First publ. as: Paper / Eurographivs 2003

Beyond Stippling - Methods for Distributing Objects on the Plane

Stefan Hiller[1], Heino Hellwig[2], Oliver Deussen[1]

[1]Department of Computer and Information Sciences
University of Constance, Germany

[2]Department of Computer Science
Dresden University of Technology, Germany

Abstract
Conventionally, stippling is an effective technique for representing surfaces in pen-and-ink. We present new efficient methods for stipple drawings by computer. In contrast to already existing techniques, arbitrary shapes can be used in place of dots. An extension of Lloyd's Method enables us to position small objects on a plane in a visually pleasing form. This allows us to generate new illustration styles. Similar methods can be used for positioning objects in other applications.

1. Introduction

In science, especially in the fields of architecture and biology, stipple drawings are widely used to illustrate books. Stippling is a general illustration technique which is used to represent tone and texture of a surface. The dots form a so called Poisson disc distribution. Such a distribution is characterized by the property, that around each object a disc of given size can be placed that covers no other object of the distribution. With stippling many different objects can be represented.

Stipple patterns in the form of dotted lines represent structured surfaces, whereas dots of varying shapes are used for rough surfaces. In other styles, small objects are distributed to represent the objects. For synthetic stipple drawings, several methods for placing points have been proposed, but so far the use of algorithms for distributing small objects instead of points, has not been introduced.

In our paper we present such methods. At the beginning, the objects are randomly distributed. A special type of relaxation method (Lloyd's Method) converts this distribution into a Poisson disc distribution. Our work is an extension of an earlier paper[3] that introduced Lloyd's Method to stippling on the basis of points. The new method allows us to incorporate the shapes of the objects. While the original method solely moves the objects, now their orientation can also be changed. If objects are not convex or if they overlap, new cases arise during the relaxation procedure. After reviewing related work and some basic definitions we discuss the method, show examples of stipple-drawings with different objects and present further applications.

2. Related Work

As mentioned above, Lloyd's Method was already applied to stippling[3]. In our earlier paper we presented a semi-automatic method that allows the user to synthesize such drawings by computer. An initial point distribution is generated manually or from a given image by applying a half toning method. During interaction, the user selects regions and edits their points. The system allows to insert, delete or to move points by special brushes. A particular brush enables the user to convert a point set into a Poisson disc distribution. This is achieved by applying Lloyd's Method that moves the points into the desired distribution by computing the Voronoi Diagram of the points and subsequently moving the points to the center of gravity of their Voronoi Areas (see below). Moving the points in such a way, the user can achieve visually pleasing stipple drawings with tens of thousands of points within some hours, whereas a manually generated drawing would be by far more time consuming.

Secord[15] extends the method by restricting the movement of points by an underlying image. The gradient of the image is used to constrain the movement of the points during relaxation: if the gradient is above a threshold, points are only allowed to move at right angles to the gradient. This extension enables him to automatically convert a given image into a stipple drawing.

Meruvia et al.[11] generate stipple patterns on 3D-objects by triangulating the surface and by positioning the dots on the triangle vertices. This is a very fast method but the resulting patterns are not optimal. Ebert et al.[9] use random point sets (not Poisson disc distributions) to visualize volume data in a style similar to stipple drawings. The method does allow to enhance volume images, but is restricted to points.

The paper most closely related to our work was presented by Hausner[6]. Based on Lloyd's Method he developed an algorithm to distribute small square tiles to form synthetic mosaics. Lloyd's Method is used to position the tiles evenly on the plane. Here the orientation is controlled by a gradient field which was calculated in a pre-processing step from a given source image. The method works in this case, because the square tiles are quite compact convex objects and the author focuses on the Manhattan distance which allows him to use the same iteration method for points and squares. We will show later that for other objects such as small lines or polygons the iteration in this form is not quite optimal.

Such small objects have to be distributed in several applications. One example is cross-hatching: In a set of articles[17, 18, 14] stroke textures are used to pre-define optimally positioned cross-hatchings of small strokes. The textures are combined to form a large non-photorealistic drawing. While this method results in beautiful images, the user has limited control over the drawing process. Deussen and Strothotte[4] use a variant of error diffusion for distributing small strokes in a drawing. Ostromoukhov[13] distributes small convex polygons to halftone images artistically. A point set similar to a Poisson disc distribution is generated by a spring mass system. The Voronoi Diagram is calculated and the Voronoi Areas of the points are used to generate small polygons that represent the grey scale level of the image.

