Weighted Voronoi Stippling
Methods and Applications in Computational Geometry

Franco R. C. Carlos - UW Tacoma, Mathematics 2023
Supervision: Dr. Ruth Vanderpool

Abstract

Stippling is an artistic technique in which images are illustrated and shaded using only points. No lines are used. The goal of this paper is to understand the requisite mathematics in computer generated stipple art, in an example using Lloyd's Algorithm. In order to understand Lloyd's Algorithm, its component concepts are explained, which include Voronoi Diagrams, density, and centroids. After understanding the process, the mathematical concepts are demonstrated in a computer program implemented in Javascript.

Contents

1 Introduction - Stipple Art

Stippling is an artistic technique in which images are illustrated and shaded using only points. No lines are used. As parts of the image get darker, more dots are necessary, and conversely lighter areas of the image have less dots. We will summarize and explain the challenges facing an artist stippling by hand, and contrast how this may be done algorithmically with a computer.

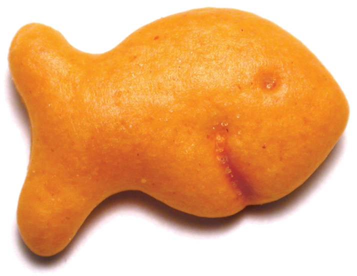

Figure 1: Our reference image.

Stippling by hand is very much a subjective practice. Parts of an image can only be lighter or darker in relation to other parts, so properly discerning between these regions is an astute, learned ability. Furthermore, stippling is usually done with respect to a reference image, and is a very unintuitive process when started from scratch. Without being able to outline or roughly sketch an image, each point must be placed deliberately, keeping in mind how it will appear in relation to the others, and with the final product always in mind.

There are many ways a computer's approach to stippling contrasts significantly to the artist's. One such process to computer generate stipple art is known as Lloyd's Relaxation, also known as Lloyd's Algorithm. As opposed to a creative approach, it is a methodical process, necessitating an algorithm. Points are not placed, through pen on a paper, but are generated randomly, in a scattered existence. As opposed to being drawn into their final position, points must be moved to it. As an instruction set, this movement must be defined as

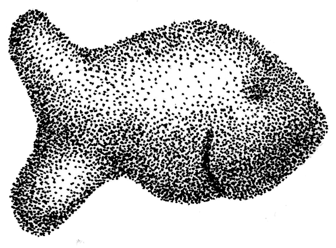

Figure 2: Hand drawn stippling.

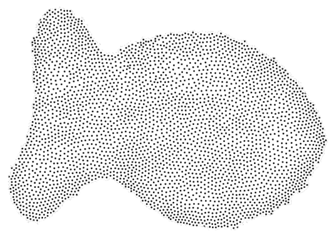

Figure 3: Computer generated stippling.

a set of rules, uniform for every point and every step in the process. Prior to movement, the dots need the relative spatial understanding of the dots around it. Finally, some aspect of the image must be quantified for the dots to have a reference for their movement. Lloyd's Relaxation is the process that allows the points to move in an incremental and uniform manner, converging to a picture.

Before fully understanding Lloyd's Relaxation, several underlying concepts must be first visited. Voronoi diagrams introduced in section 2 will first allow us to establish the spatial relations between our points. Once these relations have been defined, the concept of visual weight and density from section 3 can be integrated into our plane, as expressed by the regional color intensity of our image. When regions made by Voronoi diagrams have varying levels of color intensity, the weighted centroid, or center of color per region, may be calculated from section 4. Each point will move towards its own calculated centroid. The points will "move" according to the rules of Lloyd's Relaxation, in section 5.

2 Voronoi Diagrams

The first important distinction between stippling by hand and stippling with Lloyd's Algorithm is the initial placement of the stipple points. For an artist, each dot is placed onto the paper by hand, one at a time. Conversely, for a computer, all points begin already on the plane, and must be rearranged.

In order for the points to be arranged, there must be some form of reference information such that the computer has an understanding of the relative location of each point. This may determine the extent of each points' movement in a given iteration. Voronoi diagrams allow for such a spatial understanding.

Let $\{p_i\}$ be points on a plane, say we wanted to partition the plane into regions closest to p_i. This can be done through a Voronoi diagram.

