Journal of Graphics, GPU, and Game Tools

ISSN: 2151-237X (Print) 2151-2272 (Online) Journal homepage: http://www.tandfonline.com/loi/ujgt20

Adaptive Incremental Stippling using the Poisson-Disk Distribution

Ignacio Ascencio-Lopez , Oscar Meruvia-Pastor & Hugo Hidalgo-Silva

To cite this article: Ignacio Ascencio-Lopez , Oscar Meruvia-Pastor & Hugo Hidalgo-Silva (2010) Adaptive Incremental Stippling using the Poisson-Disk Distribution, Journal of Graphics, GPU, and Game Tools, 15:1, 29-47, DOI: 10.1080/2151237X.2010.10390650

To link to this article: http://dx.doi.org/10.1080/2151237X.2010.10390650

Published online: 11 Feb 2011.

Vol. 15, No. 1: 29–47

Adaptive Incremental Stippling using the Poisson-Disk Distribution

Ignacio Ascencio-Lopez
CICESE–Ciencias de la Computación, Ensenada, B.C., Mexico

Oscar Meruvia-Pastor
Department of Computer Science, Memorial University of Newfoundland, St. John's, N.L., Canada

Hugo Hidalgo-Silva
CICESE–Ciencias de la Computación, Ensenada, B.C., Mexico

Abstract. Recently efficient algorithms have been published for generating large point sets with Poisson-disk distribution. With their blue noise spectral characteristics, Poisson-disk distributions are considered to produce visually pleasing patterns. Some applications, e.g., non photo-realistic rendering (NPR), require, in addition to efficiency, the production of aesthetically pleasing point sets adapted to an arbitrary image or function. We present a novel linear order stippling method that generates a set of points with Poisson-disk distribution adapted to arbitrary images and compare this method with existing methods using two quantitative evaluation metrics, radial mean and anisotropy, to assess the technique.

1. Introduction

Sampling with Poisson-disk (PD) distribution is used in a variety of computer graphics contexts such as antialiasing, ray-tracing, Monte Carlo path-tracing, geometric processing, point-based rendering, digital halftoning, object positioning, procedural texturing and primitive graphics positioning in

1086-7651/09 $0.50 per page

non-photorealistic rendering. In these areas it is important to generate a distribution free of sampling artifacts. The PD distribution is closely associated to blue noise characteristics in the Fourier spectra. It is generally accepted that blue noise properties in the PD distribution make for some of the best sampling patterns for a wide range of applications (see [Lagae and Dutré 08, McCool and Fiume 92, Yellot 83, Klassen 00, Meruvia and Strothotte 04, Ostromoukhov 07, Feng et al. 08, Vanderhaeghe et al. 07]).

Existing algorithms are able to produce huge quantities of PD points in seconds [Cohen et al. 03, Lagae and Dutré 05, Kopf et al. 06, Dunbar and Humphreys 06, Jones 06, Bridson 07, Wei 08], but these algorithms do not reach the level of quality attained by the well established centroidal Voronoi diagrams (CVDs) methods [Deussen et al. 00, Secord 02], which have better characteristics in terms of mean square error (MSE) and blue noise properties.

The principal intent of this paper is to contribute a new method to produce stippled images of higher quality than those produced with high-throughput methods (such as Wang Tiles) with similar quality to images produced through CVDs, but in less time than what is typically required for CVDs.

The proposed method, adaptive incremental stippling (AIS), generates PD samples according to an input image (or an arbitrary density function, for that matter) in linear time and without the need of preprocessing. This allows the generation of stippled images with desirable blue noise characteristics and a clear edge definition obtained from the image gradient.

Since methods that generate point distributions are typically evaluated using spectral characteristic indicators such as radial mean and anisotropy, we assessed all the methods presented with these two indicators.

Adaptive incremental stippling starts with a first seed corresponding to a disk with a diameter related to the image density in the seed's position. New disks of area proportional to the image density are generated around the seed-disk and more disks are generated around the initial disks until the whole plane is covered. To improve contour definition, the disks are defined considering not only the image density but also the gradient map associated to image edges. The method is successful in highlighting features such as tone intensity, contrast, and edges of the input image.

2. Generating PD-Point Distributions

2.1. Characteristics and Specifications

The Poisson-disk distribution is a uniform distribution of points in a plane where all points are at a minimum distance from each other. Half of this minimum distance is known as the *distribution radius*. A group of points will have a Poisson-disk distribution if a disk with the corresponding radius is placed

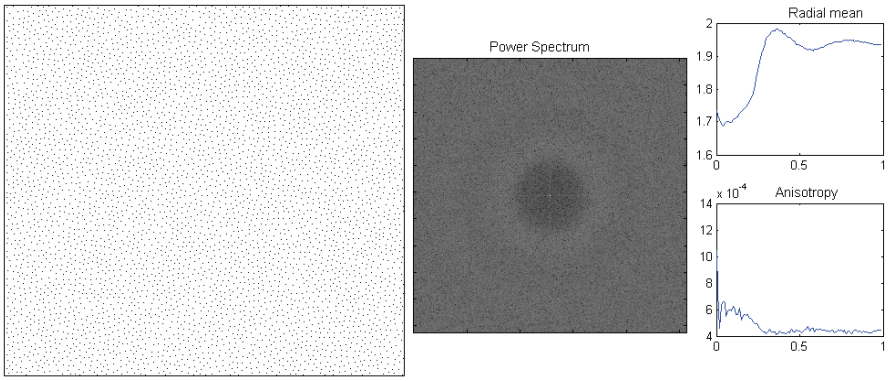


Figure 1. A Poisson-disk distribution generated by relaxed dart-throwing.

