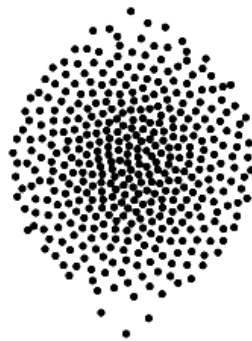


Fun With Dots:
Non-Interactive Stippling
of Greyscale Images and Animations

Adrian J. Secord



UBC ID# 27984004

June 8, 2001

Introduction

The artistic tradition of stippling has a long history, particularly in the field of scientific illustration. Stippled drawings are made up of thousands of small ink dots or *stipples* that combine to give the impression of continuous tone. The artist can control the placement and size of the stipples precisely to delineate particular details and clarify structures that might be lost in a photograph. Recently computers have been used to approximate and extend traditional artistic methods and styles. This project is an attempt to generate stippled drawings quickly from greyscale images and animations.

Related Work

Traditional Scientific Illustration Texts

Scientific Illustration

P. Wood [1]

Wood discusses many advantages of traditional stippling in this text on scientific illustration. Among the most important are the ease at which stippled drawings can be reproduced by many different methods of varying quality and still maintain their visual impact. Black and white are faithfully reproduced by most methods, unlike continuous tones. Stippled drawings can be greatly reduced without losing their detail. Indeed, stipple drawings are typically created 25-100% larger than the intended final reproduction so the reduction process will smooth away small imperfections. Since stipple drawings are normally black and white, they are inexpensive to reproduce. Fine detailing is possible with stippling, up to the resolution of the stipples on the reproduced page. Finally, stippling mixes easily and effectively with other presentation techniques, especially line drawings, which share many of stippling's advantages.

Stippling is often used when illustrating shaded smooth surfaces which are difficult to render well with only lines. In particular, textured surfaces such as stone or old metal are well-represented by stipples. The stipples suggest the high-frequency information of the surface texture without inducing much visual complexity. In a mixed stipple and line drawing, the lines can show overall structure (silhouettes in particular) while the stipples fill in shading and surface texture. The resulting images can be extremely striking and efficient in conveying information without being complicated.

Traditional artists use either a stiff crowquill or mechanical pen when creating stipple drawings. The crowquill pen is a slightly flexible nibbed pen which translates hand pressure into line thickness, allowing variable line thicknesses or stipple sizes. The mechanical pen is a refined version of the familiar ball-point pen which produces very regular stipple sizes. The placement follows fairly strict guidelines to produce the correct effect. Stipples should be placed evenly with a sufficient density to approximate the desired tone. However, the stipples cannot form regular patterns because the human eye is sensitive to them and will be distracted. The stipples should be placed randomly while maintaining

an even distribution and spacing. Stipples should not touch each other unless required to by an area of darkness, where they can start to merge together. Of course, patterns on the original object such as colourations or markings should be represented by the stippling.

Wood notes that a related technique is the use of *coquille board*, which is a rough textured board for charcoal or pencils. When the artist draws on the board, the line appears textured, much like a stone rubbing. The result is somewhat similar to stippling except the “stipple” pattern is fixed to the board and not changeable.

Scientific Illustration

Z. Jastrzebski [2]

In this text on scientific illustration, Jastrzebski introduces stippling as an effective alternative to the original half-toning methods. The original methods of half-toning were expensive and of highly varying quality, and black-and-white reproduction methods were much more acceptable. Stippling enabled the illustrator to control the “half-toning” process and reduce printing costs.

Jastrzebski has more to say about stippling style than Wood. He notes that for delicate subjects the dark areas should use large, dense stipples while light areas should have smaller stipples. A reducing lens is discussed as a tool to view the drawing in progress at such a reduced resolution that the stipples blend together to form a continuous tone. Finally, Jastrzebski notes that the three-dimensionality of a stipple or line drawing is reduced compared to a continuous-tone drawing. However, the stipple and line drawings generally convey information much more economically.

The Guild Handbook of Scientific Illustration

E. Hodges, ed. [3]

In their section on stippling in this text, Hodges and Hodges explain many of the stipple issues noted above but also point out some new items. Relatively small and dense stipples connote softness such as the surface of tissues whereas relatively larger stipples placed further apart and more evenly distributed suggests hardness. While discussing the visual impact of stippled drawings, they claim, interestingly, “Like a pointillist painting, the drawing will appear to vibrate slightly.” (page 111)

In some application-specific notes, they mention that archaeologists use line drawings for hard, shiny stone objects and use stipples for porous, grainy stone objects. The mention, humourously, that stipples are not used to illustrate birds because it can produce “the effect of an avian stone statue rather than a living creature.” (page 354)

Non-Realistic Rendering Papers

Floating points: A method for computing stipple drawings [4]

In this paper the authors introduce an interactive system for creating stipple drawings. They note that the long creation times associated with stipple drawings come from the careful placement of stipples. The stipples must connote tone accurately and not introduce unwanted patterns. They find stipple drawings interesting because their clarity and expressiveness, particularly for natural surfaces.

In non-photorealistic rendering (NPR), there is a relatively common paradigm of approximating continuous tonal values with discrete objects such as pen strokes or hatch marks. Also, stippling is strongly related to digital half-toning, but with some important differences. Half-toning generally applies a tile repeatedly across the image, changing the tile to approximate the underlying tone. Stippling treats the entire image as a single tile so there is no repetitive sub-unit. In half-toning often just the size of the dots are varied and they are fixed with respect to their tile, whereas in stippling the dots are free to move both in size and space.

Relaxing distributions

To achieve the “well-spaced” property of traditional stipple drawings, some method must be used to ensure the inter-stipple spacing is uniform in some sense. The authors start with an initial distribution and relax it using the following algorithm iteratively:

- Compute the Voronoi diagram of the stipple centres
- Move each stipple towards the centre of gravity of the Voronoi region
- Stop when magnitude of movements becomes sufficiently small

Any reasonable initial distribution can be made well-spaced by using this algorithm.

The relaxation method is relatively simple to compute, compared to similar force-based methods. The authors investigated using Delaunay neighbours instead of Voronoi regions, but found that small changes in stipple position caused large changes in the set of Delaunay neighbours. This prevents convergence of the method, and does not happen with Voronoi regions.

The stipples can be confined to a polygonal region, allowing the method to keep stipples in the region of interest and segment different areas. The method is similar to that of Ostromoukhov in [6], but integrates better with polygonal borders.

The authors compared the Fourier spectrum of a relaxed distribution and found it compared favourably to the Fourier spectrum of the ideal well-spaced Poisson distribution.

Creation procedure

The entire procedure for creating a stipple image brings the user into the algorithm:

- Define polygonal regions that segment canvas
 - This could be automatic if working from a reference image
- Assign stippling styles to each region
- Assign initial distribution by:
 - Painting stipples onto canvas en masse
 - Half-toning the reference image
 - * They use a modified half-toning algorithm based on Pulse-Density Modulation from Eschbach in [5]
- Apply relaxation method to each region to obtain well-spaced distribution
- Interactively use different brushes to modify result:
 - Edit brushes which add or remove stipples individually or at a fixed rate
 - Relaxation brush that locally relaxes stipples
 - Jitter brush adds small random position changes to stipples
 - Shape brushes to change size or shape of stipples

Computational Effort

The computation is dominated by the Voronoi diagram calculation which is in $O(n \log n)$. Several timings on a SGI Octane R10000 195 MHz machine are given by the authors. A representative timing is 10,000 points takes approximately six seconds to for each iteration and typically ten to twenty iterations are needed to relax the distribution.