Definition 1. *A **Voronoi Diagram** for a set of points P partitions a plane such that the ith region is defined as all points in the plane closest to $p_i \in P$. For each p_i, there exists exactly one Voronoi region. [1]*

Definition 2. *A **Voronoi Polygon** encloses its point p_i and is defined with intersecting half planes. The half planes for p_i are generated by the perpendicular bisectors to $\overline{p_i p_j}$ where p_j is a point from P that is the closest to p_i. [1]*

Definition 3. *Voronoi Polygons intersect at **Voronoi Vertices**, whose sides are known as **Voronoi Edges**. [1]*

Method to Generating a Voronoi Diagram

1. We begin with the arbitrary points on our plane.

2. For a single point p_0, identify each of its nearest neighbors p_i. Extend a line k_i to each nearest neighbor.

3. Identify the midpoints m_i of each k_i. Extend new lines l_i perpendicular to each k_i.

   In doing so, we begin to outline the half planes that form the Voronoi polygons. We will be able to prove that these perpendicular bisector lines become the half planes of our Voronoi polygon, done **Theorem 1.1**.

4. Any line l_i stops wherever it intersects with another line l_j.

   When these half-plane intersections meet, they enclose a boundary region that separates the plane into the space closer to p_0, and the space closer to some other point not p_0.

Voronoi Diagrams of Two or Three Points

We may begin by examining two simple cases, assessing sets of only two or three points initially. This allows the steps of Voronoi diagram generation to be better understood.

In essence, the generation of a Voronoi diagram involves extending a line through the perpendicular bisector between two points. With just two points, a complete Voronoi Diagram needs only a single line. We can prove that the space within Voronoi polygons is closest to the point enclosed by that Voronoi polygon.

Theorem 2.1. *A Voronoi diagram for two points p_1 and p_2 includes only their perpendicular bisector. Any point p not on the Voronoi edge and in the p_1 half plane of this diagram is closer to p_1. [2]*

Proof. We begin by establishing our points p_1 and p_2, and extending a line l on their perpendicular bisector. We identify the midpoint between p_1 and p_2 as m. Suppose q is an arbitrary point on the plane. We identify the line perpendicular to $\overleftrightarrow{p_1p_2}$ passing through an arbitrary point q as k. Let b be the intersection of $\overleftrightarrow{p_1p_2}$ and k as seen on Figure 4.

Without loss of generality let q exist on the p_1 half-plane. There are two cases to assess: If b lies on $\overline{p_1p_2}$, or if b does not lie on $\overline{p_1p_2}$.

In the first case, when b lies on $\overline{p_1p_2}$, we may identify two right triangles: $\triangle(p_1qb)$ and $\triangle(p_2qb)$. These two right triangles share the side $\overline{qb}$, as illustrated in Figure 4 on the left. We may say that $\overline{p_1m} = \overline{p_1b} + \overline{bm}$, $\overline{p_2b} = \overline{p_2m} + \overline{mb}$ and $\overline{p_1m} = \overline{p_2m}$. From this, we know that the line segment $\overline{p_1b}$ is shorter than $\overline{p_2b}$.

The Pythagorean Theorem tells us that in a right triangle, the squared length of a hypotenuse is equal to the sum of the squares of its shorter sides. If these two triangles share the side $\overline{qb}$, and the side $\overline{p_1b}$ of triangle $\triangle(p_1qb)$ is shorter than the side $\overline{p_2b}$ of triangle $\triangle(p_2qb)$, then the hypotenuse of $\triangle(p_1qb)$ is shorter than the hypotenuse of $\triangle(p_2qb)$. Since the hypotenuse is the distance between the given points, p_1 is closer to q than p_2.

In the second case, the logic remains the same but differs in applying the Pythagorean Theorem. Recall that b does not lie on $\overline{p_1p_2}$, but remains on the p_1 half plane. We can say that $\overline{bp_2} = \overline{bp_1} + \overline{p_1p_2}$. From this, we know that the line