Instead of offering fixed positions for strokes and other small objects, in our paper we present a technique that allows to efficiently distribute sets of objects on the plane. The placement can be constrained by polygonal borders, the positioning can be more or less uniform. The user has full control over the distribution process e.g. he/she is able to control the orientation of the objects during iteration, may switch between different relaxation variants and/or can stop the iteration after some steps, to obtain distributions that are not too even.

Besides creating non-photorealistic drawings, other applications can be found. One field of application is botany, where sets of plants with different shape –defined by their two-dimensional outline on the ground– have to be arranged for complex ecosystems. Another field of application is animation: the iteration can be used for various kinds of animation effects.

Converting objects instead of points into a Poisson disc distribution requires two problematic steps: Firstly, a fast method for calculating Voronoi Diagrams of arbitrary objects is needed. Such a method was proposed by Hoff et al.[7] and can be found in Appendix A. Secondly, the relaxation must be implemented efficiently. In addition to the movement of the objects their orientation has to be taken into account. We show different methods to incorporate the shape of objects and demonstrate their usefulness by showing example drawings. In the appendices the fast calculation of Voronoi Diagrams and mass moments is described.

3. Lloyd's Method

We assume that we are able to calculate the Voronoi Diagram of a given set of objects on the plane (cf. Appendix A). To introduce the Llody's method[8] we see the process in a broader view: The Voronoi Diagram of a given set of objects can help to solve optimisation problems[1]. For example a cost function

$$F(p_1,...,p_n) = \sum_{i=1}^{n} \int_{V_i} f(\|p-p_i\|^2)\phi(p)\mathrm{d}p \qquad (1)$$

has to be optimised for a set of points $p_1,..,p_n$. The function $\phi(p)$ denotes the desired density of points in space and $f(\|p-p_i\|^2)$ is a simple cost function using the Euclidian distance.

The cost function is optimised locally using Voronoi Regions. The Voronoi Diagram is computed and each point is moved to the center of gravity of its Voronoi Region. This ensures that after each iteration the point fulfils the optimisation goal for its local area.

The iteration is repeated until it converges. In practice the movement of the points is stopped if the differences are below a given threshold. In this case the distribution is called a Centroidal Voronoi Tessellation. The algorithm is known as Lloyd's Method[1]. The convergence of the process has been proven analytically only for some special cases[5]. Nevertheless, experiments with various object sets show convergence in almost all cases. A local oscillation of an element over some iterations occurs rarely.

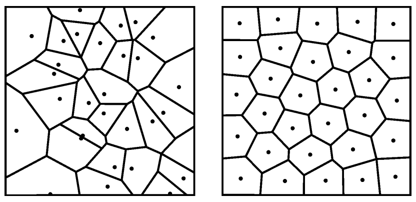

Figure 1: *Point set and its Voronoi Diagram: a) without relaxation; b) after 50 iterations.*

In Figure 1 the relaxation algorithm is applied to a point set. At the beginning, the shape of the Voronoi Regions is very irregular, during the iteration most of the inner regions reshape to hexagons. If the pixel-based algorithm of Appendix A is used, the border of the drawing area prevents points from spreading out all over the plane. In an analytic implementation, the outer Voronoi Regions have to be closed separately before calculating the center of gravity. This can be done by intersecting the Voronoi Regions with the polygonal border of the image plane or a given object.

The Lloyd's Method was introduced into computer graphics by McCool and Fiume[10] in the context of sampling. They also proposed a method called dart throwing that generates similar point patterns[12]. Random points are produced, the points are added to the already existing point set, if they do not fall into the disc of a point. The method works well for low densities, if the density is near the limit that is given by the number of points and the overall area of the discs, points are added very infrequently and the computation is costly.