Control of stippling process

The polygonal regions are important to the final result because relaxation will tend to blur edges in the distribution. The polygon boundaries maintain sharp edges in the distribution. Small polygons can also be used to create thin lines of stipples, which artists use to show surface colouration or texture.

Pseudo-random halftone screening for color and black and white printing [6]

Ostromoukhov discusses reasons why pseudo-random distributions such as those used in [4] are useful and applies them to create a digital half-toning method. He notes that pseudo-random distributions were suggested by psychologists and biologists based on observations that they describe the distribution of receptors in the human retina. Distributions with well-defined spatial patterns, e.g. regular grids, cause interference patterns when superimposed which are clearly visible in the Fourier domain as frequencies that do not appear in either source distribution. He demonstrates that the super-position of two pseudo-random distributions does not introduce new frequencies, i.e. Moire patterns. This could be the reason that the human visual system does not create Moire patterns. Since digital half-toning involves overlaying a distribution on the source image, if a pseudo-random distribution is used in the half-toning, Moire-type patterns can be avoided.

Method introduced

Ostromoukhov creates a digital half-toning of an input image with a pseudo-random distribution using the following algorithm:

- Create an initial uniform random distribution of centre points
- Apply forces to centres based on a spring model
 - Springs connect neighbouring centres together
- Randomly choose one centre to move according to the forces on it, repeat until distribution settles
- Calculate Voronoi polygonalisation of plane using the relaxed centres
- Generate a polygonal region centred inside each Voronoi region with an area equal to total image density inside Voronoi region
- Render polygonal region either black or white, depending if total density is above or below a given threshold

Results

The half-toning screen is similar in appearance to normal half-toning except the “screen” is composed of Voronoi regions. The centres of the regions are distributed evenly to approximate a Poisson distribution. In this method the screen covers the entire image, that is, the screen does not tile. The resulting elements are fairly large-grained, so Ostromoukhov suggests that the method is applicable only for high-resolution half-toning where the size of the regions falls below the human visual acuity limit. Because of the pseudo-random nature of the distribution, no additional frequencies are introduced into the resulting image.

Painterly rendering for animation [8]

Meier discusses the solution of several sub-problems related to her goal of animating “painterly” renderings of 3D models. The method associates stroke objects with particles that are attached to the surface of every object in the scene. As the objects move, the particles, and hence the strokes, move with the objects giving temporal coherence. The approach avoids the “shower door” effect in which strokes are attached to a particular screen position, giving rise to the impression that the screen is being observed through a fixed distribution.

Method

Each surface in the scene is tessellated with triangles, then particles are placed in each triangle in number proportional to the area of the triangle. The particles are distributed randomly (with an unspecified distribution) across each triangle. Since the number of particles assigned to each triangle is proportional to the area of the triangle, the total number of particles per object is roughly constant. Each particle stores information about the strokes such as colour, size and orientation. To render, the particles are sorted back-to-front and the associated strokes rendered using the painter’s algorithm.

Problems

The particle placement does not take into account their screen space density, so there is a fixed limit to how close an object can be viewed and still be rendered well. Also, because the particles are rendered from back to front, popping can occur when the animation causes one particle to cross through another.

Orientable textures for image-based pen-and-ink illustration [9]

The authors implement an interactive system for creating pen-and-ink line drawings from greyscale images. They introduce the notion of *orientable textures*, or libraries of stroke styles. They also introduce the idea of an *importance image* in which high values indicate a high “desire” for a stroke to be placed at that location.

Method

The initial important image is proportional to the density of the input image. A *difference image*, which represents the difference between the input image and the current output image, is maintained at every step. The importance image is derivable from the difference image because in regions where the input and output images differ greatly it is highly desirable to place a stroke in that region. Iteratively:

- Place a stroke at the point of highest value in the importance image

- Subtract a blurred image of the stroke from the difference image and update the importance image
- Stop placing strokes when the difference image falls below some threshold

The blurring of the stroke is mimicking the human visual tendency to view discrete strokes as part of a continuous tone variation. This is a very important point and directly applicable to stipple renderings because both place discrete objects to approximate continuous tone. Note also that the entire method acts inherently in screen space. If repeated with a higher resolution output image, the relatively smaller strokes would be placed in larger numbers until the required output densities were satisfied.

Art-based rendering of fur, grass, and trees [10]

In this paper the authors implement a system that renders 3D models in “cartoon”-like styles. Interestingly, they combine the particle method from [8] with the difference image method from [9] to create a hybrid system that allows inter-frame coherency while maintaining a particular density in screen space.

Instead of attaching strokes to the surfaces, the authors use a more advanced procedural-based object called a *graftal*. For a particular frame, the reference image is generated using some 3D graphics library such as OpenGL. Graftals from the previous frame are checked to see if they are still valid, e.g. not occluded, and a blurred version of each graftal is subtracted from the reference image. The resulting image is called the *desire* image which corresponds to Meier’s difference image. If the desire image does not satisfy some error threshold, then new graftals are placed in screen space and their blurred images duly subtracted. As the new graftals are placed, the corresponding 3D surface is looked up in a per-pixel table and the graftal attached to the surface. Once enough graftals have been placed to satisfy the error threshold, the frame is rendered. Then a new reference image is generated for the next frame is generated and the process repeats.

Because the graftals are attached to the 3D surfaces and move accordingly, the inter-frame coherence of Meier is achieved. Also, because screen space is used to place new graftals, Salisbury et. al.’s constant screen-space density is achieved. However, the problem with popping remain and the authors discuss several methods of reducing its severity.

Ideas and Methods

Motivation: The Rejection Method

A commonly-used method of generating random distributions with a given shape is the rejection method. This method begins with a uniform random distribution of n points and the shape of the desired distribution density given as a surface. Figure 1 shows the desired Gaussian surface. Each point is tested in the following way: choose another random number uniformly from zero to one and test if this number is less than the height of the surface at this point. If this test is true, keep the point, else discard it. It is easy to see that regions of the surface that are high will keep greater numbers of points than regions of the surface that are low. The random number used in the test can be visualised as a height above the plane and the points as vertical lines rising from the plane, as shown in figure 1. Then the method keeps those points whose lines fall entirely under the density surface.

Since in stippling we want higher number of stipples in regions of the reference image with higher density, the rejection method could be used to generate stipple distributions. The image would be interpreted as a piecewise-continuous surface with each piece being a flat plane covering a pixel and the height proportional to the image density.

Intersection Method

A small shift of perspective allows us to view the selection of points as intersecting their vertical lines with the intensity surface. Regions of the image with higher intensity will intersect less lines because less lines reach up to that intensity level, resulting in fewer stipples. Figure 2 shows the *intersection method* instead of the rejection method.