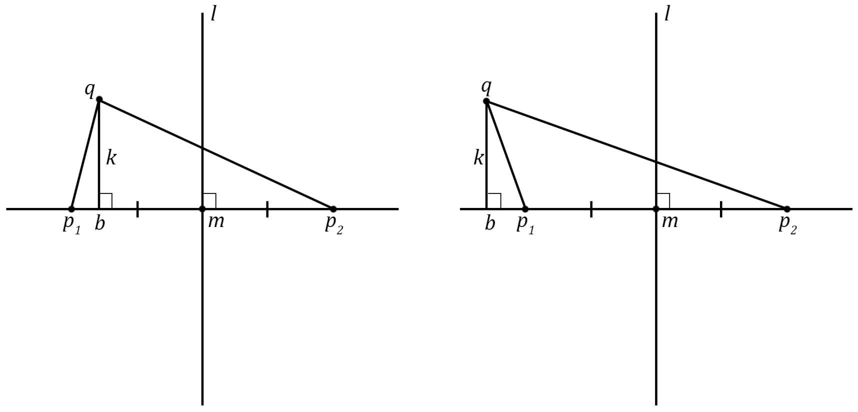

Figure 4: Voronoi diagram generation, for case 1 (left) and case 2 (right).

segment $\overline{p_1b}$ is shorter than $\overline{p_2b}$. If these two triangles share the side $\overline{qb}$, and the side $\overline{p_1b}$ of triangle $\triangle(p_1qb)$ is shorter than the side $\overline{p_2b}$ of triangle $\triangle(p_2qb)$, then the hypotenuse of $\triangle(p_1qb)$ is shorter than the hypotenuse of $\triangle(p_2qb)$. Since the hypotenuse is the distance between the given points, p_1 is closer to q than p_2. □

This proof demonstrates a two point Voronoi diagram's construction. As there are only two points, there is only one Voronoi edge, no Voronoi vertices, and as such no Voronoi polygons. With the introduction of a third point, we introduce the Voronoi polygon, and the diagram begins to take shape.

Theorem 2.2. *The perpendicular bisectors of three noncollinear points intersect at a singular point z. This point z is the circumcenter of the three points, and forms a Voronoi vertex. [2]*

Proof. We begin by considering three noncollinear arbitrary points, p_1, p_2, and p_3. The perpendicular bisector between p_1 and p_2 is defined as the first Voronoi edge v_1. The perpendicular bisector between p_2 and p_3 is defined as the second Voronoi edge v_2. The perpendicular bisector between p_1 and p_3 is defined as the third Voronoi edge v_3.

We must demonstrate that the three perpendicular bisectors must intersect at a singular point. By definition, any point on a perpendicular bisector is equidistant from the two points it bisects. Also by definition, two lines should only intersect at a single point. Say that only two Voronoi edges, v_1 and v_2, intersect. This means that v_1 is equidistant to p_1 and p_2, and at the same time v_2 is equidistant from p_2 and p_3. Say that v_3 does not intersect v_1 and v_2. This must mean that v_3 was not equidistant from p_1 and p_3. Either p_1 or p_3 therefore must move to fulfill the definition of a bisector for v_3. But we have reached a contradiction, as p_1 and p_2 were established as already equidistant from each

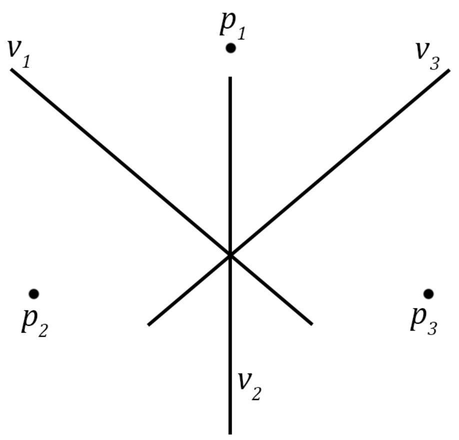

Figure 5: Theorem 1.2

other. If either moves, then the previous assumptions establishing v_1 and v_2 would be false. Therefore, v_1, v_2, and v_3 must intersect at a singular point.

Between three points, As v_1, v_2, and v_3 intersect at a singular point, known the Voronoi vertex i, this point is equidistant from p_1, p_2, and p_3 simultaneously, and reveals the circumcenter of this Voronoi polygon. □

When a Voronoi Diagram is drawn for three points, they have three Voronoi edges, and therefore a singular Voronoi vertex. This also demonstrates that we may compute the circumcenter for a given Voronoi polygon.