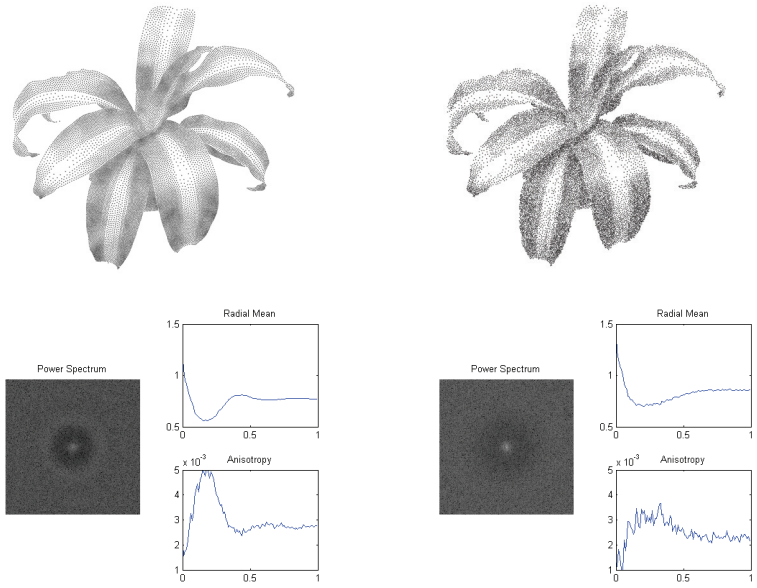
in each position without overlapping neighboring disks. The characteristics of a point distribution may be analyzed in the frequency domain using the Fourier transform [Ulichney 88, Lagae and Dutré 08, Klassen 00].

Figure 1 shows the PD distributions generated by relaxed dart-throwing followed by Lloyd's iterations. Spectral radius statistics are the most important criteria used to evaluate the quality of PD generation algorithms [Ulichney 88]. The radial mean of the power spectrum of a PD distribution shows a specific structure [Lagae and Dutré 05]. The initial peak is followed by a low energy region; this corresponds to the lower frequency disk radius. This is followed by a transition to the flat region of higher frequencies containing most of the energy.

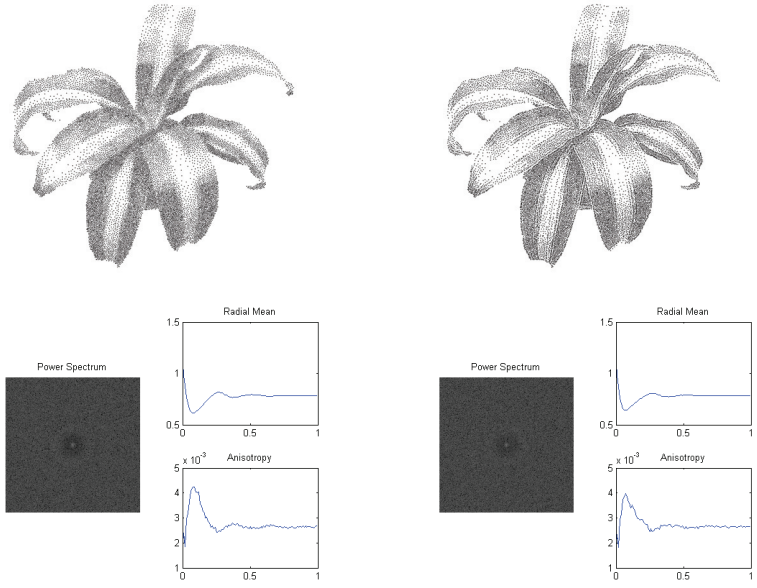
Other quality measures based on the use of the grey level co-occurrence matrix (GLCM) have been proposed [Maciejewski et al. 08] to characterize the textures produced by stippling images. However, a careful appraisal showed that these measures are dependent on the amount, form, and size of the primitive graphics used (circles, irregular-shaped dots, etc.), rather than on their distribution, and thus their use is beyond the scope of the present work.

2.2. Poisson-Disk Distributions Associated with an Arbitrary Density Function

Deussen and Secord both present methods for point generation with an arbitrary density [Deussen et al. 00, Secord 02]. In both works a stippling image is generated using Lloyd's method. They involve the calculus of CVDs, and this quantization has optimality characteristics in terms of MSE [Wong 05]. Stippling images generated with this method present the typical anisotropy and low radial mean forms shown in Figure 2(a).



(a) CVD [Secord 02] (b) WAT [Kopf et al. 06]



(c) AIS (d) AIS-E

Figure 2. Four methods for stippling and statistical characteristics on the Fourier spectrum for each method.

The CVD is computed in $O(Kn \log n)$ time using the most efficient algorithms (divide-and-conquer strategies), where n is the total number of pixels in the density function and K is the number of passes to achieve convergence. Graphics Processing Unit implementations of CVDs significantly reduce the computation time. However, CVD-based algorithms require high-resolution image buffers to avoid artifacts and to produce the expected results; so, in spite of being able to be executed on the GPU, they still take a long time to finish. One of the advantages of incremental stippling algorithms, such as the one introduced here, is that they have much shorter execution times, allowing for faster response times when fine-tuning the input parameters to try to get a particular result.

