Computational Aesthetics in Graphics, Visualization, and Imaging (2007)
D. W. Cunningham, G. Meyer, L. Neumann (Editors)

Stipple Placement using Distance in a Weighted Graph

David Mould[1]

[1]University of Saskatchewan

Abstract

We present a stipple placement method which provides extra emphasis to image features, especially edges. Our algorithm transforms an image into a regular graph, with edge weights given by the local gradient magnitude of the input image. Then, a variant of Disjkstra's algorithm is used to place stipples: new stipples are placed along the frontier, which tends to be aligned with image edges. The resulting stipple distribution approximates a blue noise distribution, but emphasizes edges from the input image. We also show how irregular mosaics can be organized and the mosaic tiles shaped using the same underlying mechanism.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Line and Curve Generation

1. Introduction

Stippling is a process whereby an image is constructed from a large number of small dots. In computer graphics, stippling has been applied to the problem of halftoning: approximating a continuous-tone image with low color resolution (in this case just black and white) but high spatial resolution (large numbers of dots). Pioneering work by Secord [Sec02] did an excellent job of tone reproduction with stipples, and recent work by Kopf et al. [KCODL06] has provided the ability to produce millions of stipple points per second.

However, human artists use stipples not only for tone reproduction, but also to illustrate features, including edge emphasis and texture indication [Gup76]. Also, placing stipples by hand is a time-consuming process, and handmade stipple drawings contain fewer stipples than modern computer-generated stipple images. In this paper, we concentrate on improving the quality of image reproduction by moving away from a strict halftoning regime.

Existing algorithms inherently integrate darkness (more generally, importance) over a two-dimensional area, while the features needing emphasis are often one-dimensional, such as edges. Our proposed approach in effect takes path integrals, where one-dimensional features can easily be resolved; there is a point at which the path crosses the feature. In the area integral formulation, however, nonzero importance away from the feature of highest interest will drag the stipple away. It is not clear how to resolve the issue within a pure halftoning framework – hence our proposed method.

We present a novel algorithm for primitive placement in the plane, and show its application to stipple generation. The algorithm takes an input image, say a photograph, and outputs a stippled version. Our method operates by computing a least-cost path planning operation through a graph representation of the image, and progressively placing sites of distance zero whenever the length of the shortest path exceeds a threshold. This process enforces a minimum distance between sites (since the path cost must exceed the threshold) and gives even small-scale features considerable influence over site placement: some path reaches every feature, or crosses it, in the case of linear features (i.e., edges).

Having shown the use of our progressive algorithm for stippling, we then demonstrate a variant of our algorithm which can produce an irregular tesselation of the plane constrained by an input image. Such tesselations are akin to historical pebble mosaics and chip mosaics [Lin98, Dun99].

This paper is organized as follows. Section 2 reviews previous work in computer-generated stipple and mosaic art, as well as other relevant literature. Section 3 describes our stippling algorithm, and in Section 4 we show stippled images resulting from our algorithm and compare them against stipplings produced using Secord's weighted Voronoi stippling (WVS). Next, in Section 5 we briefly discuss application of

© The Eurographics Association 2007.

delivered by EUROGRAPHICS DIGITAL LIBRARY www.eg.org diglib.eg.org

our algorithm to the problem of irregular mosaics. Finally, in Section 6 we conclude and suggest future work.

2. Previous Work

Automatic stippling methods have employed Centroidal Voronoi Diagrams (CVD's) to place points. CVD's are often achieved by Lloyd's algorithm [O'R90], a relaxation process that repeatedly moves the Voronoi centres to the centroids of their regions. The CVD process has formed the basis for considerable work in stipple creation, as well as mosaic tile placement [DHvOS00, Hau01, Sec02, HHD03], since it is a good way to distribute points on the plane. In particular, the weighted Voronoi stippling (WVS) of Secord [Sec02] provides a benchmark for high-quality stipple halftoning. WVS moves Voronoi centres to the centre of mass of the regions, where local density is given by darkness or some other measure of importance.