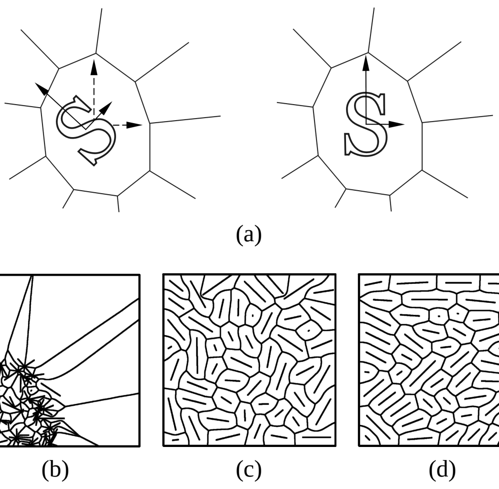

Figure 2: *a) Rotation of an object due to relaxation; b) Lines and their Voronoi Diagram in the initial distribution; c) after 150 iterations without rotation; d) after 150 iterations with rotation.*

4. Lloyd's Method for Line Segments and Polygons

In general, the relaxation can be applied to arbitrary objects, provided that their Voronoi Diagram can be calculated. For each object the center of gravity is determined. During iteration, the object is moved so that its center of gravity lays upon the center of gravity of its Voronoi Region. Additionally, it is possible to rotate the objects.

To do so, we need some mathematical background for the determination of the geometrical moments and the main inertia axis which is given in Appendix B.

In our pixel-based relaxation approach, the object as well as its Voronoi Region is represented pixelwise by a characteristic function $\psi_A(x,y)$. To determine the mass moments the sum in Eq. (3) of appendix B is computed over all pixels that belong to the object and its Voronoi Region respectively. The characteristic function ψ_A of an object A, is defined by

$$\psi_A(x,y) = \begin{cases} 1, if(x,y) \in A \\ 0, if(x,y) \notin A \end{cases} \quad (2)$$

The following formulae (cf. Appendix B) determine the moments up to the order of one:

$$\begin{aligned} m_{0,0} &= \sum_{x=0}^{N-1}\sum_{y=0}^{M-1} \psi_A(x,y), \\ m_{1,0} &= \sum_{x=0}^{N-1}\sum_{y=0}^{M-1} x\cdot\psi_A(x,y), \\ m_{0,1} &= \sum_{x=0}^{N-1}\sum_{y=0}^{M-1} y\cdot\psi_A(x,y) \end{aligned}$$

For $A = V(e_i, S)$ the center of gravity is computed by (c_x, c_y):

$$\begin{aligned} c_x &= m_{1,0}/m_{0,0}, \\ c_y &= m_{0,1}/m_{0,0}. \end{aligned}$$

Similarly for $A = Q_i$ (the object) the mass centroid (z_x, z_y) is computed. We determine the central moments of order one $(\mu_{1,1})$ and two $(\mu_{2,0}, \mu_{0,2})$:

$$\begin{aligned} \mu_{1,1} &= \sum_{x=0}^{N-1}\sum_{y=0}^{M-1} \tilde{x}\cdot\tilde{y}\cdot\psi_A(x,y), \\ \mu_{2,0} &= \sum_{x=0}^{N-1}\sum_{y=0}^{M-1} \tilde{x}^2\psi_A(x,y), \\ \mu_{0,2} &= \sum_{x=0}^{N-1}\sum_{y=0}^{M-1} \tilde{y}^2\psi_A(x,y). \end{aligned}$$

Using these moments and mass centroids we are able to calculate the main inertia axes (Eq. 4 Appendix B) and the desiered rotation angle φ for the object (Eq. 5) . This allows us to set up the algorithm for our extension of Lloyd's Method that is able to incorporate object rotations:

```
Algorithm(MODIFIED LLOYD METHOD)
Input  A set of objects $\{Q_i\}_{i=1}^k$ on the plane and a density
    function $\rho(p)$.
Output  A relaxed centroidal Voronoi tessellation and a re-
    laxed object-distribution.
  1. Determine the mass centroids and main intertia axes
     of the objects $\{Q_i\}_{i=1}^k$
  repeat
    2. Determine the Voronoi-Regions $\{V_i\}_{i=1}^k$ of the
       objects $\{Q_i\}_{i=1}^k$.
    for i=1 to k do begin
      3. Calculate the mass centroids $z_i$ of the Voronoi-
         Regions $V_i$
      4. Move the mass centroids $c_i$ of the objects into the
         mass centroids $z_i$ of the Voronoi-Regions.
      5. Call the Procedure ORIENTATION($Q_i$, $V_i$),
         which adapts the angle of the objects main axis
```