On the other end of execution times for stippling renditions, Kopf et al. can generate point sets at rates of millions of points per second using pre-computed tile bases, adequately conveying the intensity of the input image [Kopf et al. 06]. In exchange of this high throughput, the images are not as compelling as the ones obtained through CVDs. There is a loss of definition at the edges of the regions that convey the shapes of the images and, as noted by Ostromoukhov in [Ostromoukhov 07], Wang-tile-based stippling produces a noisy stipple distribution (see Figure 2(b)). As an alternative, Ostromoukhov presents a fast tile-based algorithm that generates PD distributions much like those achieved with CVD-based solutions, which is designed to generate sampling points and diffuse gradient transitions, rather than to convey the sharp boundaries and contours characteristic of stippling renditions.

Using a parallel implementation on GPU, Wei [Wei 08] presents a method that generates up to four million points per second with similar spectral characteristics to Kopf et al.. Their Poisson-disk sampling algorithm subdivides the sample domain into grid cells and draws samples concurrently from multiple cells without inter-cell conflicts. They use a hierarchical n-dimensional tree structure for adaptive sampling.

Mould presents a progressive $O(n \log m), m << n$ stipple placement method using a weighted graph [Mould 07]. He transforms an image into a regular graph, with edge weights given by local gradient magnitude and tone. The method operates by computing a least-cost path planning operation through a graph representation of the image, and progressively places stipples whenever the length of the shortest path exceeds a threshold. The method integrates the image intensity and gradient information, and is thus well suited to convey sharp features. In the same way, Adaptive Incremental Stippling can also incorporate such sharp features and in Section 4 we show how the image definition improves at the edges.

There are other approaches to stippling that are frame-coherent and based on 3D geometry or video input. Meruvia and Strothotte present an algorithm to create a hierarchy of PD point sets using a variation of Lloyd's method on a 3D graph [Meruvia and Strothotte 04]; Vanderhaeghe et al. present a

hybrid 3D-image-based algorithm to create non-uniform PD distributions for 3D scenes and video sequences [Vanderhaeghe et al. 07]. Both approaches aim at maintaining spatio-temporal coherence at the particle level. Since AIS is not intended for producing animated sequences, we don't further consider these stippling techniques.

3. Adaptive Incremental Stippling

Dunbar and Humphreys, Jones, and Bridson have presented incremental algorithms to produce Poisson-disk distributions for constant density functions [Dunbar and Humphreys 06, Jones 06, Bridson 07]. In Dunbar's and Humphreys's "boundary sampling" algorithm, each new candidate is elected from a region at distance r of a previously existing point. It uses an ordered generation relationship defined by the vicinity available to each point $p \in P$. In the most simple case, the neighborhood is limited to a collection of circular arcs centered in the points, known as *available boundary*. A new candidate is chosen at a random position along the available boundary. Once a candidate has been elected, the available boundary should be recalculated with the intersection of the candidate circle and its immediate neighbors. After the valid ranges have been determined, new points may be added repeating the process with all available positions of the border until the borderline is empty. To add a new point requires only a subtraction of an angular range from the candidate's boundary.

In a similar way to [Dunbar and Humphreys 06], AIS attempts to accommodate points complying with the minimal distance restriction (characteristic of a Poisson-disk distribution), but with the difference that this distance should be proportional to the local density of the emulated density function (an arbitrary grayscale image). As in the approach of Meruvia et al., we consider these points as the center of disks that cover a region that corresponds to the density function within the space they cover [Meruvia et al. 03]. Since all disks are meant to convey the same mass K (amount of darkness), the size of the disks will be larger in clear zones (low density), while in darker zones (higher density) the radius will be smaller. As a result, clear areas will have few points and darker areas will produce more points. For the purpose of this article we can consider the terms *mass*, *density*, and *amount of darkness* synonyms.

3.1. Algorithm Overview

The algorithm (see Figure 1) starts with a queue `Queue` that contains a disk at a random valid position. At the available boundary (which is initially the entire circumference), a new disk P with initial radius r_1 is generated. The

Algorithm 1. (Adaptive incremental stippling algorithm.)

```
Queue ⟵ Ø
D ⟵ newDisk(randomPosition, r1)
D ⟵ changeDiskRadiusBasedOnLocalDensity(D,Q,K)
enqueue(D,Queue)
insert(D, outputPointList)
drawDisk(D, disksBuffer)
while(Queue ≠ Ø)
{
    Q ⟵ dequeue(Queue)
    while (AvailableRange(Q) ≠ Ø)
    {
        ∝ ⟵ getRandomAngleFrom(AvailableRange(Q))
        P ⟵ newDisk([xq+(rq+r1)cos∝, yq+(rq+ r1)sin∝], r1)
        D ⟵ changeDiskRadiusBasedOnLocalDensity(P,Q,K)
        if (not(Overlap(D,disksBuffer)))
        {
            enqueue(D,Queue)
            insert(D, outputPointList)
            drawDisk(D, disksBuffer)
        }
        substractFrom(AvailableRange(Q),D)
    }
}
```

function `changeDiskRadiusBasedOnLocalDensity` modifies P's radius proportionally to the local density at the disk's position (more detail is presented below), which we rename D. If disk D does not overlap with other disks, it is inserted in the queue, placed on the `outputPointList` and drawn on the disks surface (`disksBuffer`). The available range of Q is updated then (with `substractFrom(AvalaibleRange(Q),D)`),with successful or failed insertion, and a new disk P is placed at random position on the available boundary of Q using the same process. When there is no available range at Q, a disk is removed from the queue (`Q` $\leftarrow$ `Dequeue(Queue)`) and the cycle repeats. The algorithm terminates when no more disks are left in the queue.

