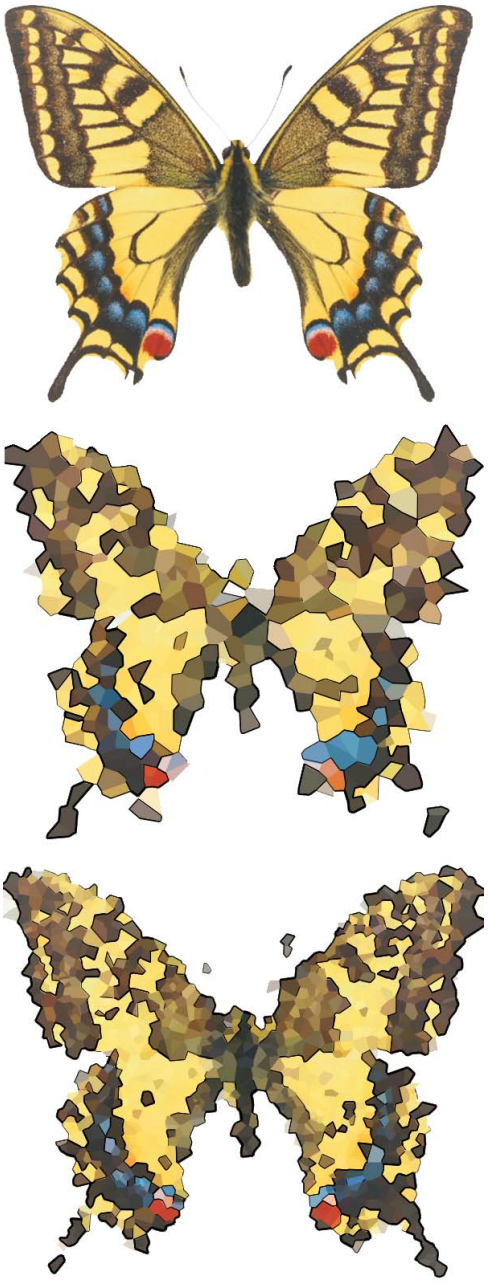

Figure 5: An example image (top), rendered using uniformly random point selection. The middle image has $N = 1000$ points sampled, and the lower image has $N = 5000$.

tection. Results presented from application of this method, shown on figs. 4 and 5, are both significantly cleaner than the mosaics created using random sampling, and more reflective of key image features.

References

Mark de Berg. 2000. *Computational geometry: algorithms and applications*. Springer, Berlin, 2nd rev. ed edition.

Y. Dobashi, T. Haga, H. Johan, and T. Nishita. 2002. A method for creating mosaic images using voronoi diagrams. In *Proc. EUROGRAPHICS 2002 Short Presentations*, pages 341–348.

S Fortune. 1986. A sweepline algorithm for voronoi diagrams. In *SCG '86: Proceedings of the second annual symposium on Computational geometry*, pages 313–322, New York, NY, USA. ACM.

Leonidas J. Guibas, Donald E. Knuth, and Micha Sharir. 1990. Randomized incremental construction of delaunay and voronoi diagrams. In *Proceedings of the seventeenth international colloquium on Automata, languages and programming*, pages 414–431, New York, NY, USA. Springer-Verlag New York, Inc.

N. E. Muller and F. P. Preparata. 1977. Finding the intersection of two convex polyhedra. Technical Report ADA056889, U. Illinois at Urbana Champaign, Coordinated Science Lab, October.