Corollary 2.2.1. *Three noncollinear Voronoi points containing a Voronoi vertex v, form a circle containing no Voronoi points. [2]*

Proof. We begin by considering three noncollinear points, p_1, p_2, and p_3, and a circle C that passes through them. We will demonstrate that no point is inside the circle C. The proof for **Theorem 1.2** demonstrated to us that the circumcenter of these three points, and therefore the circle, is at the Voronoi edge, which we can generate. We will then introduce a fourth noncollinear point to reach a contradiction. Suppose there was a point p_4 inside the circle. If p_4 was inside the circle, then it would be closer to v than any of the three points. By definition, the vertex v should be closest to and equidistant from each point generating its circle C. Thus, if v_4 was inside C, it would violate the definition of a Voronoi vertex. As such, there cannot exist any points inside a circle C made by points surrounding a Voronoi vertex v. □

Looking at the simple cases of two and three points delineates the importance of Voronoi diagrams: They partition the plane into polygons that show which point each area of the plane is closest to. With more than just three points, these perpendicular bisectors can intersect others at more than one location. If a region can be completely enclosed within Voronoi edges, it creates a Voronoi polygon, enclosing a Voronoi region.

Diagrams of Many Points

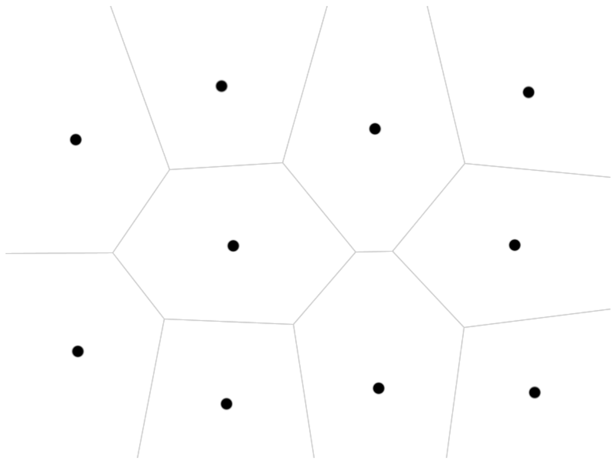

Figure 6: An example of a Voronoi Diagram for 10 points.

Moving on from cases of two or three points to many points, the process would remain the same. For each of a point's nearest neighbor, identify the perpendicular bisectors. Extend them, until they intersect with another perpendicular bisector.

The focal property of Voronoi Diagrams is how the plane is partitioned into regions closest to the point it encloses. Having taken the time to prove this property, we can then understand how it plays a vital role in stipple art. Voronoi polygons impose limits on the extent of movement for each point within a given iteration. To determine the direction of this movement, we transition to understanding visual density within computer images.

3 Density

In stippling, in order to contrast the lighter and darker areas of an image, an artist would have more dots present in darker areas of the image. In other words, the dots should be more dense.

An example of shading in stippling can be seen in Figure 7. In depicting the sphere in Figure 7, the reader's eye can deduce that the light source is placed above the object. Additionally, the denser dots below the sphere add shadow and provide additional depth.

Somehow, this visual property must be captured in a computer image's data. For the stippling algorithm we will be using, an image's RGBA data is used.

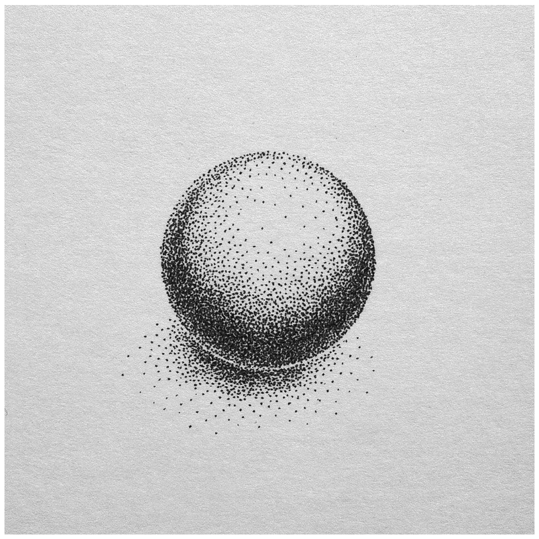

Figure 7: Shading as applied in stippling.

RGBA stands for Red, Green, Blue, and Opacity (A). The RGBA value is just one form of capturing image color data in computer graphics. There are four values captured, representing Red, Blue, Green, and Opaque channels. Each color channel ranges in value from 0, with no value of that color, to 255, with full value of that color. Opaqueness differs in ranging from the value 0, being fully transparent, to 1, being fully opaque.