Approaches for blue noise generation have been the subject of recent work [DH06, KCODL06, ODJ04]. These results drastically speed up the stippling process, and have provided the ability to generate millions of stipple points per second. However, while these results can be used for halftoning, they do not attempt to treat sharp features. As we argued in the introduction, sharp features are inherently underemphasized by area-based importance matching.

Mosaics share with stipples the problem of distributing primitives, and have received substantial attention. Hausner [Hau01] used hardware-accelerated CVD's to distribute tiles. Hausner also identified a crucial issue in mosaics: that tile edges should be aligned with image edges, and resolved this in his work by having tiles move away from user-specified edges. An alternative method for achieving edge alignment was given by Elber and Wolberg [EW03], who arrange rows of tiles along streamlines parallel to initial user-specific curves. Yet another way of resolving it was given by Di Blasi and Gallo [BG05], who proposed cutting the rectangular tiles where they crossed image edges. Kim and Pellacini's Jigsaw Image Mosaic [KP02] produces an irregular tiling of the image plane with predefined tiles. The Jigsaw Image Mosaic algorithm strives to pack user-specified regions with a known set of tiles, minimizing a set of error criteria including tile overlap and color mismatch; it gives good results, but is resource-intensive and is not easy to extend, as was also observed by Dalal et al. [DKLS06].

The RenderBots postulated by Schlechtweg et al. [SGS05] are a general method for distributing primitives and have been applied to stipples. RenderBots distribute stipples equivalently to CVD in homogeneous regions. The authors propose attracting stipples towards image edges to increase edge emphasis, but sharp edges are not conveyed by the resulting stipples: instead, an increasing density of stipples appears in the vicinity of the image edge.

Other methods for distributing primitives and tiling the plane have been devised, and we briefly mention a few others. Smith et al. [SLK05] focussed on coherent movement of tiles to create animated mosaics; later, Dalal et al. [DKLS06] used Fourier transforms to find good packings of input primitives. Kaplan and Salesin [KS00] worked on automatically controlling tile shapes to produce Escher-like tilings where the tiles were close to an input goal shape.

3. Algorithm

For greatest generality, we concentrated on stippling grayscale images. We use gradient magnitude as a measure of local contrast, similar to Winnemöller et al. [WOG06]; any other measure of importance can be substituted instead.

Our method uses the same underlying mechanism as Mould's image-guided fracture [Mou05]. Mould employed multiple-source graph search (brushfire) to obtain region boundaries aligned with image features. However, that work did not attempt to address site placement other than trivially. Making the observation that the frontier consists largely of nodes of high cost, and identifying high cost with image features, we can notice that the frontier is the best area for site placement. We developed what we call a "progressive" algorithm, described as follows.

We first construct a graph from an input image: a regular eight-connected lattice with one node per input image pixel. Importance is computed for each node as $\alpha d + \beta g$, where d is the darkness (1-intensity) and g is the gradient magnitude; α and β are normalizing factors, where we had $\alpha = 1/\sum d$ and $\beta = 1/\sum g$ – that is, the inverses of the respective values summed over the entire image. Edge weights are the average importances of the pixels on either end of the edge.

The progressive algorithm begins with a single site (say, one corner of the image) at distance zero and floods outward, computing shortest paths in the graph. When the shortest path exceeds a threshold length, a new site at distance zero is placed; we continue until the entire graph is explored.

Shortest paths are computed using Dijkstra's algorithm. In the remainder of this section, we give a description of Dijkstra's algorithm, followed by additional details of how we modified the algorithm to perform progressive site placement, resulting in what we call the *progressive algorithm*. Once the progressive algorithm chooses stipple sites, we produce our final images by placing uniformly-sized dots at all sites. In this work, we are concentrating on stipple placement; naturally, stipple size is a dimension that can be employed, but in presenting our results this way we are attempting to remove distraction and encourage the reader to focus on the stipple positions.