Since any collection of lines will produce a distribution of points when intersected with such a surface, we are not limited to vertical lines. However, we do want the average number of intersections to decrease as the intensity increases. Ideally, at intensity $I = 0$ the distribution of points would cover the plane, at $I = 1/2$ the plane would be half-covered and at $I = 1$ there would be no points at all. Covering the plane with infinitesimal points requires an infinite number of points, but if we convert each point into a small circular disc, or *stipple* of

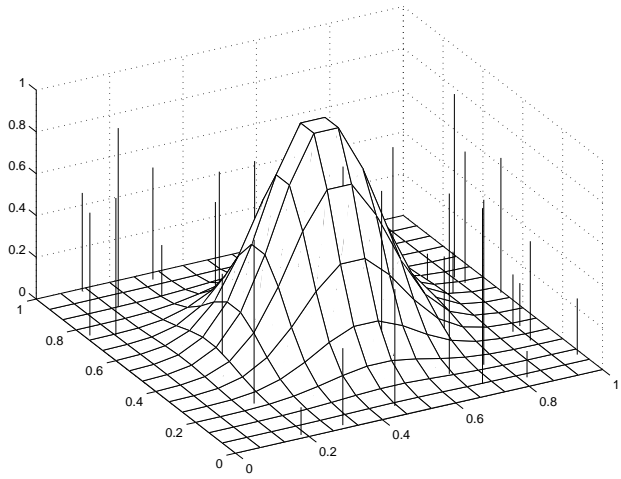


Figure 1: Illustration of rejection method with 40 uniformly random points in $[0, 1] \times [0, 1]$ and a Gaussian surface.

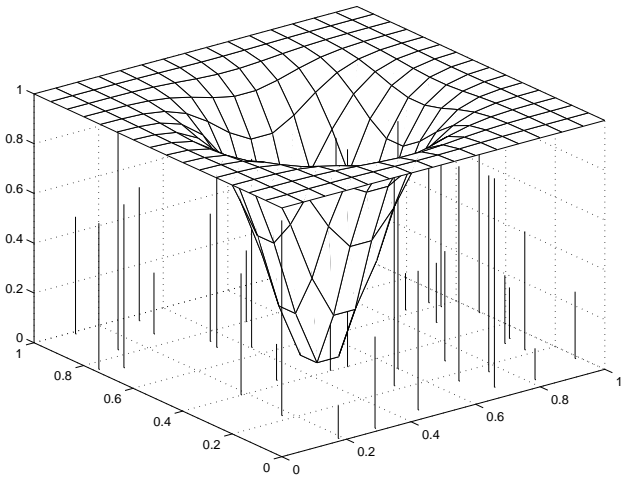


Figure 2: Illustration of intersection method analogous to figure 1.

radius R , then only a finite number is needed. Furthermore, each line in the distribution could carry a potentially different radius R_i , which would be converted to a stipple of radius R_i if that line intersects with the intensity surface.

General Method for Stippling

The general method for producing a stipple drawing is as follows:

1. Convert input image into a piecewise function $I : [0, 1] \times [0, 1] \Rightarrow [0, 1]$ by simple scaling.
2. Generate a distribution of lines in the unit cube of R^3 .
3. Intersect each line with the surface defined by I and record the projection of the intersection point and radius, if any.
4. Display intersection points and radii as stipples.

Distributions

Any number of distributions of lines could be used to generate stipple drawings. However, appropriate distributions should have the following properties:

- Should be defined on the unit cube in $R^3 = [0, 1] \times [0, 1] \times [0, 1]$
- The distribution produced by an input image with uniform intensity I should
 - have an average intensity of I
 - not have overlapping stipples unless necessary to produce the required intensity
 - not have stipples that produce any visible patterns

The last two sub-points are artistic in nature and ensure that the stippling method does not introduce information into the drawing that was not originally in the reference image. The average intensity requirement forces with increasing height lines either to end, to separate from each other, or to do both.

Current Results

Vertical Line Distributions

Four types of distribution have been implemented in the current system, though this is the most easily extendible part of the program and is sure to increase in diversity. All four types are examples of vertical line distributions where lines simply end with increasing height to maintain appropriate densities. To illustrate the distribution, each is used to produce a stipple drawing from a reference image of a grey-scale ramp, shown in figure 3.

Uniform Random Distribution

The simplest distribution is just vertical lines positioned uniformly at random in the unit square with height chosen at random. This is the distribution used in the descriptions of the rejection and intersection methods above. Figure 4 shows the ramp stipple drawing and the output stipple drawing density for different input intensities. Note that this distribution does not have a linear intensity response nor does it ensure that stipples do not overlap.

Random Sobol Distribution

A Sobol sequence is an example of a quasi-random distribution, often used in Monte Carlo integration scheme to sample space efficiently. A quasi-random sequence of numbers gradually fills the entire number line at finer and finer resolutions, never hitting the same point twice. The “random” aspect comes from the sequence’s apparent non-determinism. The Sobol sequence is a particular quasi-random sequence that works well and is simple to compute. After using the Sobol sequence to select a point in the plane, the height of the line is selected uniformly at random. This is the source of the word “random” in the name. Figure 5 shows the ramp stipple drawing and the intensity response of this distribution. Note that it is quite similar to the uniform random distribution, especially in its intensity response.



Figure 3: Ramp greyscale image

Ordered Sobol Distribution

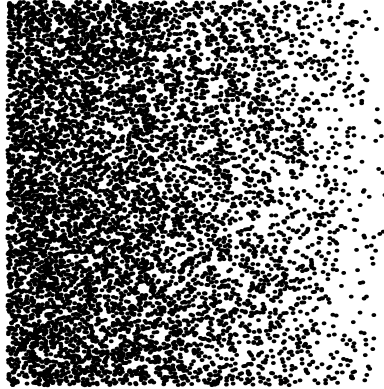
Part of the reason that the random Sobol distribution did not show an improvement is that the Sobol sequence fills in the plane *in order*, at increasing finer resolutions. That is, the second ten percent of the points falls nicely and evenly in between the first ten percent of the points. This indicates that we should not be treating all points equally when testing against the intensity surface, but preferring points earlier in the sequence. We assign the height of the n th line the height n/N , where N is the total number of lines. With this scheme an intensity of 0.1 will select the first ten percent of the points (which are nicely distributed) and an intensity of 0.2 will select the first and second ten percent of the points (which fit nicely together on the plane). Figure 6 shows the ramp and the intensity response. The ramp is much better, especially with respect to overlap, but the intensity response is still not linear.