Each pixel in a computer image has unique RGBA values. We can therefore consider a computer image a plane of pixels of varying RGBA values. Each pixel has reference information for the color intensity of the image at its location. Some areas of an image are more red, for example. This qualitative property of an image is quantifiably captured by identifying the regions of pixels where greater values of red are clustered. This would follow for how blue or green areas of an image is, or even how opaqe an area is to begin with.

This proposes that there are non-uniform regions of color intensity for images. If we attribute color intensity to the weight (or mass) of an image, then there should be a center of mass for the image. This center of mass, or centroid, can be for any color or the opacity itself. In the next section, we will go over how to calculate the centroid in a general context. Later, this will be used to calculate the centroids of color intensity of an image, which is the foundation for Lloyd's Relaxation.

4 Weighted Centroids

The final step in understanding Lloyd's Relaxation is to understand how our computer will use the RGBA values. The idea is to compute the coordinates for the RGBA centroid of each Voronoi Polygon. Each Voronoi Polygon, after all, is imposed onto our computer-generated stipple art canvas. The computed centroid of each Voronoi Polygon is the location that each point must move to in a given Lloyd's Relaxation iteration. We will go over how the centroid itself is computed for each Voronoi Polygon.

Consider an image's pixels as squares, forming a plane on a grid. Each side of the square is of length 1.

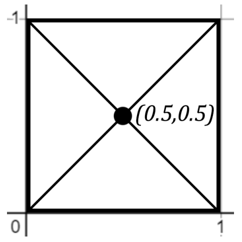

Figure 8: Visual demonstration of the center of a single pixel (square).

The center of a single pixel is the point at which the diagonals of the square intersect one another. As demonstrated by Figure 8, for a single pixel, the coordinate of the center would simply be (0.5, 0.5).

If we wanted to calculate the center for a collection of pixels, this would involve the summation of centers from each individual pixel. The coordinates of the center $(\overline{x}, \overline{y})$ for a composite shape may be expressed as

$$\overline{x} = \frac{\sum_{i=0}^{n} x_i}{n+1}, \overline{y} = \frac{\sum_{i=0}^{n} y_i}{n+1}$$

where n is the number of squares in the shape, and (x_i, y_i) is the coordinate value of each center for every square in the shape. We index the set beginning at the first element $i = 0$ to match the same notation used by our computer program. [3]

In other words, if we were to have a shape consisting of several pixels of uniform density, the coordinates for the center of the shape would be the average locations of each component square's centers.

Consider an example where we must determine the center for a shape comprised of several pixels, such as the letter L in Figure 9:

Figure 9 is made up of several squares: S_1, S_2, S_3, and S_4. From Figure 8, we know that the center of S_1 is $(0.5, 0.5)$. This logic should follow for the remaining squares, readjusted for their position in space. Therefore, S_1 has a center of $(0.5, 0.5)$, S_2 has a center of $(0.5, 1.5)$, S_3 has a center of $(0.5, 2.5)$, and S_4 has a center of $(1.5, 0.5)$.

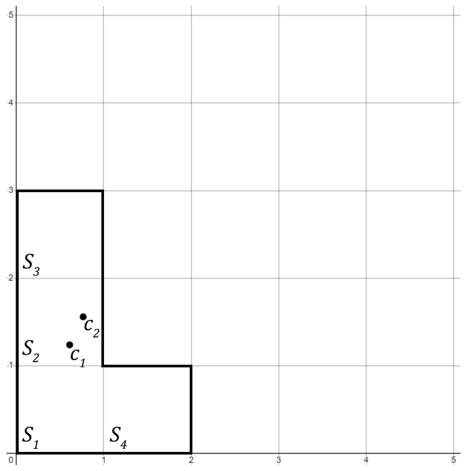

Figure 9: A composite shape of several squares.

According to our formula for the composite center, we may compute that for this shape it is as follows:

$$\overline{x} = \frac{(.5 + .5 + .5 + 1.5)}{(1 + 1 + 1 + 1)} = \frac{3}{4}, \overline{y} = \frac{(.5 + 1.5 + 2.5 + .5)}{(1 + 1 + 1 + 1)} = \frac{5}{4}$$

The center for this composite shape therefore is located at $(\frac{3}{4}, \frac{5}{4})$, located at c_1 in Figure 9.