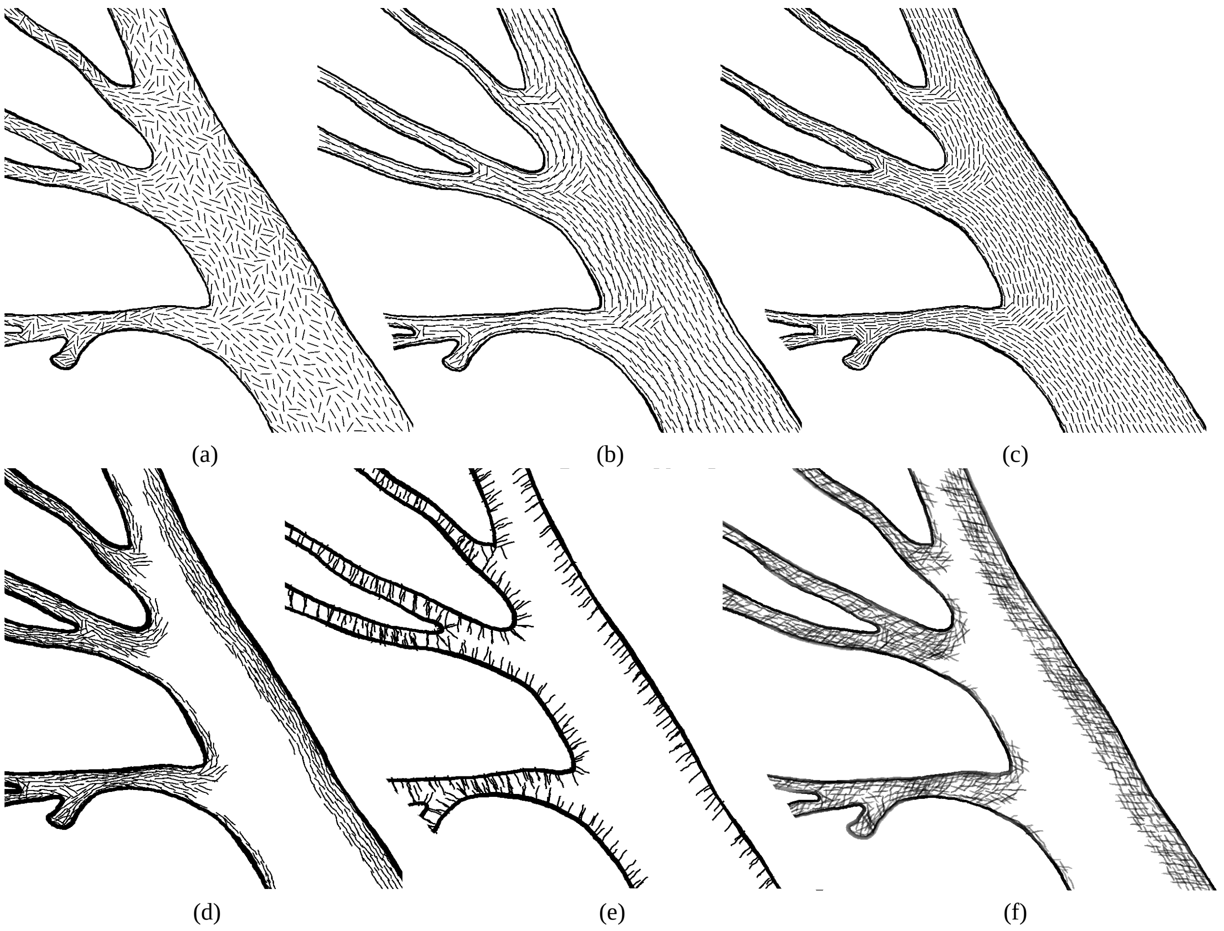

Figure 3: *Several variations of the iteration to fill a bended object shape: a) Iteration with rotation according to case 3; b) Point-based iteration at right angles to gradient; c) Region-based iteration at right angles to gradient; d) Region-based iteration according to gradient; e) at a given angle to the gradient; f) cross-hatching.*

```
            according to an angle $\varphi_i$.
      end
   until the points $z_i$ satisfy some convergence criterion
   end
```

Algorithm(ORIENTATION (Q_i, V_i))

Input A Voronoi-region V_i and an object Q_i, with mass-centroid lying upon the mass-centroid of the Voronoi-region.

Output The Object Q_i turned around its mass centroid with an angle φ_i.