3.1. Dijkstra's Algorithm

Dijkstra's algorithm [CLR90] computes shortest distances from a source point to the other nodes in the graph. Each

© The Eurographics Association 2007.

node N has an estimated distance to the source, written $E(N)$, and an actual distance, written $A(N)$. Dijkstra's algorithm uses a priority queue to store a set of "frontier" nodes: nodes with unknown actual distance, but which lie adjacent to nodes with known actual distances. The priority queue is ordered by increasing estimated distance, with smallest distances at the top; it is often implemented as a heap. Initially, the frontier contains only the source node, at estimated distance zero; the graph is initialized with all nodes' estimated distances set to infinity. Then, the following is repeatedly performed. First, the top node, say T, is removed from the frontier, and its actual distance set to its estimated distance. Second, for each node N_i adjacent to T, a new estimate is computed $E_{new}(N_i)$ as $A(T)$ plus the cost of the edge linking N_i and T; those nodes whose old estimate exceeds E_{new} have their estimate replaced by the new estimate and are added to the frontier. Dijkstra's algorithm terminates when the frontier is empty, at which point all nodes in the graph have known distance values.

To find the distances to n nodes, Dijkstra's algorithm makes $O(n)$ removal operations. Maintaining the heap has a cost of $O(\log m)$ per interaction, where m is the number of nodes in the frontier; thus, the overall cost is $O(n \log m)$. Typically, $m << n$; that is, only a small fraction of the graph is in the heap at any one time.

3.2. Progressive Site Placement

We propose a fast, non-iterative algorithm for placing stipples. Site are placed progressively, building up a stippled region and adding new stipples at its fringe. The algorithm works as follows.

An initial seed is placed at some location in the image. Dijkstra's algorithm is used to expand a region around the seed; when the path cost for a newly expanded node exceeds a threshold, the expansion step triggers the addition of a new stipple. The new stipple location is the frontier node with highest gradient magnitude, since we want to place stipples preferentially on image edges. Crucially, the distance of the new site is set to zero, so that expansion occurs initially in the vicinity of the new site (since all other nodes in the heap have distance at least the threshold). When no part of the graph remains to be expanded, the progressive algorithm terminates.

The progressive tile placement is illustrated in Figure 1: first, a region is expanded about the first centre; a new centre is chosen on the frontier; the new region is expanded until the threshold is reached again; a third centre is placed, and it expands; and so on. In the figure, the black outlines indicate the sets of nodes nearest the central stipple, analogous to Voronoi regions. At the time of placing a new stipple, the frontier is the boundary between the grey and white areas.

The above discussion glosses over the details of how to choose the highest gradient magnitude node from the frontier. A naive approach to this task would require linear search of the entire frontier. We instead implement a second heap, ordered by gradient magnitude; when we extract a node from this heap, we discard stale references until we reach a node on the current frontier. Although this approach requires a little extra memory, it makes the extraction process quite fast.

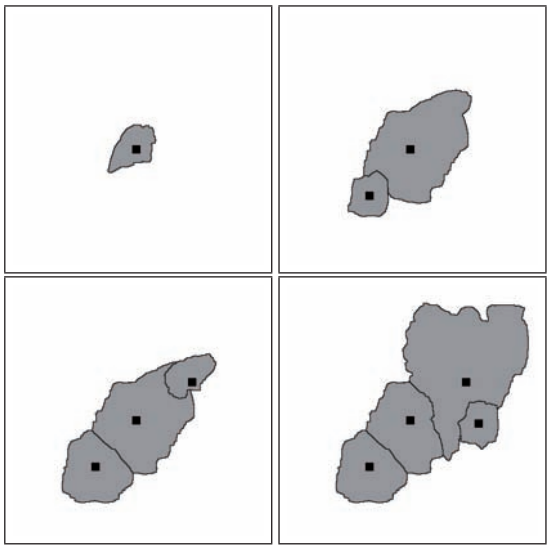

Figure 1: *Progressive site placement. New sites are placed on the frontier when the cost of the last expanded node exceeds a threshold.*

One feature of this algorithm is that the least-cost path between any two stipples is at minimum the chosen threshold. This property arises directly from the use of the frontier to place sites. At the time a new stipple is placed (triggered by the expansion of a node whose cost exceeds the threshold) all nodes on the frontier are at least the threshold distance away from the nearest stipple. Distributions with a minimum separation between point are often identified as "blue noise".