3.1.1. Adaptive Disk Placement

A new disk is placed at a random angle α on the available boundary, with initial radius r_1, obtained with the formula

$$\sqrt{M \cdot N/(\text{TotalDensity}/K)}.$$

In this expression, K is provided by the user and represents the amount of density conveyed by each stipple in the final image. In our implementation the density of a single pixel is a value from 0 to 255, with 255 representing total darkness. If K is equal to 255, each stipple in the resulting image will

correspond to a single fully dark pixel, and there will be as many stipples as the amount known as `TotalDensity`, defined as

$$\text{TotalDensity} = \sum_{x=1}^{M} \sum_{y=1}^{N} f(x, y), \tag{1}$$

where $f(xy)$ is the image intensity, M is the width, and N is the height of the image. Conversely, if K is equal to `TotalDensity` (the maximum possible value of K), there will only be a single stipple in the image, and the size of the stipple will be such that it will correspond to the value of the `TotalDensity`. The initial position of P is calculated as

$$(x_p, y_p) = (x_q + (r_q + r_1)\cos\alpha, y_q + (r_q + r_1)\sin\alpha), \tag{2}$$

with r_1 the initial radius, to be used by function `changeDiskRadiusBasedOnLocalDensity` to proportionally recalculate radius r_p to the approximated density occupied by the disk centered at (x_p, y_p) over its local space density (see Figure 3). A small radius is produced for darker regions and a larger radius is produced for clearer zones. The local space density can be approximated by sampling a Gaussian-filter window with height $2r_p$ and width $2r_p$, or using a circle within this window. Here, we use a square window centered at (x_p, y_p) with a circle in it, to assess the total density covered by the disk. This process is faster than assessing the Gaussian filter and still attains good results. This procedure is performed iteratively within `changeDiskRadiusBasedOnLocalDensity` until the local density matches the target value K, updating the position of the disk (x_p, y_p) and radius r_p according to the local density. `LocalDensity` is the sum of intensities within the area covered by the disk, where each pixel has an intensity value that ranges from 0 to 255. In our implementation we limit the number of iterations to

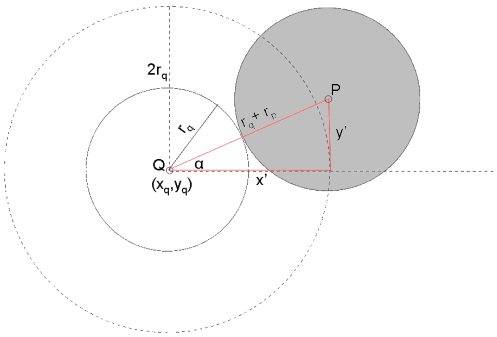


Figure 3. Once the radius r_p, is set, the position of P will be $(x_q+(r_q+r_p)\cos\alpha, y_q+(r_q+r_p)\sin\alpha)$.

Algorithm 2. (Algorithm for the function `changeDiskRadiusBasedOn LocalDensity (P,Q,K).`)

```
diskDensity ← localDensity(P, inputImage)
ratioR ← √K/diskDensity
epsilon← 0.001
i ← 0
while (ratioR > 1± epsilon  and i<6)
{
    r_n ←ratioR*r_p
    P ← Disk([x_q+(r_q+r_n)cos∝, y_q+(r_q+ r_n)sin∝], r_n)
    diskDensity ← localDensity(P, inputImage)
    ratioR ← √K/diskDensity
    i ← i +1
}
```

six, although two cycles are usually enough to find a suitable candidate (see Figure 2). Finally, the position of P is recalculated to

$$(x_p, y_p) = (x_q + (r_q + r_p)\cos\alpha, y_q + (r_q + r_p)\sin\alpha)$$

and called D.

When a new disk D is inserted, the available boundary of Q is affected according to proportion r_q/r_d, as shown in Figure 4. When there is no more available border in Q, the last disk D is considered the new Q, around which new disks will be inserted, repeating the process until all inserted disks have been considered and all available boundaries are exhausted.

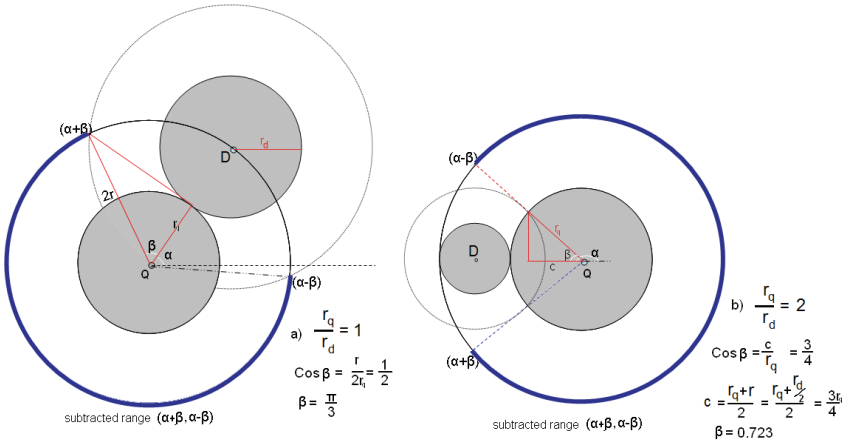
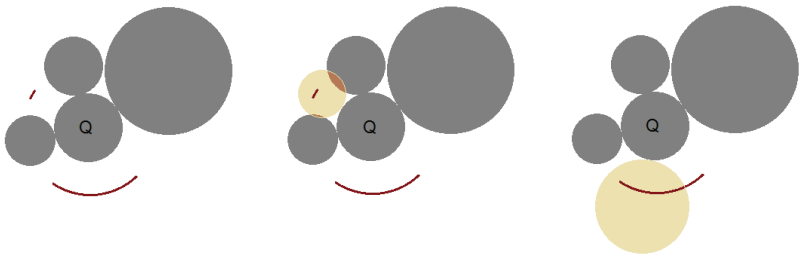


Figure 4. Geometry to calculate the available border based on $\frac{r_q}{r_d}$. The cases of $\frac{r_q}{r_d} = 1$ and 2 are shown, as well as examples of the precalculated values. The substracted range is shown as a bolded arc.