In the subroutine ORIENTATION the adaptation of the iteration to the geometric shape of the objects is performed. We assume that first the mass centroid of the object is moved over the mass centroid of the corresponding Voronoi Region, and that the orientation is adapted. In the implementation we experimented with several variants for the calculation of the angle φ_i. Without loss of generality we restricted ourselves to line segments as well as convex and concave polygons:

1. If we set φ_i to zero, we end up with the basic Lloyd Method that distributes objects properly. Here we can separate two cases: In the first case the objects are regarded as to be point objects at the objects center of gravity, in the second they are treated as objects with extentions. In Figure 2(a) a set of lines is shown. Applying the region-based Lloyd Method we achieve an even distribution as shown in Figure 2(b). Please note that the iteration works well with the very badly distributed initial set shown in Figure 2(a).
2. It is also possible to orient the main inertia axis of the objects according to a given gradient field. If the main axis is rotated to be at right angle to such a field, we obtain a direct extension of the method proposed by Hausner[6] that now incorporates the object shape in combination with Euclidean distances.

3. Heuristically, we also can assume that the direction of the main inertia axies should be similar for each input object and its Voronoi Diagram. Doing so, we rotate the main inertia axes of the object Q_i using φ_i within each iteration into the direction of the main inertia axes of its corresponding Voronoi Region $V(e_i, S)$. In Figure 2(c) the difference is shown. Of course, this operation fails for an object like a disc that has no clear main extension. In this case, the rotation has also no influence on the output.
4. Another idea which we have not implemented yet is to decrease the disturbance of the angle φ_i during the iteration.

The variants of the iteration can be mixed while working with an object set. Similar to what was proposed earlier[3], we build an interactive editor that allows us to model sets of objects in various ways. The user is able to move objects, insert or delete them using a number of “brushes”. A special variant of the editor allows us to apply one or more steps of each variant of the iteration. This kind of mixing allows us to create various distributions for the objects.

In the first steps the basic Lloyd Method might be applied to get a good initial distribution of the objects. In Figure 3(a) such a distribution is shown. The iteration results in a nice hatching like style with nearly no intersections. If too much lines are inserted, such intersections cannot be avoided but our graphical variant of the iteration is surprisingly stable for these cases.

In Figure 3(b) the point-based variant of Lloyd’s Method is applied but the lines are oriented at right angles to the gradient (case 2). After the relaxation, the centers of gravity of the lines are evenly distributed, and due to the orientation of the lines visual patterns occur. This can be avoided by considering the shape of the objects as done in Figure 3(c). Now the lines are evenly distributed according to the gradient. In Subfigure (d) another variant of the iteration is shown: lines are oriented at a certain angle to the gradient, two sets of lines are overlaid. This creates a cross-hatching-like result.

5. Further Results

Additionally, we demonstrate how to change the drawing style for classic stipple drawings. Figure 4 shows a detail of a grass hopper eye that was stippled conventionally with our stippling algorithm[3] using nearly uniform dots (b). Subfigure (c) shows a similar image using dots of varying size. Our extension of Lloyd’s Method allows us to place the dots in a way that no overlapping occurs.

In Figure 5 another illustration is shown. Several objects were distributed for the fish. In the interactive editor the small objects were distributed according to a given source image. First, the image was represented using dots, later the user converted some of the dots into other objects using a special “brush”. To perform the relaxation of the objects, the extended Lloyd Method was used by applying another type of brush. In the drawing we also used an orientation for those parts were lines have to run parallel to each other. As we were not able to implement a gradient here –in Figure 3 the gradient was given by the shape of the tube– an “orientation brush” was used. The same could be achieved by using a hand-drawn gradient field.

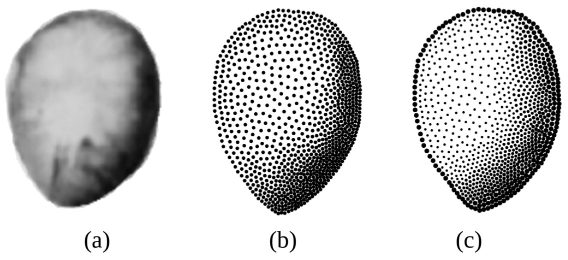
Figure 4: *Extension of stipple drawings: a) input image; b) detailed view; c) version with varying dot size.*

The generation of the fish still needed some hours. Most of the time was spent in editing and selection of the points. The generalized relaxation for sets with several hundred objects is computed with 3-6 iterations per second and therefore plays no significant role during editing.

6. Future Work

In the future we will apply the relaxation to other illustration styles and extend our algorithm to various types of cross hatchings. Also a size-dependent stippling method has to be implemented so that objects can be illustrated on differently sized displays.