4. Results

Here, we give results and describe some advantages and disadvantages of our system. First, we show the effect of applying our technique to a greyscale ramp, in Figure 2. The range of tones is compressed, because we have in effect reserved half our dynamic range for conveying edge information rather than tone. Nonetheless, multiple tones are clearly visible. In real images, especially photographic images, edges are common and the ability to express both intensity and edge information is invaluable.

We next show results from applying our method to different input images; the originals of six of these images are shown in Figure 3. Two stippled images are shown in Figure 4. The cat image we consider a success: despite the low

© The Eurographics Association 2007.

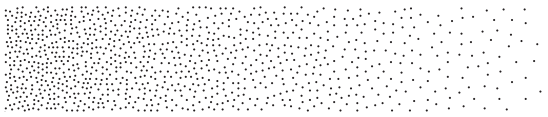

Figure 2: *A greyscale ramp, halftoned with the progressive algorithm.*

Figure 3: *Original images. Top: cat, old man; middle: subway, sunflower; bottom: Lena, artificial.*

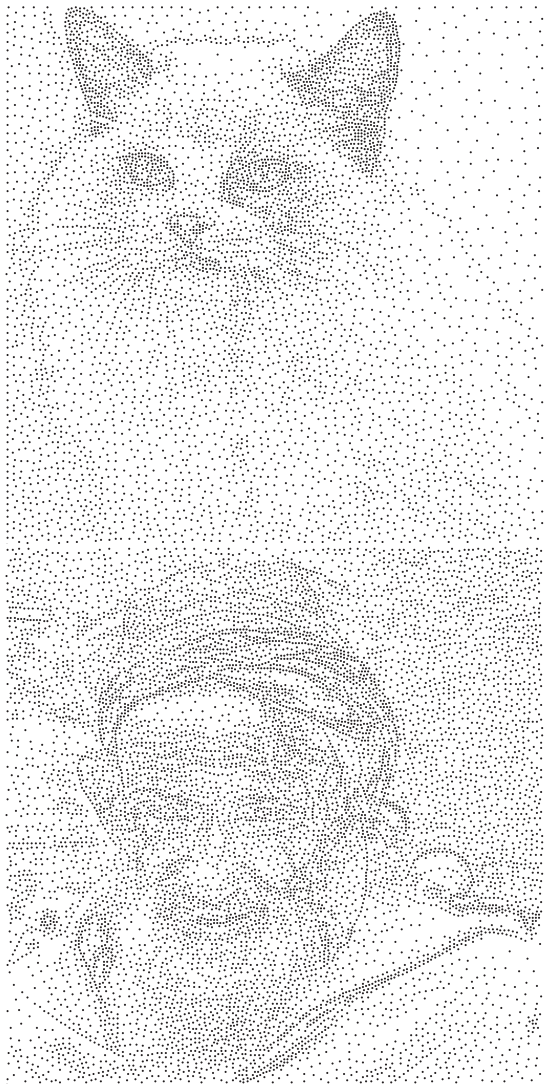

Figure 4: *Above: stippled cat, 5600 points. Below: stippled old man, 9000 points.*

stipple count (about 5600) we are able to convey the image content quite successfully. The cat image is fairly low-contrast, so a difficult case for halftoning. Image edges, including details such as the whiskers and the fur direction in the ears, have been preserved. The old man image is at best a qualified success. The prevalence of high-frequency detail, including wrinkles and facial hair, make this a challenging subject; despite the larger number of stipple used (about 9000) the features are not expressed clearly.

In Figure 5 we seek to show the results of employing our algorithm with varying stipple counts. With very low stipple count (under 1000) details are not visible, and the scene is just recognizable by someone familiar with the photograph. With medium stipple count (3000) details begin to emerge, and lines such as the jawline, forehead, and structure of the wall to the image right are successfully described. With relatively high numbers of stipples (11,000) there remains considerable edge emphasis, but this quantity of stipples also allows us the luxury of genuine halftoning, and details such as the shading on the shoulder can be indicated.