This computes the center of a shape given a uniform density. We can then work towards implementing the density of our pixels. Upon applying an individual value of mass to each square, such as a color value, the computation of the center becomes the computation for the center of mass, otherwise known as the centroid. With uniform weight, each square was assumed to have an equal weight of 1. This can be expressed in the equation for our center, now a centroid, by multiplying each coordinate value by its weight of 1. First we can modify our equation for the centroid may be rewritten to reflect this:

$$\overline{x} = \frac{((1 * .5) + (1 * .5) + (1 * .5) + (1 * 1.5))}{(1 + 1 + 1 + 1)} = \frac{3}{4}$$
$$\overline{y} = \frac{((1 * .5) + (1 * 1.5) + (1 * 2.5) + (1 * .5))}{(1 + 1 + 1 + 1)} = \frac{5}{4}$$

Doing so incorporates the fact that each square is of mass 1 into our computation. What happens when this mass is no longer assumed to be 1 for each square? We now implement a non-uniform density to our shape. Let S_4 have twice as much mass S_1 and S_2, and S_3 have three times as much mass as S_1 and S_2. In computing the weighted average of coordinate centers, the value of each component center from each square must affect its overall centroid twice and three times as much, respectively. Our equation for the new, weighted centroid may be rewritten to reflect this:

$$\overline{x} = \frac{((1 * .5) + (1 * .5) + (3 * .5) + (2 * 1.5))}{(1 + 1 + 3 + 2)} = \frac{5.5}{7},$$

and

$$\overline{y} = \frac{((1 * .5) + (1 * 1.5) + (3 * 2.5) + (2 * .5))}{(1 + 1 + 3 + 2)} = \frac{10.5}{7}.$$

As opposed to multiplying each center by a value of mass 1, we now multiply them by their respective values of mass. Understandably, the centroid would shift both upwards and towards the right, located at c_2 in Figure 9.

This concept of calculating centroids will be applied to our Voronoi polygons. The computed centroid within a given Voronoi Polygon is the location for each points' movement for a given iteration of Lloyd's Algorithm. [4]

5 Lloyd's Relaxation

Having taken the time to understand all the requisite steps of Lloyd's Algorithm, we may now put them all together and understand how each step informs the process. Going over an example of algorithm implementation will allow us visualize each step. Beginning with Voronoi Diagrams, we have partitioned the plane into regions closest to each point. These regions determine the maximum extent of a point's movement within a given iteration. Using an image's RGBA data, we may extrapolate a visual density and compute a center of mass, or centroid, for which the point to move to for each step. First we review all the steps so we can see the algorithm all at once. Following this we will go into each step in detail with an example.

1. We begin with the arbitrary points on our plane.
2. Construct the Voronoi diagram for the points on our plane.
3. Determine the weighted centroid for each Voronoi polygon.
4. Move each point fully towards its centroid.
5. Repeat from step 2.
6. The Algorithm concludes when the initial position is equal to the calculated centroid. In other words, the points are "relaxed".

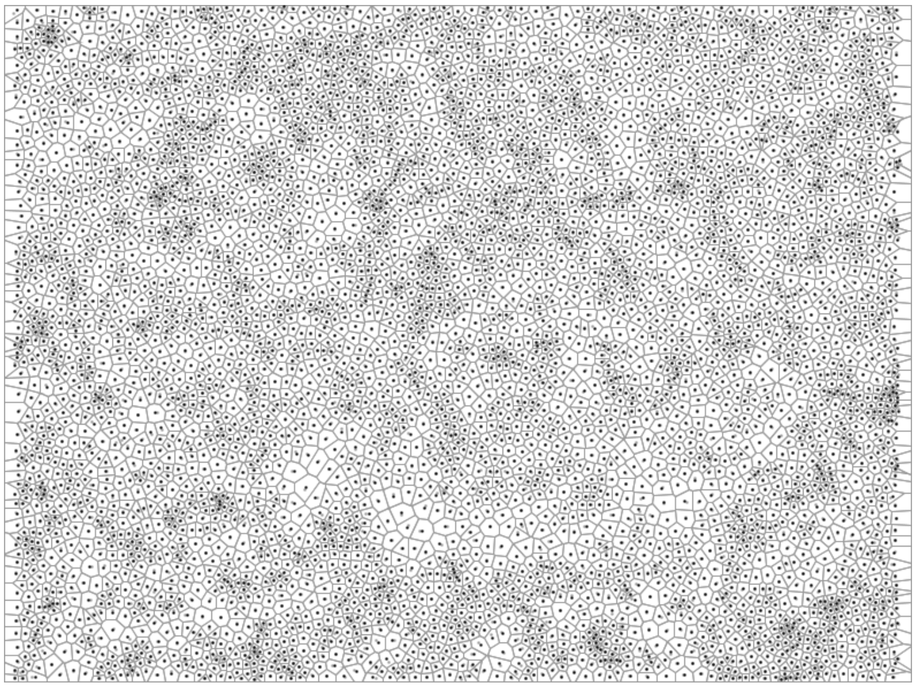

Figure 10: A Voronoi Diagram of many points randomly scattered, which serve as the beginning canvas for Lloyd's Algorithm stipple art.