Osculating Distribution

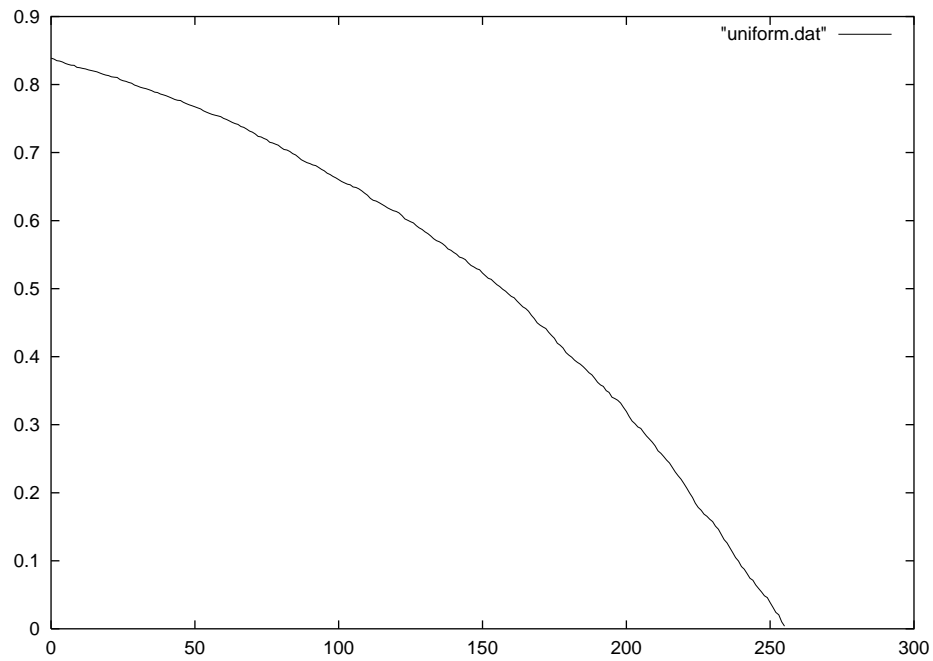
We can create a distribution of stipples that has a “perfect” distribution at intensity $I = 0$ by calculating the positions of the centres of a set of discs packed onto the plane. The discs overlap just enough to allow no gaps in between. The resulting lines are given heights uniformly at random as before. The ramp and the intensity response are shown in figure 7. Note that the intensity response is nearly perfect because the discs do not overlap any more than they must. However, the resulting distribution is very regular in the plane, resulting in a more *dithered* look than stippled. The regularity of the distribution introduces patterns into the output.

Osculating Distribution with Jitter

To relieve the regularity of the osculating distribution, we can *jitter*, or add small random displacements to the centres of the discs. However, we do not

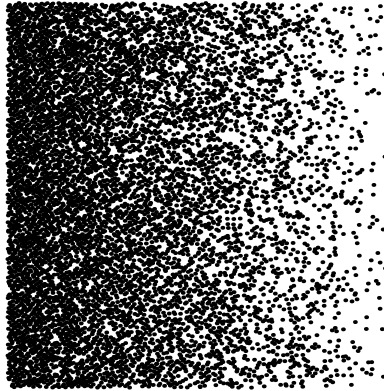


(a) Stipple drawing output

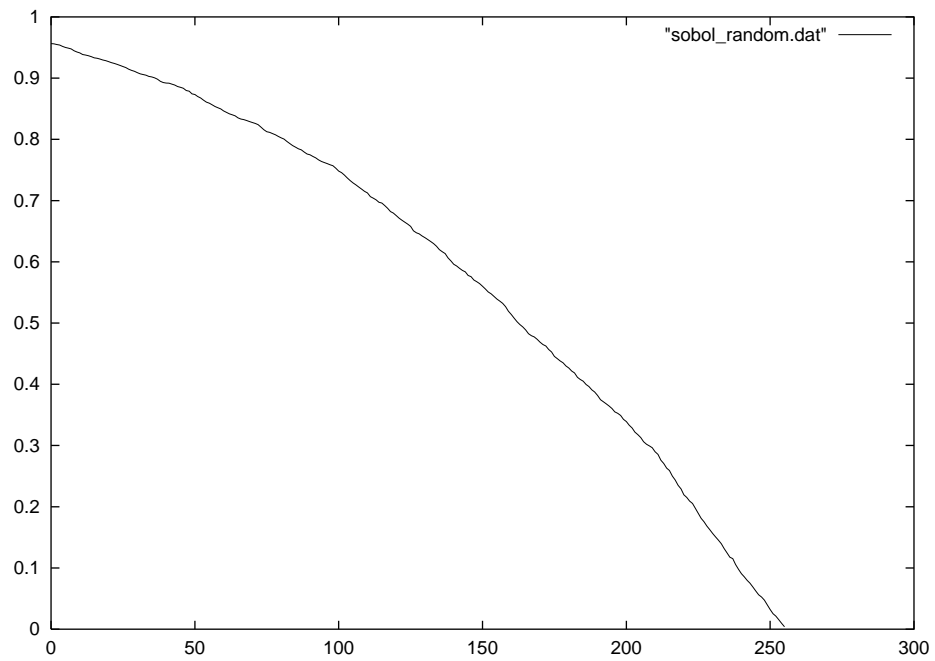


(b) Density of stipple drawing (vertical) versus intensity of a uniform intensity input image (horizontal).

Figure 4: Uniform random distribution of 15 000 lines of radius 0.005 and 7526 stipples.

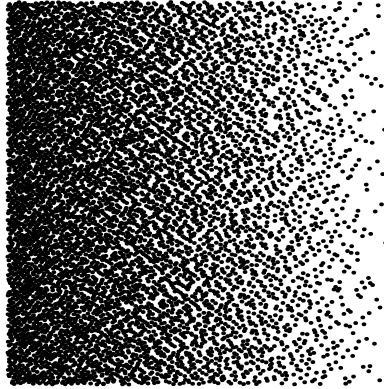


(a) Stipple drawing output

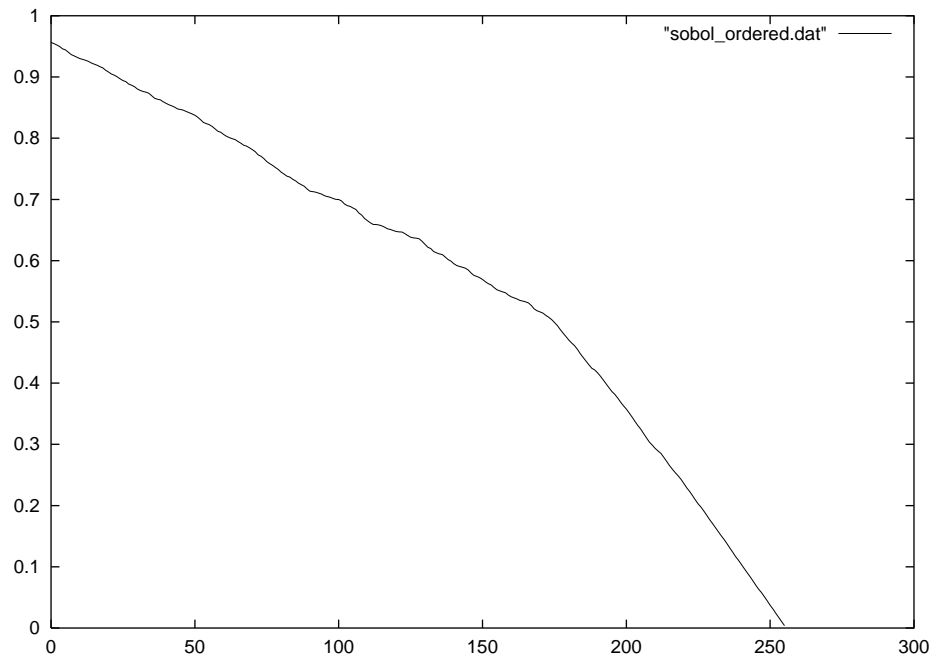


(b) Density of stipple drawing (vertical) versus intensity of a uniform intensity input image (horizontal).

Figure 5: Random Sobol distribution of 15 000 lines of radius 0.005 and 7529 stipples.