We make direct comparisons with the results from Secord's WVS in Figures 6, 7, 8, 9, and 10. Although an older algorithm, WVS has not been improved on in dimensions relevant to our proposed method; extensions have concentrated on generalizing the stipple shape, adding temporal behaviour, and accelerating stipple placement.

Weighted Voronoi stippling matches tone superbly. However, as previously stated, stipples are not used solely for tone reproduction, but to indicate image features. When

© The Eurographics Association 2007.

Figure 5: *Subway image with different stipple counts. Leftmost: 945 points; middle: about 3000 points; rightmost: about 11000 points.*

halftoning alone is used, moderate stipple counts have difficulty conveying sharp features.

We are not restricted to faithfully matching the tones in an input image, however, and can instead construct an importance image that gives higher weight to areas such as edges. In principle it should always be possible to construct an importance map so that tone matching the importance image yields the desired stippling result. In practice, however, it is not clear how the importance map should be built. The right of Figure 6 shows one effort to stipple an importance image; here, the importance of a pixel is a weighted average of darkness and gradient magnitude, with global weights chosen such that the integral of gradient magnitude times its weight equalled the integral of darkness times its weight. Stippling of this importance map yielded a better representation of the input image, but still not a very good one.

We found the best way for WVS to represent the Lena image with low stipple counts was to abandon darkness entirely. Figure 7 shows a direct comparison between progressive stippling and WVS of the gradient magnitude. Here, the choice of which is better is not altogether clear-cut. However, we can point to elements of the progressive image with better detail: the separation of the forehead and hat; the lines across the crown of the hat; the structures on the wall behind Lena, especially directly above Lena's shoulder. Also, the WVS image no longer displays tone: darker areas (such as the curved rim of the mirror, in the image's mid to upper right) receive the same uniform shading as lighter areas. The progressive algorithm does show a difference in tone there.

Comparisons using the artificial image are also illuminating. Because of the sharp edges and large differences in tone, neither darkness alone nor gradient alone are suitable for importance, as Figure 8 shows. A mix allows WVS to reveal both the shading and the edges, shown in Figure 9. However, although WVS does place stipples on the circular edge, these stipples are excessively integrated into the overall distribution, and (especially in the lower portion of the circle) tend to vanish into the crowd. By comparison, the edge-aligned stipples placed by the progressive stippling method disrupt the pattern, as well as being spaced more closely together, and in consequence are considerably more visible.

Figure 9 also displays a perceptual artifact well-known to be present in uniform Voronoi packings of points [INC*06]: the tendency of the eye to see and follow curved paths through the distribution. Figure 10 shows a comparison between uniform distributions generated with CVD's and with progressive stippling. The progressive stippling result lacks the perceptual defect of the CVD result. It also has a rougher look, difficult to quantify but reminiscent of hand-made placement of dots. Notice that the progressive distribution cannot simply be achieved by jittering the positions of the CVD packing, since it maintains the blue noise characteristic of a minimum distance between any two points; jitter will tend to bring some points closer together than is desired.

In Table 1 we show timing results and exact stipple counts for our images. All results are for 512×512 resolution images on a 3.2GHz P4 with 1 GB RAM; our C implementation was not highly optimized, but stipple results can be obtained in less than 1 second. The times depend mainly on the resolution of the underlying graph, although there is a weak dependence on stipple count as well (since larger numbers of stipples tend to require larger heaps).

5. Mosaic Construction

Mosaics and stipples have a close relationship: the same primitive distribution problems are seen in each. However, while stipples should be placed on image edges, mosaic tile centres should be placed away from edges; the boundaries between mosaic tiles, rather than the tiles themselves, offer opportunity to highlight edges. Mosaics differ from stipples also in that beyond the problem of tile distribution, mosaics have the additional problems of tile shape and orientation. We can modify the algorithm of Section 3 to place mosaic tiles, and employ an identical distance calculation to com-

© The Eurographics Association 2007.