(a) (b) (c)

Figure 5. (a) The available boundary of Q and previously inserted disks with a darker background. (b) Insert attempt with overlap. (c) Successful disk placement. The disk with the lighter background represents the insertion attempt.

As a new point Q is selected, the available boundary is approximated with a *stencil buffer*, allowing the insertion of new disks only when they do not overlap with previous disks. (The stencil buffer is a working buffer on video memory to draw the inserted disks.) When a new disk is considered for insertion, the required memory space should be checked in order to assure it has not been previously assigned. We use an SDL surface, which initially is filled with the background color (black) and draw the disks on it. Before we try to insert a new disk D, we check that all the pixels in the region covered by the disk have the background color. If this happens, the function `Overlap(D,disksBuffer)` returns false and the disk D is drawn, otherwise the function `Overlap(D,disksBuffer)` returns true. `Simple DirectMedia Layer` (SDL) is a cross-platform multimedia library designed to provide low-level access to audio, keyboard, mouse, joystick, 3D hardware via OpenGL, and 2D video framebuffer. Figure 5 presents two insertion attempts based on the available range of Q.

Another alternative is the use of a mesh to determine the neighbors of Q before starting insertion of the surrounding points, as proposed by [Dunbar and Humphreys 06] in their basic incremental algorithm, but the variation in the radii of the disks complicates calculations. The final stippled image is obtained with a stipple placed at every disk centroid allocated in $outputPointList$.

A variant of AIS is obtained by incorporating Mould's observation that the gradient is helpful in defining borders and shapes in the images [Mould 07]. In this case, the position of P will now be in the mass centroid given by $mass = \gamma D + \beta E$, where D is the density, E is the edge intensity, and γ and β are weights ($\in [0, 1]$) controlling both quantities. We call this variant *adaptive incremental stippling with edge information* (AIS-E).

AIS has an order $O(n)$, where n is the number of output primitives; for each point (or disk), a constant number of operations that execute in constant time

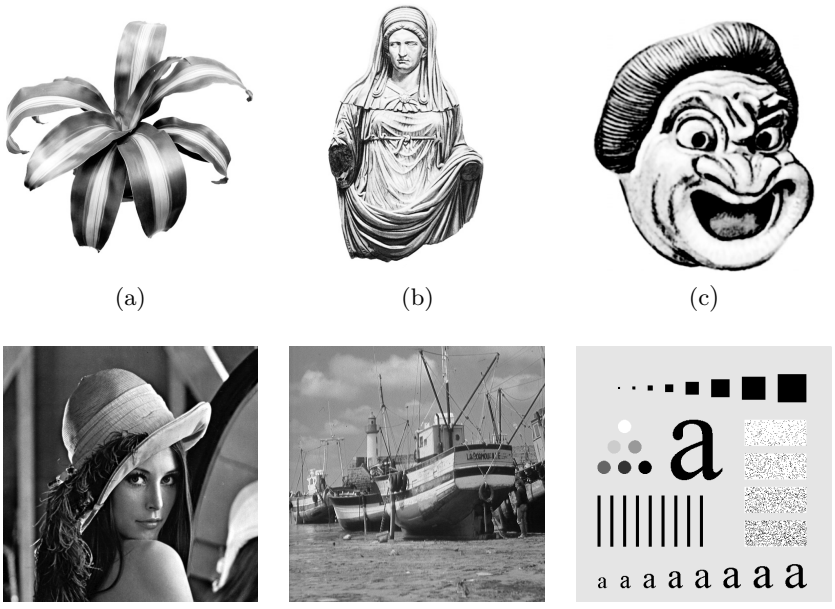
is performed: initial positioning, radius approximation according to density occupied by the disk, disk repositioning, and available boundary computation.

4. Examples and Discussion

We applied our method to the images presented in Figure 6. The results of AIS are observed in Figures 2(c) and 7(c), while examples of AIS-E are presented in Figures 2(d) and 7(d), with $\gamma = \beta = 1/2$.

In terms of radial metrics, we observe that all methods present low anisotropy (see Figure 2). Radial mean shows the typical shape of the PD distribution (as in Figure 1) with the best fit observed for CVD, and a relatively low fit observed for Wang Tiling (WAT), while AIS and AIS-E are closer to CVD. Observations at the spectrum for the WAT method [Kopf et al. 06], show that the ring surrounding low frequencies presents lower definition, reflected as a somewhat flatter radial mean. Note that Figures 2(d), 7(d), 8(d), 9(c), and 13 tend to better reproduce shape and border features, while keeping the blue noise features in the power spectrum. For a fair comparison, all test images in Figures 2 to 13 are shown with stipples of constant size.