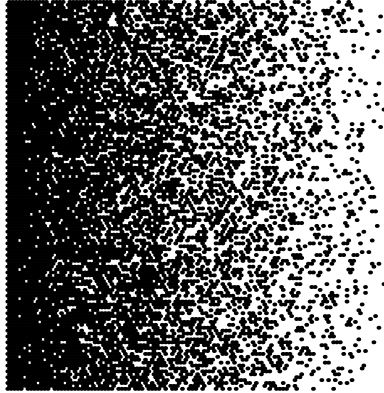


(a) Stipple drawing output

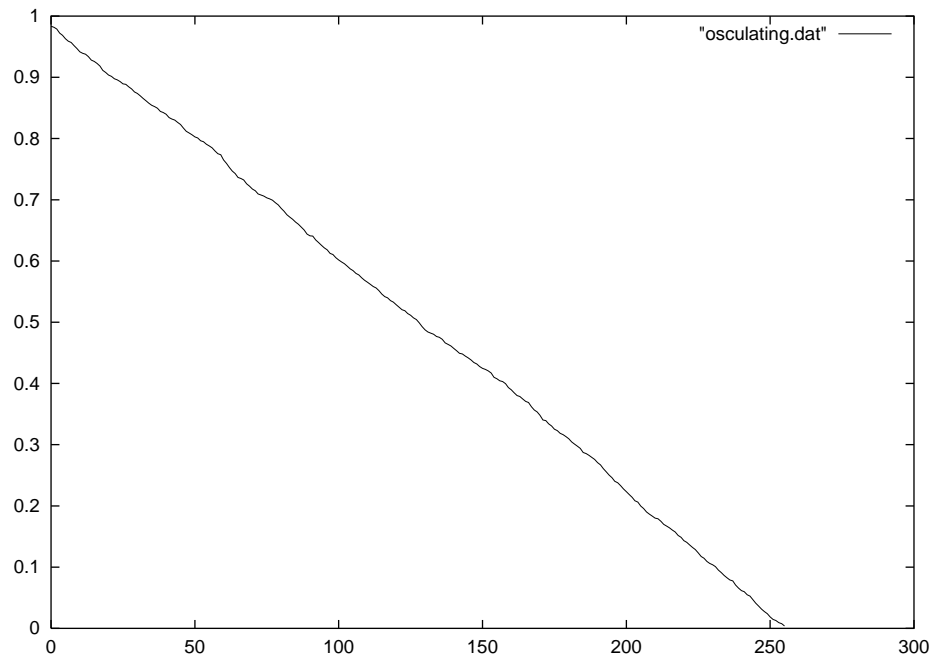


(b) Density of stipple drawing (vertical) versus intensity of a uniform intensity input image (horizontal).

Figure 6: Ordered Sobol distribution of 15 000 lines of radius 0.005 and 7504 stipples.



(a) Stipple drawing output



(b) Density of stipple drawing (vertical) versus intensity of a uniform intensity input image (horizontal).

Figure 7: Osculating distribution of 15 477 lines of radius 0.005 and 7749 stipples.

want to give up the perfect covering distribution at $I = 0$ so we pack the discs closer to ensure no gaps appear. If the jitter is given as a percentage p of the stipple radius then we can guarantee that a disc of radius $(1 - p) * radius$ will be covered at the original position of the stipple. With this information we can calculate the positions so that they do not produce gaps. Figure 8 shows the ramp and intensity response for 40% jitter. Note that the regular patterns are gone from the output and there are no gaps in the $I = 0$ regions, but the intensity response is worse than any of the others so far. Note also that the number of lines required to ensure that no gaps appear is dramatically more than in the unjittered case.

Relaxed Distribution

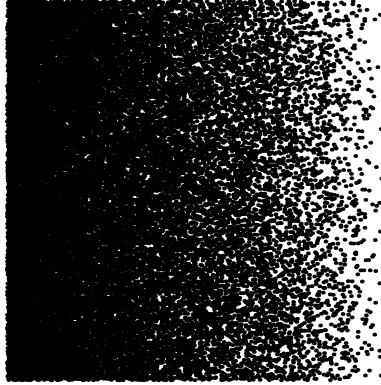
The artist creates a distribution by continually watching the current result and adding stipples in the correct positions to maintain spacing and desired density. This is very difficult to approximate algorithmically in any efficient manner. A related method is to start with a uniformly random distribution and *relax* it into a new distribution with better inter-stipple properties than the original. The uniform random distribution generally has no undesirable spatial patterns, but the stipples overlap with high probability. We can borrow from physics to modify the uniform distribution.

Define the force on a stipple to be $-\sum_{i=0}^n \mathbf{v}_i/d_i$ where \mathbf{v}_i is the unit vector pointing from the stipple to the i th stipple in the distribution and d_i is the distance in between them. Then the distance-based force is highly repulsive at close distances. If we simulate the movement of the stipples under the influence of this force, then stipples that are very close or overlapping will move apart from each other. After many iterations the distribution should converge or relax into a local minimum of the related energy. That is, the distribution will find some set of positions that maximises the inter-stipple distances. Figure 9 shows the ramp with a relaxed distribution.

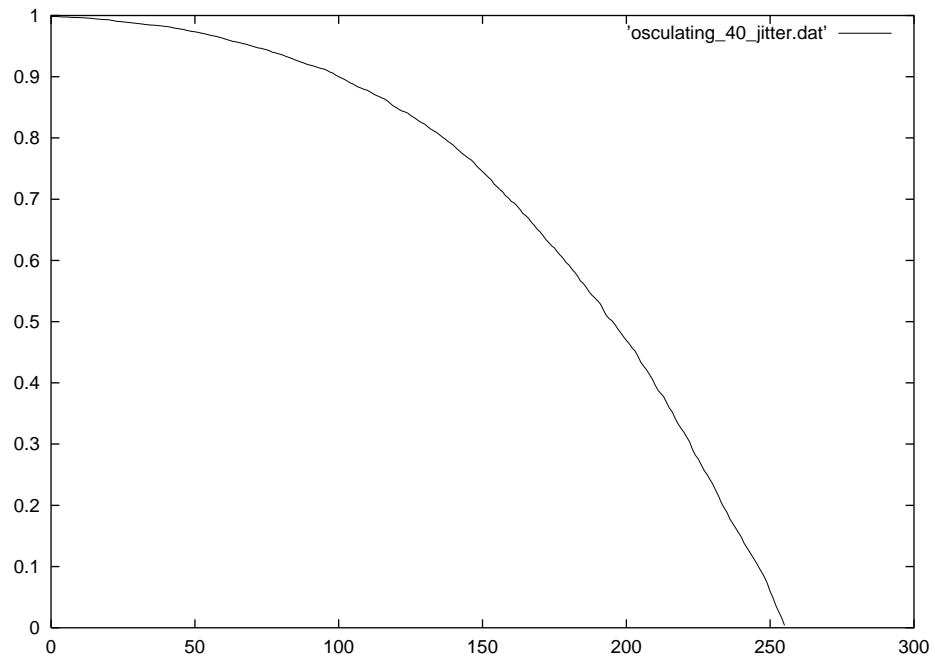
Note the very flat response but that the maximum density is quite low, only about 0.65. This can be improved by increasing the number of lines but the computation is very expensive and slow.