There are many readily available and open source choices for software for Lloyd's Relaxation. Figure 10 is a screenshot of an open-source computer program known as StippleGen2. StippleGen2 is a useful asset in pedagogy as the Voronoi Diagrams can be visualized as the process is being iterated. In the screenshot, Lloyd's Algorithm has already begun its iteration. In Step 2, we start by generating a Voronoi Diagram for our plane because it dictates the maximum extent of movement for each point. Voronoi Polygons record how the points are initially spatially distributed for each iteration. For our algorithm, the extent of movement for each point should remain within its respective polygon. We can see the points from previous centroids in grey, and the calculated centroid for the current step in black.

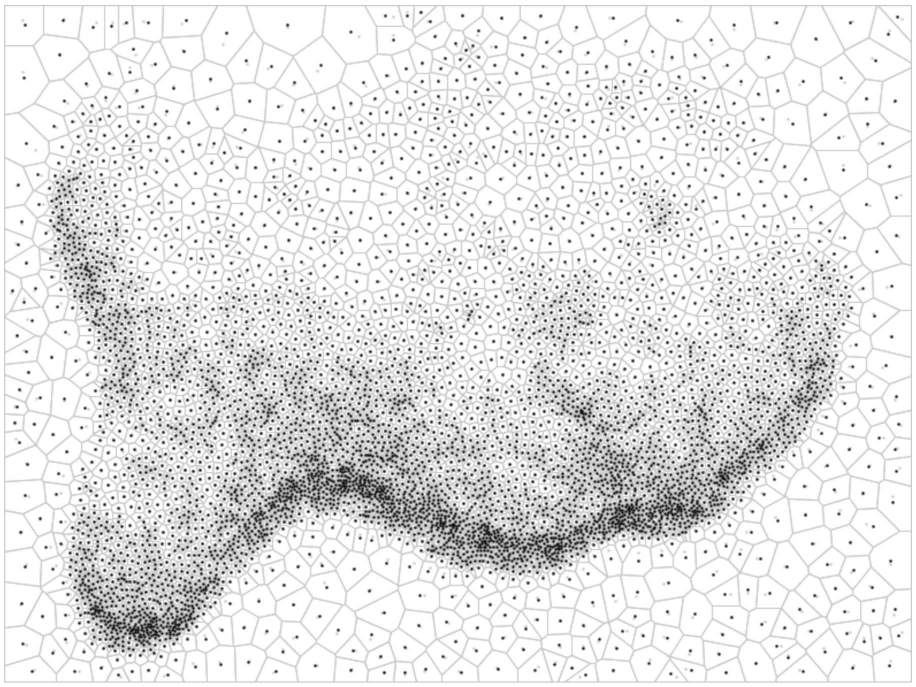

Figure 11: The canvas of Figure 10 having been loaded with the image of the goldfish after 19 iterations. The initial position for each point is in grey, where the calculated centroid for the next step is in black.

Figure 11 demonstrates Lloyd's Relaxation applied to the image of the goldfish. Step 1 of the algorithm, shown in Figure 10, are in a position that now roughly begins to resemble the goldfish. Using the R value from each pixel in our image quantifies a visual aspect of image data. The centroid for each Voronoi Polygon is the location at which the color intensity is greatest in that region. Where there are greater concentrations of a given color or opacity, it should follow that a stipple point is best placed at the greatest concentration of color intensity.

This computes where the points' movement should occur. Simply put, the computer's method of stippling an image is just moving the stipple points towards the areas of the image with the greatest visual density. This is best done through computing the center of mass, as we have demonstrated for centroids. Since the Voronoi Diagrams delineated regions of the canvas closest to each point, the centroid of a Voronoi polygon is the point at which its color intensity is the greatest. Repeated calculation of the location of greatest color intensity continually brings each point to its best approximation for stipple art.