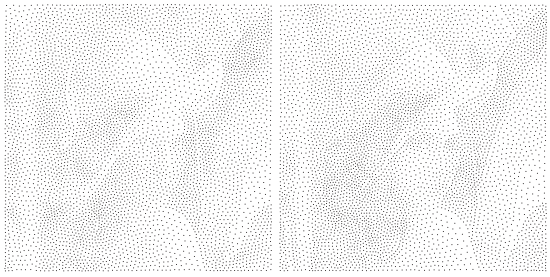

Figure 6: *WVS applied to the Lena image: left, pure halftoning; right, importance is a mix of gradient magnitude and darkness.*

Image	point count	processing time
Cat	5582	0.8s
Old man	9082	0.8s
Lena	4815	0.8s
Artificial	1221	0.8s
Subway (low res)	945	0.7s
Subway (medium res)	2987	0.8s
Subway (high res)	11146	0.9s
Sunflower (mosaic)	1067	1.6s
Lena (mosaic)	1319	1.7s

Table 1: *Timing results for some of the stippling processes.*

pute irregular shapes surrounding tile centres. We present this method as a means of approximating the earliest historical mosaics, pebble and chip mosaics; such constructions used heterogeneous sets of tiles rather than the regular tesserae employed in the archtypal tesselated mosaic, commonly studied by NPR practitioners.

We follow the same basic procedure for mosaic tile placement as for stipple placement: the progressive algorithm is used to place tiles, so that new tile centres placed at the frontier. Now, though, we want to place new tiles at the lowest gradient location on the frontier, rather than the highest. Also, we now care about the size of regions, rather than (strictly) the separation between tile centres. This latter point can be achieved by a straightforward conceptual modification of the algorithm: rather than placing a new centre when the path length exceeds a threshold, we place a new centre when the region size of the most recently placed tile exceeds a threshold. This idea requires an adjustment to the implementation: we grow a single region at a time until it attains the desired size; as a new region grows, it is forbidden from growing into nodes already claimed by previously placed tiles. The regions arising from this process often possess defects (such as holes), and we compute final tile shapes from a second pass of the distance calculation, starting from all tile centres simultaneously, labeling each node with its nearest tile centre.

Figure 7: *Comparison of WVS and PS. Above, WVS of gradient magnitude; below, progressive stippling.*

An example of the boundaries of the initial regions, and the resulting tile shapes, is shown in Figure 11. Some mosaic results from input images are shown in Figure 12. We need fewer tiles than stipples to express an image, since the tile edges convey so much information; the examples shown contain approximately one thousand tiles each. Notice how well the irregular tile edges conform to real image edges, despite the absence of an explicit edge detection step.

6. Conclusions

We have described a novel stippling algorithm arising from progressive distance calculation over a graph representation of an input image. The method can capture both edge and

© The Eurographics Association 2007.

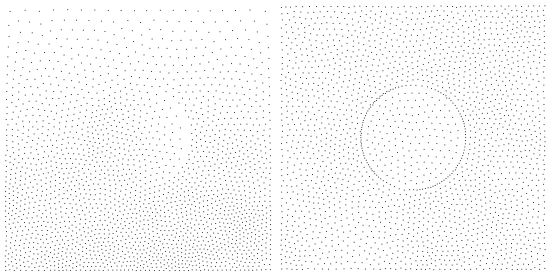

Figure 8: *WVS applied to the artificial image: left, pure halftoning; right, importance is gradient magnitude.*

darkness information, and is better suited to depicting sharp features than are previously proposed methods.

The output stippled images have blue-noise-like characteristics where features are absent, but are also able to show sharp features such as edges. Our output is free of the distracting "chain" artifacts Voronoi packings are prone to. We contend that the progressive algorithm produces higher quality images than does the classic weighted Voronoi method, and that the images also look more handmade.

There are a few directions for future work. At present, texture indication is done only by finding intensity edges. We would like to investigate methods for generating a more coherent depiction of high-frequency texture structure. We would like to further explore and refine the application of this algorithm to irregular mosaics, and to incorporate stipple size changes into our framework. Finally, further investigation of automatic importance computation would be useful in NPR applications including halftoning, painterly rendering, and many other artistic styles.