Full Line Distributions

The previous distributions used vertical lines in the three-dimensional space, resulting in stipples that stayed in the same position as one moved to smaller and smaller densities. However, the technique applies equally well to general lines as long as they satisfy the previously stated conditions for distributions. The relaxed distributions from the previous section were extended in the following way. First, a relaxed distribution D_1 of points in the $I = 0$ plane was created as above. This distribution of points forms the 'anchors' of the lines to the $I = 0$ plane. Next, some fraction of the points are deleted and the distribution

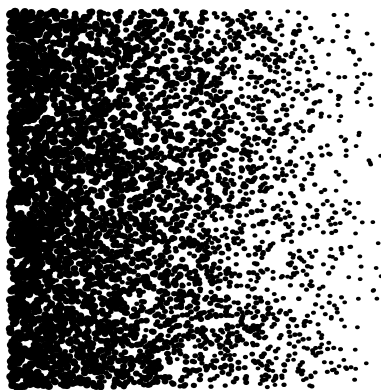


(a) Stipple drawing output

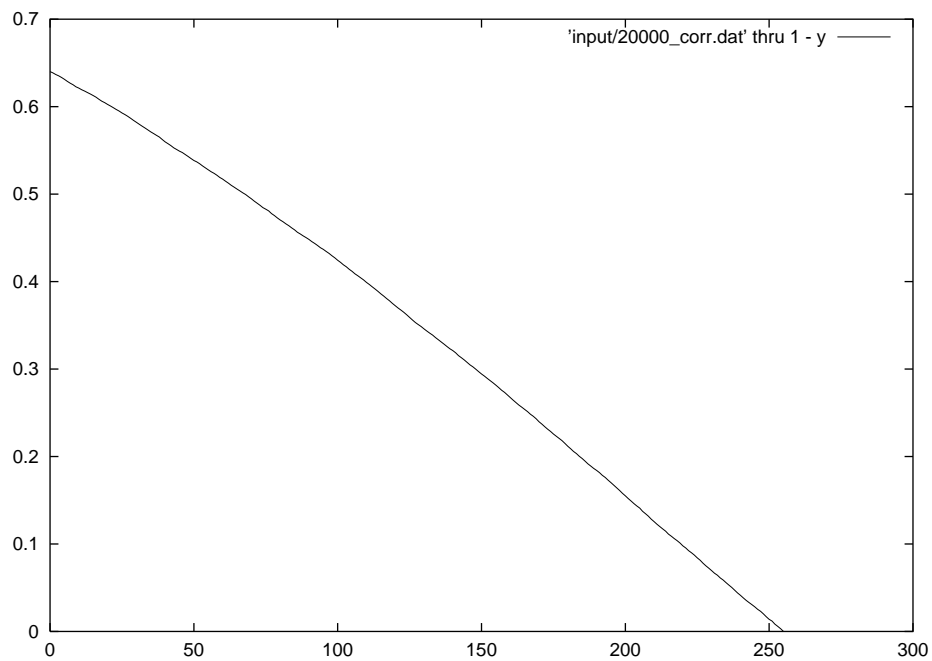


(b) Density of stipple drawing (vertical) versus intensity of a uniform intensity input image (horizontal).

Figure 8: Osculating distribution with 40% jitter of 42 928 lines of radius 0.005 and 21 418 stipples.



(a) Stipple drawing output



(b) Density of stipple drawing (vertical) versus intensity of a uniform intensity input image (horizontal).

Figure 9: Relaxed distribution with 20000 lines and 4972 stipples.

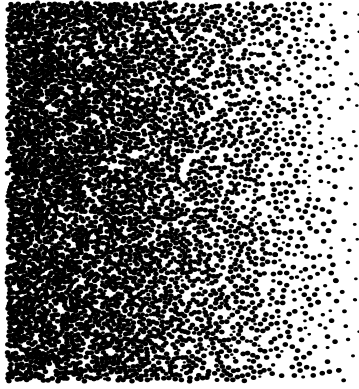
is re-relaxed to create D_2 . The points of D_1 are connected to the points of D_2 at some intensity level $I = I_1$ unless they were deleted, in which case they are simply tapered down to zero radius. The process continues, deleting a fraction of the original points, re-relaxing them to get a new distribution, then extending the previous lines up to the new intensity level. This process is continued until $I = 1$ is reached, at which point all the points should have been deleted and all lines tapered down to zero radius. Note that the ramp is slightly better, but the real advantage is that during animation, the stipples shift position in the plane to maintain good separation from each other. This simultaneously reduces visual popping and means that the distribution is good for all levels of intensity. Note, however, that the computational cost of creating the distribution is much higher due to the repeated relaxations. The stipple radii have also been varied so to reduce the 'popping' that happens when the image intensity moves past the end of the line. This is most noticeable in the animations.

Compensating for Non-Linear Intensity Responses

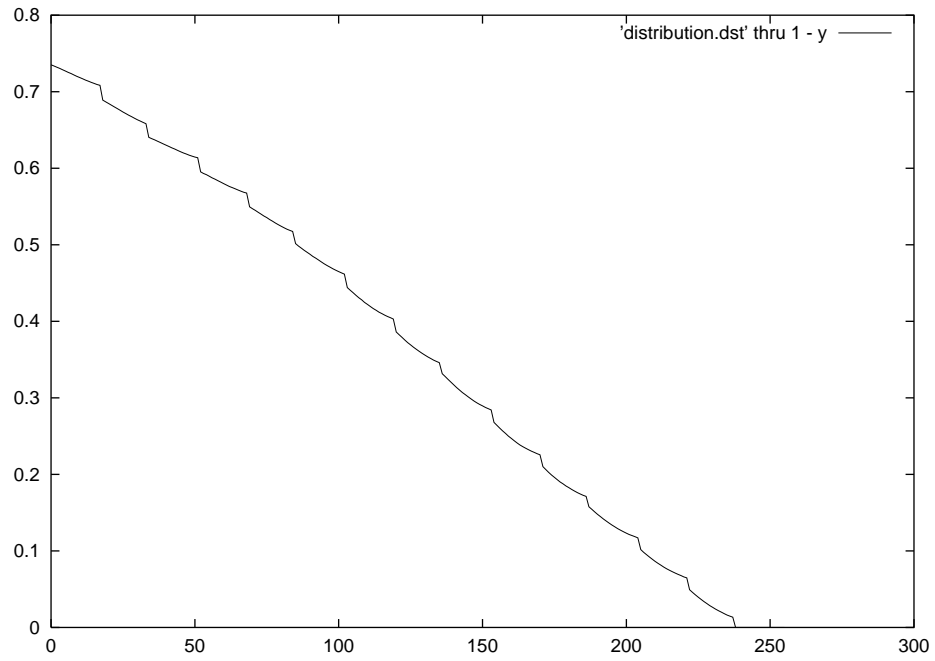
In the previous sections it was found to be quite difficult to create a distribution of lines that simultaneously produced well-spaced stipples and a linear intensity response. With a non-linear response the stipple image typically has much larger shadow regions or highlights than the input image.