Another interesting field of research is the specification of complex ecosystems. We will integrate various density functions and modifications like using different 3d shapes for the objects. Doing so, we hope to be able to generate various distributions ranging from aggregated to evenly spaced sets. This is an extension of work done in [2] in which eco systems were modelled by using efficient specification algorithms.

References

1. B. Boots, A. Okabe, and K. Sugihara. *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. John Wiley, 1992.
2. O. Deussen, P. Hanrahan, M. Pharr, B. Lintermann, R. Měch, and P. Prusinkiewicz. Realistic modeling and rendering of plant ecosystems. In *SIGGRAPH 98 Conference Proceedings*, pages 275–286. ACM Siggraph.
3. O. Deussen, S. Hiller, K. van Overveld, and T. Strothotte. Floating points: A method for computing stipple drawings. *Computer Graphics Forum*, 19(4):40–51, Eurographics 2000 Conference Proceedings.

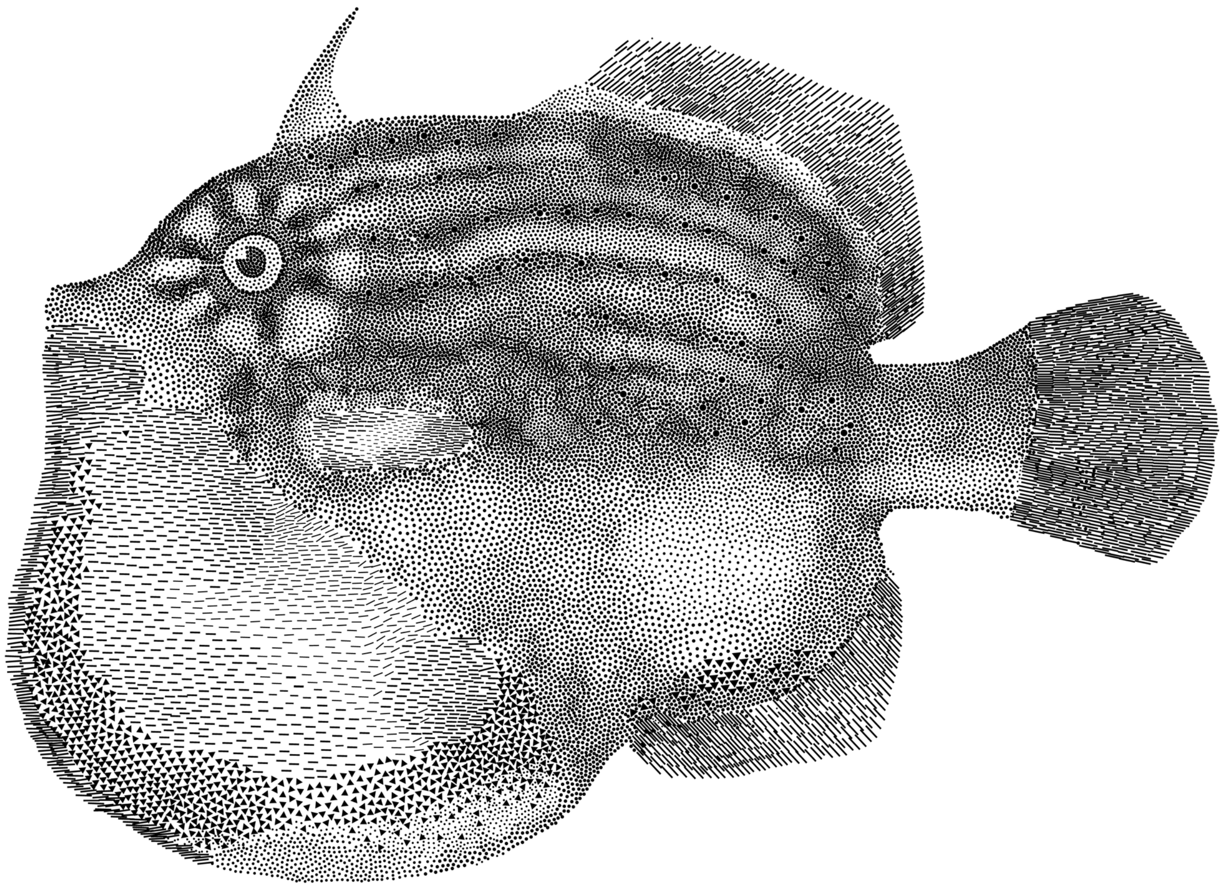

Figure 5: *A fish rendered by distributing various types of objects.*

4. O. Deussen and T. Strothotte. Computer-generated pen-and-ink illustration of trees. *Computer Graphics*, 34(4):13–18, SIGGRAPH 2000 Conference Proceedings.