References

[BG05] BLASI G. D., GALLO G.: Artificial mosaics. *The Visual Computer 21*, 6 (2005), 373–383.

[CLR90] CORMEN T., LEISERSON C., RIVEST R.: *Introduction to Algorithms*. MIT Press, Cambridge, Massachusetts, 1990.

[DH06] DUNBAR D., HUMPHREYS G.: A spatial data structure for fast poisson-disk sample generation. In *Proceedings of SIGGRAPH 2006* (2006), ACM Press, pp. 503–508.

[DHvOS00] DEUSSEN O., HILLER S., VAN OVERVELD C., STROTHOTTE T.: Floating points: A method for computing stipple drawings. *Computer Graphics Forum 19*, 3 (2000), 40–51.

[DKLS06] DALAL K., KLEIN A. W., LIU Y., SMITH K.: A spectral approach to NPR packing. In *NPAR '06: Proceedings of the 4th international symposium on Non-photorealistic animation and rendering* (2006), ACM Press, pp. 71–78.

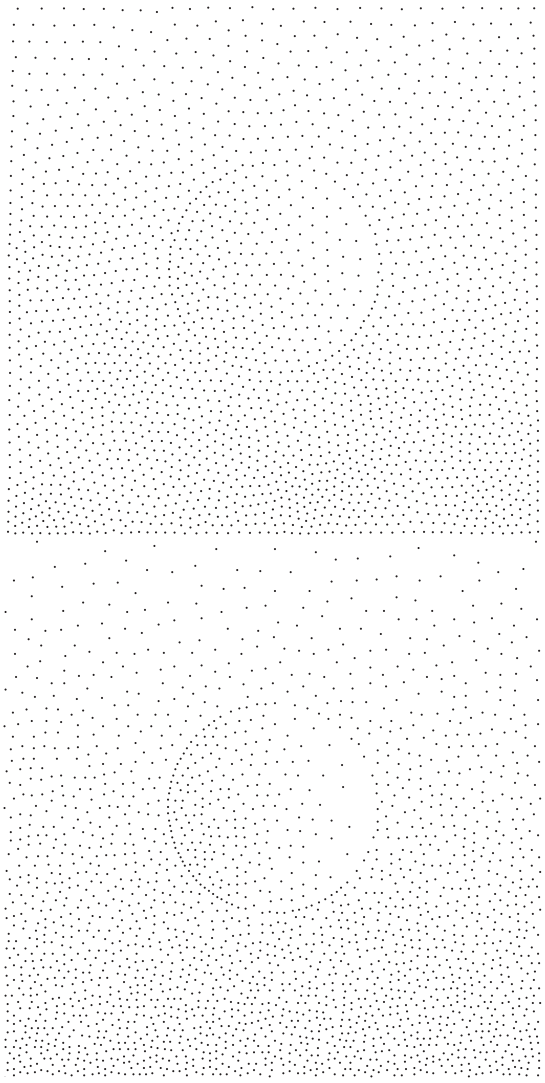

Figure 9: *Artificial image stipplings; above, stippled by WVS with importance an average of gradient magnitude and darkness; below, stippled by PS.*

[Dun99] DUNBABIN K.: *Mosaics of the Greek and Roman world*. Cambridge University Press, Cambridge, UK, 1999.

[EW03] ELBER G., WOLBERG G.: Rendering traditional mosaics. *The Visual Computer 19*, 1 (2003), 67–78.

[Gup76] GUPTILL A.: *Rendering in Pen and Ink*. Watson-Guptill Publications, New York, 1976.

[Hau01] HAUSNER A.: Simulating decorative mosaics. In *Proceedings of SIGGRAPH 2001* (2001), ACM Press, pp. 573–580.

[HHD03] HILLER S., HELLWIG H., DEUSSEN O.: Be-

© The Eurographics Association 2007.