Table 1 shows a comparison of the execution times in seconds for all methods used in Figures 2–10. CVD takes much more time than AIS or AIS-E, and

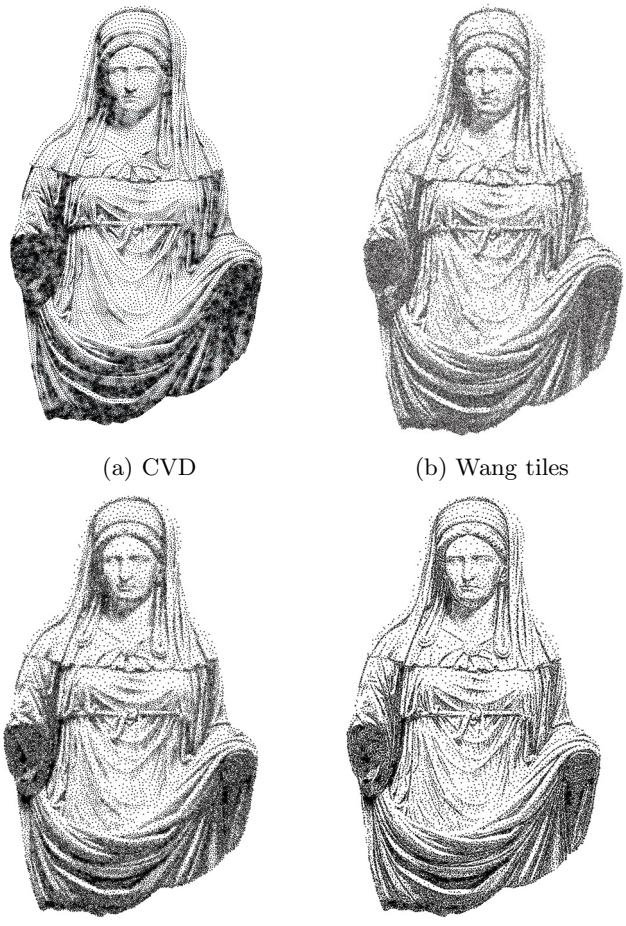


(a) (b) (c) (d) (e) (f)

Figure 6. Test images: (a) a cornplant, (b) Madonna, (c) a mask, (d) Lena, (e) a boat, and (f) type characters.

Method	CVD		WAT*		AIS		AIS-E	
Figure	stipples	time	stipples	time	stipples	time	stipples	time
Plant 2	15000	54s	15054	0.07s	15116	1.79s	15159	1.85s
Madonna 7	25000	90s	24326	0.11s	24633	3.01s	24744	3.07s
Mask 8	166000	**3h	166506	0.4s	163643	55.1s	166858	61.2 s
Lenna 9	35000	78s	35481	0.11s	-	-	36019	4.7s
Boat 10	43000	110s	43775	0.13s	-	-	43547	4.9s

Table 1. Execution times for the different algorithms. CVD: centroidal Voronoi diagram [Secord 02], WAT: Wang Tiling [Kopf et al. 06], AIS: adaptive incremental stippling, and AIS-E: AIS with edge information. *WAT preprocessing time is 12 minutes.



(a) CVD (b) Wang tiles

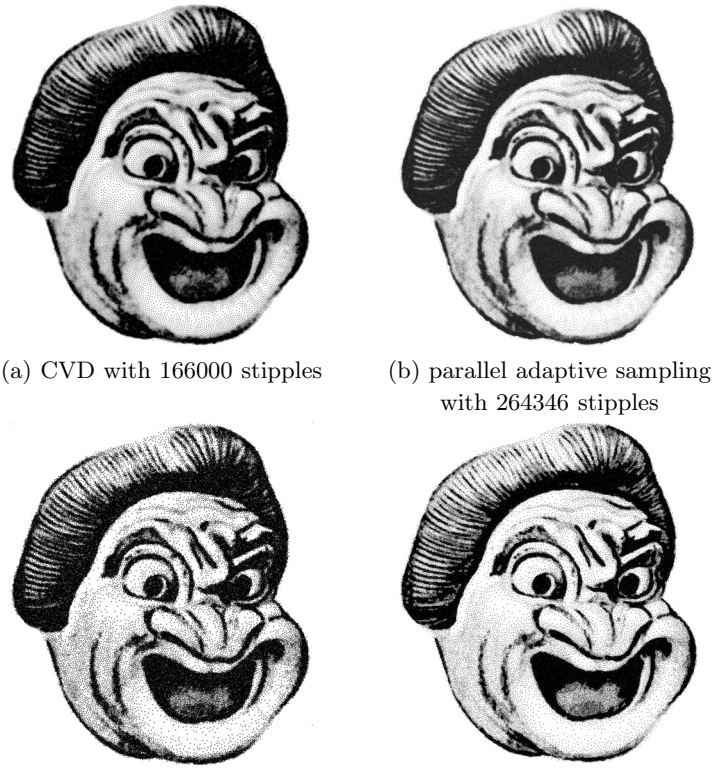
(c) AIS (d) AIS-E

Figure 7. Comparison of stippling results for images with high contrast.

none of these three methods require preprocessing. The execution time for WAT is the lowest overall (the time required for producing the WAT base tiles is approximately 12 minutes), but once computed, the base tiles can be reused for other images. These results were obtained using a PC with an Intel Quad processor at 2.66 GHz with 3GB RAM. NVIDIA QUADRO NSV 290 GPU was used for AIS, AIS-E and WAT; a Macintosh with an Intel Duo processor at 2.5 GHz with 4GB RAM and NVIDIA GeForce 8600M GT with 512MB RAM was used for producing the CVD images.

Execution time to produce stippling Figure 8(a) with CVD is uncertain because several magnifications are necessary to avoid the appearance of non-image patterns, requiring over three hours. Several circular patterns can be observed in Figures 7(a), 8(a), 9(a), and 10(a) because more magnification is needed.