5. Q. Du, V. Faber, and M. Gunzburger. Centroidal voronoi tesselations. *Siam Review*, 41(4):637–676, 1999.

6. A. Hausner. Simulating decorative mosaics. In *SIGGRAPH Conference Proceedings*, pages 573–578, 2001.

7. K. Hoff, T. Culver, J. Keyser, M. Lin, and D. Manoc ha. Fast computation of generalized voronoi diagrams using grap hics hardware. In *SIGGRAPH Conference Proceedings*, 1999.

8. S. Lloyd. Least square quantization in PCM. *IEEE Transactions on Information Theory*, 28:129–137, 1982.

9. A. Lu, C. Morris, D. Ebert, P. Rheingans, and P. Hansen. Non-photorealistic volume rendering using stippling techniques. In *IEEE Visualization 2002*.

10. M. McCool and E. Fiume. Hierarchical poisson disk sampling distributions. In *Graphics Interface '92*, pages 94–105, 1992.

11. O. Meruvia. Frame-coherent stippling. In *Eurographics 2002 Proceedings (short presentation)*.

12. D. Mitchell. Generating antialiased images at low sampling densities. In *SIGGRAPH '87 Conference Proceedings*, pages 65–72, 1987.

13. V. Ostromoukhov and R. Hersch. Stochastic clustered-dot dithering. *Journal of Electronic Imaging (Special issue on Color Imaging: Device-Independent Color, Color Hardcopy, and Graphic Arts)*, 8(5), 1999.

14. M. Salisbury, M. Wong, J. F. Hughes, and D. Salesin. Orientable textures for image-based pen-and-ink illustration. In *SIGGRAPH '97 Conference Proceedings*, 1997.

15. A. Secord. Weighted voronoi stippling. In *2nd International Symposion on Non-Realistic Animation and Rendering (NPAR)*, pages 37–43. ACM Press, 2002.

16. M. Teague. Image analysis via the general theory of moments. *Journal of the Optical Society of America*, pages 920–930, 1980.

17. G. Winkenbach and D. Salesin. Computer-generated pen-and-ink illustration. In *Proceedings of SIGGRAPH 1994*, pages 91–100.

18. G. Winkenbach and D. Salesin. Rendering parametric surfaces in pen and ink. *Computer Graphics*, 30(4):469–476, SIGGRAPH '96 Conference Proceedings.

Appendix A: Voronoi Diagrams

Let n objects $S = \{e_1, e_2, ...e_n\}$ be defined in two dimensional space and a distance measure $\|\ \|$ be given. This can be the Euclidian norm but any other norm will work. For each pair of elements $e_i, e_j \in S$ the bisector $B(e_i, e_j)$ is defined by all points $p \in R^2$ with equal distance to e_i and e_j:

$$B(e_i, e_j) = \{p \in R^2; \|e_i - p\| = \|e_j - p\|\}$$

The plane is divided into two half spaces by this bisector, we define one of this half spaces by

$$D(e_i, e_j) = \{p \in R^2; \|e_i - p\| < \|e_j - p\|\}.$$

The half space contains all points of the plane that are nearer to e_i than to e_j. The Voronoi Region of an element $e_i \in S$ is the Region of all points nearer to e_i than to any other element of S

$$V(e_i, S) = \bigcap_{e_j \in S \setminus \{e_i\}} D(e_i, e_j)$$

The Voronoi Regions of each two elements are disjoined because there is no $p \in R^2$ which belongs at the same time to $D(e_i, p)$ and $D(e_j, p)$ for any $e_i, e_j \in S, i \neq j$. The set of Voronoi Regions of all elements of S tessellates the plane, but some of the regions are open.

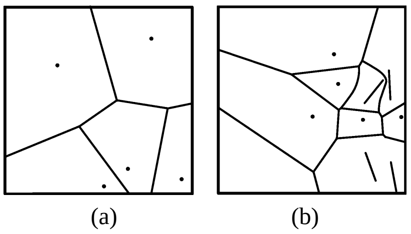

Figure 6: *a) point set and its Voronoi Diagram; b) object set and its Voronoi diagram.*

To calculate the Voronoi Diagram of arbitrary objects we extend the distance function. The distance between a point p and an element e_i is defined as the minimum distance of p to points from e_i

$$\|e_i - p\| = min\|q - p\|, q \in e_i, p \in R^2$$

The bisector in this case is not a straight line, i.e. for a point and a line segment it contains a line segment and two paraboloids (cf. Fig. 6).