To compensate automatically for the intensity response of a particular distribution, we calculate a map of input intensities to corrected intensities such that when use the corrected intensities the resulting stipple image has linear intensity response. To construct the map we stipple and render an artificial image of constant intensity at several discrete values I_i . For each resulting output image, we average the values of all the pixels to get an average intensity O_i . If the intensity response of the distribution is linear then $I_i = O_i$ for all i . If not, then the corrected intensity I_i/O_i will produce I_i when passed through the system. We store the input intensity I_i and the corrected intensity I_i/O_i in a table. When stippling an image after the correction table has been created, we simply map the input intensities through the correction table to get a linear intensity response.

Figure 11 shows both corrected and uncorrected versions of the ramp with a Sobol distribution. Note first that the corrected version has an intensity closer to $1/2$ at the centre than the uncorrected version. However, the correction map has stretched the dark regions of the image. This is because the distribution does not reach true black at intensity zero, but only 85% black. Further work needs to be done to correct for both the non-linearity and this 'non-black' black level.

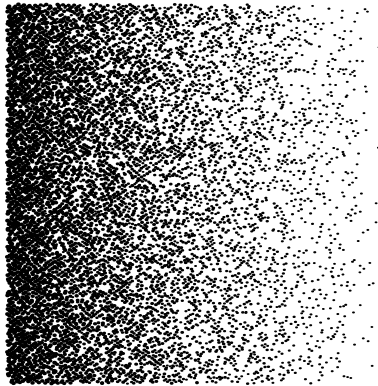


(a) Stipple drawing output

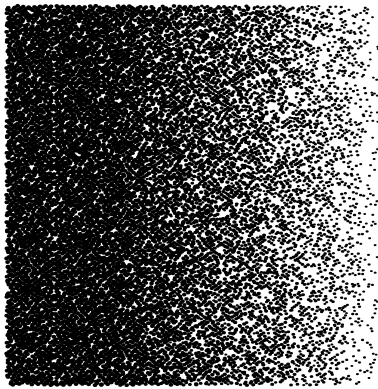


(b) Density of stipple drawing (vertical) versus intensity of a uniform intensity input image (horizontal).

Figure 10: Full relaxed distribution with 10000 lines, 16 levels and 4670 stipples.



(a) Uncorrected



(b) Corrected

Figure 11: Uncorrected and corrected intensity responses for Sobol random distribution of 15000 lines of radius 0.005.

Example Images

The following are a set of example images showing the current results on static images.

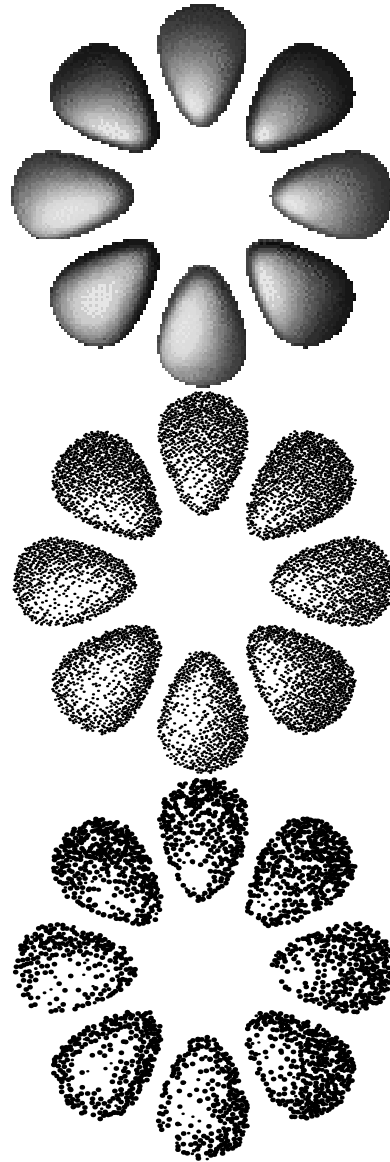


Figure 12: “Well” reference image and stippled version with ordered Sobol distribution, 30 000 lines, 7877 stipples of radius 0.003 and full relaxed distribution, 10 000 lines, 2469 stipples of radius 0.006.

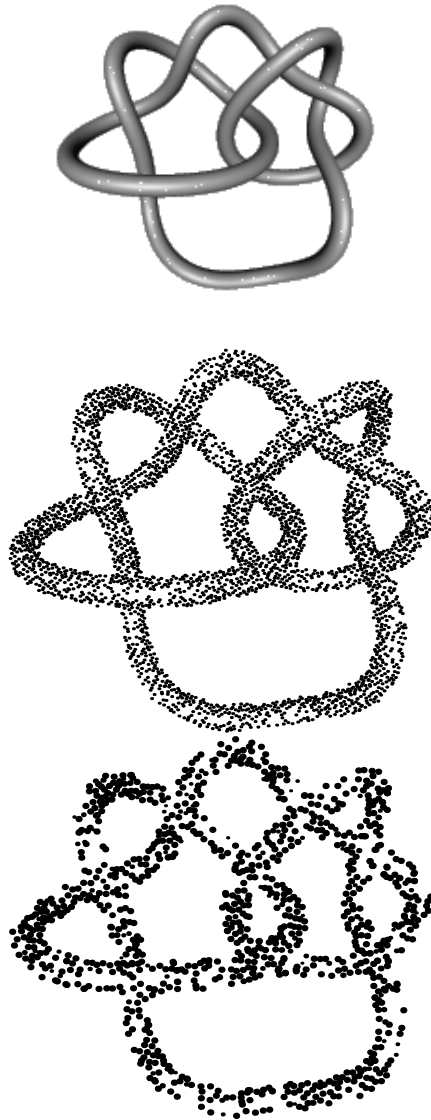


Figure 13: “Knot” reference image and stippled version with ordered Sobol distribution, 30 000 lines, 3477 stipples of radius 0.003 and full relaxed distribution, 10 000 lines, 1180 stipples of radius 0.006.



Figure 14: Statue head reference image and stippled versions with ordered Sobol distribution and full relaxed distribution, both with 10 000 stipples and radius 0.006.

Implementation

The implementation uses C++ as the base language, OpenGL and GLUT for display and interface, GNU flex and bison for parser generation, ImageMagick++ for image reading and writing. The PostScript output is written natively. There are far too many options to list, so here's the output of the usage message:

```
Usage: viewer [-v[VV] -p -r -g -o OO -t TT -n NN
-b BB -s SS -lLL,BB -d DD -m MM -a AA -j JJ] input_name
-t Set the title to TT for output files
-v Increase verbosity level, optionally by VV steps
-p Print to file and exit
-r Allow simplified (faster) PostScript output
-g Allow simplified (faster) OpenGL output
-n Set number of distribution points to NN
-o Save output to file name OO
-b Set border size to BB. Range 0% - 50%. (in percent)
-s Force stipple size to SS. Range 0.0 - 1.0.
-l Force # levels to LL, # bins to BB
-j Add jitter JJ to distros that support it. 0% <= JJ < 100%
-d Use distribution DD where DD can be:
  1 for uniform random
  2 for randomly selected Sobol
  3 for ordered Sobol
  4 for osculating (perfect for black levels of image)
-m Use calculation method MM where MM can be:
  1 for naive method --  $O(n^2)$  in time
  2 for discrete method
-a Display an animation AA where AA can be:
  1 for flat square changing intensity
  2 for ball with rotating light source
```

Speed

The actual intersection code, if done naively, can be very slow, $O(mn)$ in time where m is the number of pixels and n is the number of lines. Instead, we precompute the stipples produced by the lines at each possible intensity level

represented in the image. In general this will give us 16 or 256 sets of stipples. Furthermore, we spatially divide the stipples into bins equal in size to a single horizontal row of pixels. As we scan the pixels of the input image, we select the appropriate set of stipples based on the current pixel value and then check if any of the stipples in the current bin falls within the pixel's bounds. The current bin is always just the current row of the image. This method is $O(m\sqrt{n})$ with a much smaller constant than before.