As shown in Figures 8(b) and 8(c), stippling images generated with Wang Tiles [Kopf et al. 06] and the parallel implementation of [Wei 08], present fewer



(a) CVD with 166000 stipples (b) parallel adaptive sampling with 264346 stipples

(c) Wang tiles with 166506 stipples (d) AIS-E with 166553 stipples

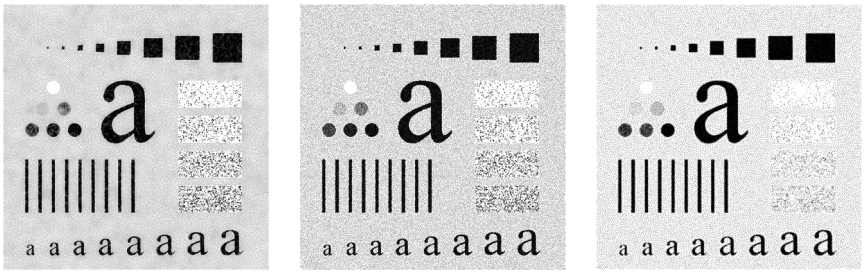
Figure 8. Comparison of images with high amount of stipples.

(a) CVD with 35000 stipples (78s) (b) Wang tiles with 35481 stipples (0.11s) (c)AIS-E with 36019 stipples (4.7s)

Figure 9. Comparison of images with a combination of smooth gradients and details of local contrast observed in the hair, eyes and parts of the hat.

(a) CVD (110s) (b) Wang tiles (0.13s) (c) AIS-E (4.9s)

Figure 10. Images with fine details. See fine lines composed of ropes and mast. illustrations formed with approximately 43000 stipples.



(a) CVD (95s) (b) Wang tiles (0.2s) (c) AIS-E (10s)

Figure 11. Image formed with approximately 76000 stipples.

obvious characteristics of distribution, even with a large amount of stipples. The picture generated by AIS-E presents a better defined structure.

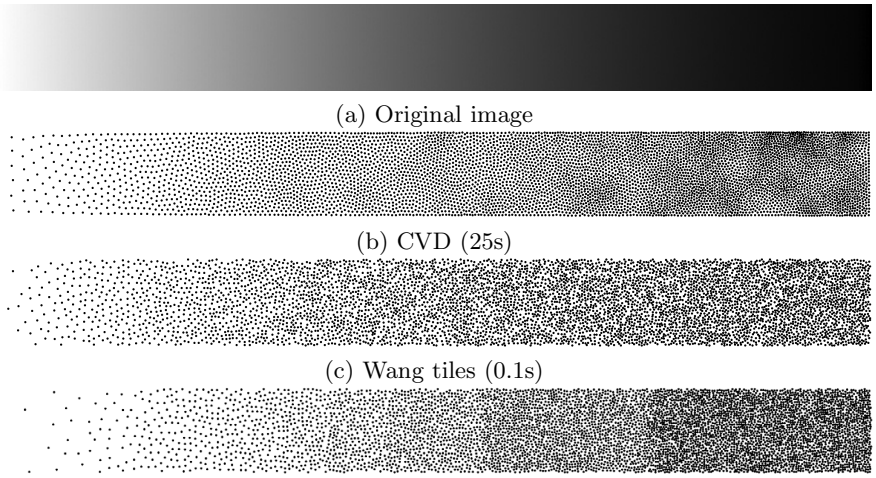
The spectral characteristics of the method used by Li-Yi Wei are very similar to WAT (as discussed in the same document). It's faster than WAT (and needs no preprocessing time), but like WAT, has the disadvantage of producing images with less aesthetic detail, particulary in high-contrast areas (see details in Figure 8).

Images in Figure 9 present areas with soft gradient changes (especially in face and background). For those regions, CVD presents better results, perhaps affected by the chain pattern.

Figures 9, 10, and 11 present a comparison of the methods when images with great richness of detail are considered. The WAT method [Kopf et al. 06] presents problems generating fine details such as ropes and masts, as shown in Figure 10, or hair and expression lines, as shown in Figure 9.

Figure 12 shows the stippling images of a linear ramp generated with the CVD, WAT, and AIS-E methods. We can observe subtle differences in each method. While AIS-E shows some steps over the gradient from left to right, it has the least noticeable clustering. CVD shows some diffuse clustering, which is more evident in the darker regions of the ramp. Finally, the WAT method shows an intermediate representation, with more clearly defined clusters of dark and white regions and no apparent discrete steps in the gradient.

The contribution of AIS is that it produces stippling illustrations with similar characteristics to those of CVD, but in shorter execution time, while providing a new adaptive PD-distribution method to the community.



(a) Original image

(b) CVD (25s)

(c) Wang tiles (0.1s)

(d) AIS-E (0.2s)

Figure 12. Linear ramp with 6000 stipples.

Figure 13. Drawing formed with 76884 primitive graphics of radius 1.0, generated in 13 seconds

AIS-E has major limitations compared with Wang Tiles and Wei's parallel implementation when the stipple number exceeds a few hundred thousand. However, using such a large amount of stipples is essentially the same as halftoning.

AIS and AIS-E use less resources than CVD [Secord 02], allowing for fast production of images with fewer points, such as the drawing in Figure 13. As can be seen from Figure 2, the images from AIS have better spectral characteristics and less undesirable noise than WAT [Kopf et al. 06], and are produced in less time than CVDs (see Table 1), creating an intermediate solution in terms of image quality and image creation time compared to existing methods.

Acknowledgments. The first author was funded by a scholarship from the CONACYT, México. The authors would like to thank Adrian Secord and Lourdes Peña-Castillo for their assistance in the production of the CVD images and Christoph W. Sensen at the Sun Center of Excellence for Visual Genomics, University of Calgary, as well as the Department of Computing Science at Thompson Rivers University in Kamloops, B.C., for supporting this endeavor.