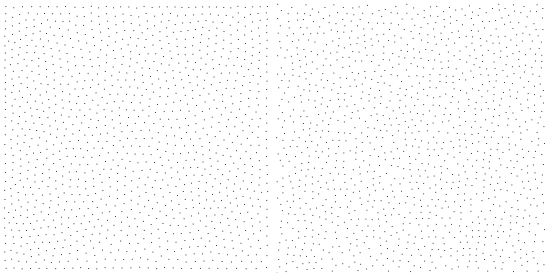

Figure 10: *Comparison of WVS and PS with uniform distribution of dots. Notice the distracting "chain" artifacts Voronoi packing is prone to, absent in the progressive method.*

Figure 11: *Right: initial regions; left: final tile boundaries.*

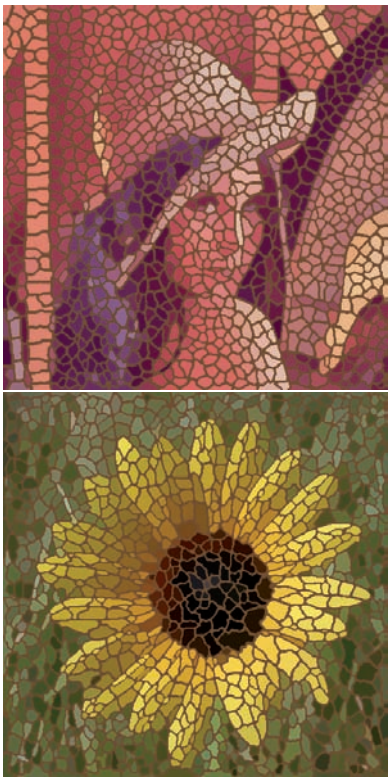

Figure 12: *Irregular mosaics: Lena, above, and the sunflower, below. Notice how image edges and tile edges align.*

yond stippling – methods for distributing objects on the plane. *Computer Graphics Forum 22*, 3 (2003), 515–522.

[INC*06] ISENBERG T., NEUMANN P., CARPENDALE S., SOUSA M. C., JORGE J.: Non-photorealistic rendering in context. In *NPAR '06: Proceedings of the 4th international symposium on Non-photorealistic animation and rendering* (2006), ACM Press, pp. 115–126.

[KCODL06] KOPF J., COHEN-OR D., DEUSSEN O., LISCHINSKI D.: Recursive Wang tiles for real-time blue noise. In *Proceedings of SIGGRAPH 2006* (2006), ACM Press, pp. 509–518.

[KP02] KIM J., PELLACINI F.: Jigsaw image mosaics. In *Proceedings of SIGGRAPH 2002* (2002), ACM Press, pp. 657–664.

[KS00] KAPLAN C. S., SALESIN D. H.: Escherization. In *Proceedings of SIGGRAPH 2000* (2000), Akeley K., (Ed.), ACM Press, pp. 499–510.

[Lin98] LING R.: *Ancient Mosaics*. British Museum Press, London, UK, 1998.

[Mou05] MOULD D.: Image-guided fracture. In *Proceedings of Graphics Interface 2005* (2005), Canadian Human-Computer Communications Society, pp. 219–226.

[ODJ04] OSTROMOUKHOV V., DONOHUE C., JODOIN P.-M.: Fast hierarchical importance sampling with blue noise properties. In *Proceedings of SIGGRAPH 2004* (2004), ACM Press, pp. 488–495.

[O'R90] O'ROURKE J.: *Computational Geometry in C*. Cambridge University Press, Cambridge, 1990.

[Sec02] SECORD A.: Weighted Voronoi stippling. In *NPAR '02: Proceedings of the 2nd international symposium on Non-photorealistic animation and rendering* (2002), ACM Press, pp. 37–43.

[SGS05] SCHLECHTWEG S., GERMER T., STROTHOTTE T.: RenderBots—Multi Agent Systems for direct image generation. *Computer Graphics Forum 24*, 2 (2005), 137–148.

[SLK05] SMITH K., LIU Y., KLEIN A.: Animosaics. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2005), ACM Press, pp. 201–208.

[WOG06] WINNEMÖLLER H., OLSEN S. C., GOOCH B.: Real-time video abstraction. In *Proceedings of SIGGRAPH 2006* (2006), ACM Press, pp. 1221–1226.

© The Eurographics Association 2007.