To test the performance, some simple animation options are included. Figure 15 shows a single frame of an animation of a ball with a light source rotating horizontally around it. Typically the animation runs around 15 frames a second for a 300 by 300 sized screen. This should be improvable, especially if we can find a better method of intersecting lines to a pixel. The above method is still brute force within a particular bin.

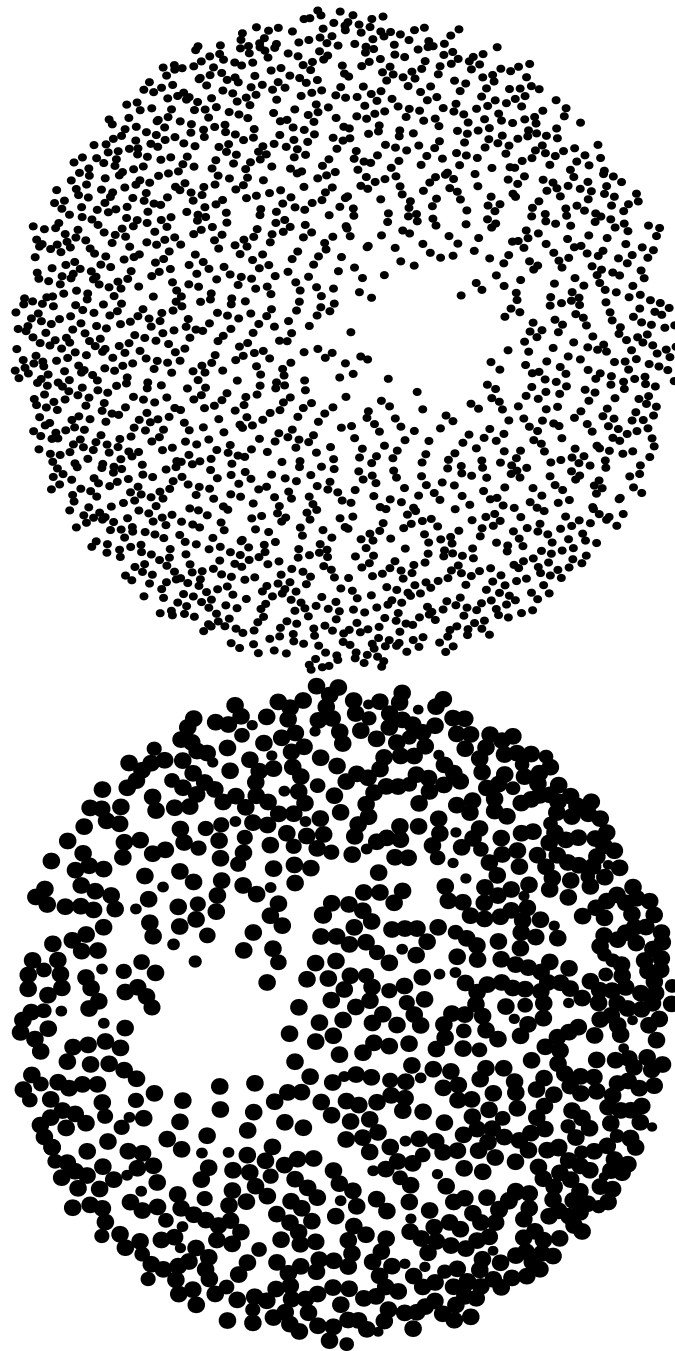


Figure 15: Stippled drawing of a frame of the ball animation with 20 000 lines of radius 0.003 and a different frame using full relaxed distribution of 10 000 lines of radius 0.006.

Future Work

The current main weaknesses of the approach are lack of quality in the stippled drawings (compared to an artist's results) and the adjustment of parameters to achieve a desired result. The quality of the images can be improved by working with different relaxed distributions. In particular it would be interesting to attempt to mimic the artist's point-by-point choice of stipple placement instead of generating a full distribution and filtering it.

The parameter choice problem is tractable through further analysis. The two main parameters now are the number of lines used in a distribution and the stipple radius. Together they constitute a control over the 'grain' of the resulting image and should definitely be combined in some fashion. The osculating distribution already combines the two because there is an analytical expression mapping the radii of osculating discs to the number that can fill the unit square. A more general solution for all distributions is needed.

Many practical and usability issues remain with the code. It desperately needs a better user interface, since it has left the realms of batch processing far behind. The relaxed and fully relaxed distributions are very new and are not particularly well integrated with the rest of the package. They definitely need more investigation into particular values for the parameters of the relaxation process. A library of distributions of various sorts should be built.

The speed of the code should be improved if it is to be used with anything more than toy animations. The current speed is sufficient for images of considerable size, but maintaining an acceptable frame rate during animation is more difficult. Ideally it would be nice to be able to bolt the code on to any existing package that can generate images and convert them into stipples. Imagine an architectural walk-through or Quake in stipples!

Bibliography

- [1] P. Wood. *Scientific Illustration*. Van Nostrand Reinhold, second edition, pages 31-43, 1994.
- [2] Z. Jastrzebski. *Scientific Illustration*. Prentice-Hall, pages 8, 42, 112-113, 271, 1985.
- [3] E. Hodges, ed. *The Guild Handbook of Scientific Illustration*. Van Nostrand Reinhold, pages 91, 109-111, 354, 1989.
- [4] O. Deussen, S. Hiller, C. van Overveld, T. Strothotte. Floating points: A method for computing stipple drawings. In *Proceedings of Eurographics 2000*, 19(3), 2000.
- [5] R. Eschbach. Pulse-density modulation on rastered media: combining pulse-density modulation and error diffusion. *Journal of the Optical Society of America*, 7(4):708-716, April 1990.
- [6] V. Ostromoukhov. Pseudo-random halftone screening for color and black and white printing. In R. Eschback, editor, *Recent Progress in Digital Halftoning IS&T*, pages 130-134, 1994.
- [7] V. Ostromoukhov. Digital facial engraving. In *SIGGRAPH 99 Conference Proceedings*, Annual Conference Series, pages 373-378. ACM SIGGRAPH, Addison Wesley, 1999.
- [8] B. Meier. Painterly rendering for animation. In *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, pages 477-484. ACM SIGGRAPH, Addison Wesley, 1996.
- [9] M. Salisbury, M. Wong, J. Hughes, D. Salesin. Orientable textures for image-based pen-and-ink illustration. In *SIGGRAPH 97 Conference Proceedings*, pages 401-406. ACM SIGGRAPH, Addison Wesley, 1997.
- [10] M. Kowalski, L. Markosian, J. D. Northrup, L. Bourdev, R. Barzel, L. S. Holden, and J. F. Hughes. Art-based rendering of fur, grass, and trees. In *SIGGRAPH 99 Conference Proceedings*, Annual Conference Series, pages 433-438, ACM SIGGRAPH, Addison Wesley, 1999.