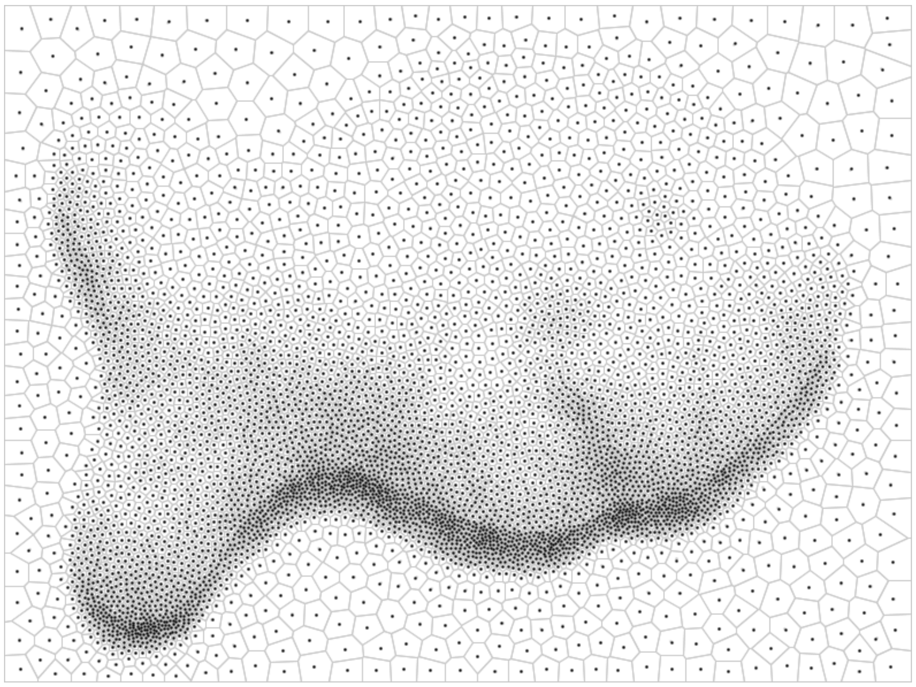

Figure 12: 34 iterations after Figure 10, demonstrating a much clearer art rendition.

Figure 12 shows Lloyd's Relaxation after 34 iterations, where each point is now also the calculated centroid, completing the algorithm. The likeness of the goldfish from Figure 10 is much clearer. Adding more points would further help define the image. This is where Lloyd's Algorithm has finished, as there is no more movement to be made. Thus completes the stipple art, having "relaxed" the points to their computed final locations.

6 Conclusion and Further Work

Lloyd's Algorithm is one example to computer generate stipple art. The algorithm does so by iteratively rearranging randomly scattered points in order to represent an underlying image. The algorithm begins by generating a Voronoi Diagram for all points on the canvas. This delineates the proximity between points through Voronoi Polygons and establishes a spatial relationship with respect to the underlying image canvas. The computer then draws the image's RGBA data from each pixel. The RGBA data for an image provides information about the color intensity for each pixel, and subsequently the density of color throughout the image. The visual center of mass for each Voronoi Polygon is the location for which the points in the stipple art will move towards. This iterative process is Lloyd's Algorithm. The algorithm is finished when the Voronoi point is in the same position as the centroid for its Voronoi Polygon.

In assessing opportunities for future work, there are two directions, computational and mathematical, that can drive further understanding. Further work through computer science drives understanding more options for implementation. This may involve live integration as a website element, or the ability to adjust settings in real time to visually observe changes made. Further mathematical work seeks to understand alternative algorithms for computer-based stipple art generation. For example, other computer generated stipple art methods use the Delaunay Triangulation as opposed to Voronoi Diagrams. The Delaunay Triangulation is the straight-line dual of the Voronoi Diagram.

References

[1] F. P. Preparata and M. I. Shamos, *Computational Geometry: The Locus Approach to Proximity Problems: The Voronoi Diagram.* Springer-Verlag, 1985, p. 204–211.

[2] S. S. Snibe, “Introduction to voronoi diagrams - lecture,” March 1993.

[3] R. Larson and B. Edwards, *Calculus: Early Transcendental Functions*, 6th ed. Cengage Learning, 2014, p. 486–487.

[4] P. Rosin and J. Collomosse, *Image and Video-Based Artistic Stylisation.* Springer London Heidelberg New York Dordrecht, 2013, pp. 45–53.