References

[Bridson 07] Robert Bridson. "Fast Poisson Disk Sampling in Arbitrary Dimensions." *ACM SIGGRAPH Sketches*. (2007), article 22.

[Cohen et al. 03] Michael F. Cohen, Jonathan Shade, Stefan Hiller, and Oliver Deussen. "Wang Tiles for Image and Texture Generation." *ACM Transactions on Graphics*. 22:3 (2003), 287–294.

[Deussen et al. 00] Oliver Deussen, Stefan Hiller, Cornelius van Overveld, and Thomas Strothotte. "Floating Points: A Method for Computing Stipple Drawings." *Computer Graphics Forum*. 19:3 (2000), 40–51.

[Dunbar and Humphreys 06] Daniel Dunbar and Greg Humphreys. "A Spatial Data Structure for Fast Poisson-Disk Sample Generation." *ACM SIGGRAPH 2006 Transaction on Graphics*. 25:3 (2006), 503–508.

[Feng et al. 08] Louis Feng, Ingrid Hotz, Bernd Hamann, and Kenneth Joy. "Anisotropic Noise Samples." IEEE *Transactions on Visualization and Computer Graphics*, 14:2 (2008), 342–354

[Jones 06] Thouis R. Jones. "Efficient Generation of Poisson-Disk Sampling." *Journal of Graphics Tools*. 11:2 (2006), 27–36.

[Klassen 00] R. Victor Klassen. "Filtered Jitter." *Computer Graphics Forum*. 19:4 (2000), 223–230.

[Kopf et al. 06] Johannes Kopf, Daniel Cohen-Or, Oliver Deussen, and Dani Lischinski. "Recursive Wang Tiles for Real-Time Blue Noise." *ACM Transactions on Graphics Proceedings of SIGGRAPH*. 25:3 (2006), 509–518.

[Lagae and Dutré 05] Ares Lagae and Philip Dutré. "A Procedural Object Distribution Function." *ACM Transactions on Graphics*. 24:4 (2005), 1442–1461.

[Lagae and Dutré 08] Ares Lagae and Philip Dutré. "A Comparison of Methods for Generating Poisson Disk Distributions." *Computer Graphics Forum*. 27:1 (2008), 114–129.

[Maciejewski et al. 08] Ross Maciejewski, Tobias Isenberg, William Andrews, David Eber, Mario Costa Sousa, and Wei Chen. "Measuring Stipple Aesthetics in Hand-Drawn and Computer-Generated Image." *IEEE Computer Graphics and Aplications*. 28:2 (2008), 62–74.

[Meruvia et al. 03] Oscar Meruvia-Pastor, Bert Freudenberg, and Thomas Strothotte. "Real-Time Animated Stippling." *IEEE Computer Graphics and Aplications*. 23:4 (2003), 62–68.

[Meruvia and Strothotte 04] Oscar Meruvia-Pastor and Thomas Strothotte. "Graph-Based Point Relaxation for 3D Stippling." *Proceedings of the Fifth Mexican International Conference in Computer Science, ENC 2004*. Los Alamitos, CA: IEEE Press, 2004. 141–150.

[McCool and Fiume 92] Michael McCool and Eugene Fiume. "Hierarchical Poisson Disk Sampling Distributions." *Graphic Interface*. (1992), 94–105.

[Mould 07] David Mould. "Stipple Placement Using Distance in a Weighted Graph." *Computational Aesthetics*. 3 (2007).

[Ostromoukhov 07] Victor Ostromoukhov. "Sampling with Polyominoes." *ACM Transactions on Graphics*, 26:3 (2007), 1-6.

[Secord 02] Adrian Secord."Weighted Voronoi Stippling." *Proceedings of the Second International Symposium on Non-Photorealistic Animation and Rendering*. New York: ACM, 2002. 37–43.

[Ulichney 88] Robert A. Ulichney. "Dithering with Blue Noise." *Proc. of the IEEE*. 76:1 (1988), 56–79.

[Vanderhaeghe et al. 07] David Vanderhaeghe, Pascal Barla, Joëlle Thollot, and François Sillion. "Dynamic Point Distribution for Stroke-Based Rendering." *Proceedings of the Eurographics Symposium on Rendering*. (2007), 139–146.

[Wong 05] Ping Wah Wong. "Image Quantization, Halftoning, and Printing." *Handbook of Image & Video Processing*, second edition. Edited by Al Bovik. Burlington, MA: Elsevier, 2005. 925–937.

[Wei 08] Li-Yi Wei. "Parallel Poisson Disk Sampling." *ACM Trans. Graph.* 27:3 (2008), 20:1–20:9.

[Yellot 83] J. I. Yellot, Jr. "Spectral Consequences of Photoreceptor Sampling in the Rhesus Retina." *Science*. 221 (1983), 382–385.

Web Information:

http://jgt.akpeters.com/paper/Ascencio-Lopez10

Ignacio Ascencio-Lopez, CICESE–Ciencias de la Computación, Ensenada, B.C., Mexico (ascencio@uabc.edu.mx)

Oscar Meruvia-Pastor, Department of Computer Science, Memorial University of Newfoundland, St. John's, N.L., Canada (oscar@mun.ca)

Hugo Hidalgo-Silva, CICESE–Ciencias de la Computación, Ensenada, B.C., Mexico (hugo@cicese.mx)

Received May 13, 2009; accepted in revised form July 15, 2010.