While Voronoi Diagrams of point sets can be calculated efficiently, the sets for general objects are hard to compute. Fortunately, Hoff et

al.[7] propose a simple method that uses graphics hardware. In this approach, the Voronoi Diagram is calculated pointwise for each pixel of the image plane. Looking from the negative z direction using a parallel projection, all elements are positioned in the $z = 0$ plane. Each element is represented by a special 3d shape that contains the distance to the element in each of its pixels depth values. In the case of a point object this shape is a cone, in the case of a line object it consists of two half cones and their straight connection (see also Fig. 7).

If these objects are colored individually, the Voronoi Regions can be obtained by reading back the image viewed from the negative z-direction and by determing the pixels of the desired color [7].

Appendix B: Calculation of Mass Moments

In the second appendix we give some mathematical background for computing geometrical moments and the main inertia axis[16]. Let $p = (x, y)$ be a point on the plane. Geometrical moments of the order $k + l$ for a given density function $f(x, y)$ are defined as:

$$m_{k,l} = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} f(x,y) x^k y^l dxdy.$$

The mass centroid $c = (c_x, c_y)$ of the object is then defined as

$$c = \frac{1}{m_{0,0}} (m_{1,0}, m_{0,1}).$$

A central coordinate is defined as

$$\tilde{p} = (\tilde{x}, \tilde{y}) = (x - c_x, y - c_y).$$

With the help of these central coordinates the central moments of order $k + l$ are defined as:

$$\mu_{k,l} = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} f(x,y) \tilde{x}^k \tilde{y}^l dxdy$$

In the discrete case the geometric (Eq. (B)) and the central (Eq. (B)) moments are calculated as a finite sum over all pixels of the given $N \times M$-picture. We receive:

$$m_{k,l} = \sum_{x=0}^{N-1} \sum_{y=0}^{M-1} f(x,y) x^k y^l \qquad (3)$$

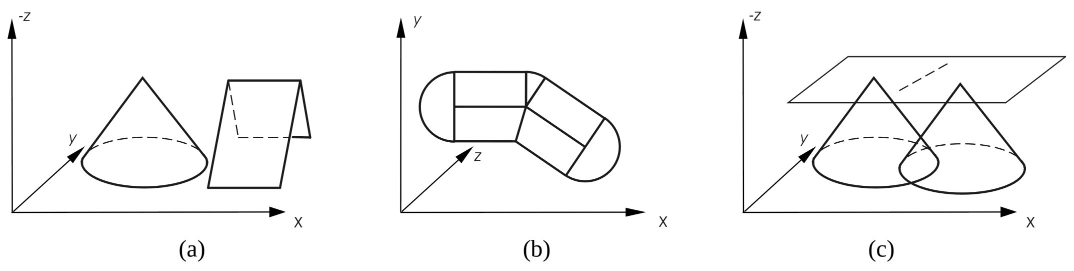

Figure 7: *Construction of 3d shapes: a) shapes for a point and the body of a line segment; b) shape for two adjacent line segments viewed from above. The half cones can be omitted partially; c) projection of two shapes onto a plane and resulting bisection.*

and

$$\mu_{k,l} = \sum_{x=0}^{N-1} \sum_{y=0}^{M-1} f(x,y)\tilde{x}^k \tilde{y}^l.$$

The two-dimensional inertia tensor is given as

$$\mathbf{J} = \begin{pmatrix} \mu_{2,0} & \mu_{1,1} \\ \mu_{1,1} & \mu_{0,2} \end{pmatrix}$$

The eigenvalues of **J** form the maximal and the minimal inertia moments j_1, j_2. These are computed by:

$$j_{1,2} = \frac{1}{2}(\mu_{2,0} + \mu_{0,2} \pm \sqrt{(\mu_{2,0} - \mu_{0,2})^2 + 4\mu_{1,1}^2}. \quad (4)$$

The angle φ of the main inertia axis is the angle of the eigenvector v_1 of **J** which belong to the eigenvalue j_1:

$$\varphi = \frac{1}{2}\arctan\left(\frac{2\mu_{1,1}}{\mu_{2,0} - \mu_{0,2}}\right). \quad (5)$$