Eurographics Symposium on Rendering 2011
Ravi Ramamoorthi and Erik Reinhard
(Guest Editors)

Volume 30 (2011), Number 4

Efficient Computation of Blue Noise Point Sets through Importance Sampling

Nima Khademi Kalantari and Pradeep Sen

Advanced Graphics Lab, University of New Mexico

Abstract

Dart-throwing can generate ideal Poisson-disk distributions with excellent blue noise properties, but is very computationally expensive if a maximal point set is desired. In this paper, we observe that the Poisson-disk sampling problem can be posed in terms of importance sampling by representing the available space to be sampled as a probability density function (pdf). This allows us to develop an efficient algorithm for the generation of maximal Poisson-disk distributions with quality similar to naïve dart-throwing but without rejection of samples. In our algorithm, we first position samples in one dimension based on its marginal cumulative distribution function (cdf). We then throw samples in the other dimension only in the regions which are available for sampling. After each 2D sample is placed, we update the cdf and data structures to keep track of the available regions. In addition to uniform sampling, our method is able to perform variable-density sampling with small modifications. Finally, we also propose a new min-conflict metric for variable-density sampling which results in better adaptation of samples to the underlying importance field.

Categories and Subject Descriptors (according to ACM CCS): Computer Graphics [I.3.3]: Picture/Image Generation—Antialiasing; Image Processing and Computer Vision [I.4.1]: Digitization and Image Capture—Sampling

1. Introduction

Among the different sampling patterns used in computer graphics, Poisson-disk distributions are of interest because they have excellent blue noise properties due to the low-energy annulus around the DC spike in their frequency spectrum. In Poisson-disk sampling, samples are placed uniformly in space based on a minimum Euclidean distance criteria between any two samples. This idea can be extended to variable-density sampling, in which the minimum distance is selected according to an underlying importance field.

The brute-force method to generate Poisson-disk samples is dart-throwing [DW85, Coo86], where samples are thrown with uniform probability and kept only if they have a distance larger than a predefined value from all samples already placed. This algorithm becomes very expensive for maximal point sets, however, since the probability of hitting the available region in the space becomes zero as the method approaches maximal point set. Maximal point sets are desirable because we want to pack Poisson-disk samples as efficiently as possible in the available space. By doing so, the samples are spaced out as far as possible, resulting in better frequency properties because of the larger low-energy annulus.

In this paper, we propose that Poisson-disk sampling can be considered a form of importance sampling, where sampling is based on a given probability density function (pdf). Although earlier work (e.g., [GM09]) achieved Poisson-disk distributions by sampling the domain with certain probability distributions using specialized data structures, by explicitly formulating the Poisson-sampling problem as a pdf we can apply the rules of probability to decompose it into simpler 1-dimensional pdf's. This theoretical foundation allows us to develop a novel method for generating maximal Poisson-disk distributions which are similar to those produced by dart-throwing but without the rejection of samples. Furthermore, by posing it as an importance-sampling problem, the extension to variable-density sampling is fairly straightforward. Specifically, our work makes the following contributions: (1) poses the problem of Poisson-disk sampling within the context of importance sampling to avoid rejection, (2) presents an algorithm that discretizes the space for calculating the required cdf's to approximate dart-throwing, (3) extends the uniform sampling algorithm to variable-density sampling, which is straightforward because of the use of importance sampling, and (4) introduces a min-conflict metric which improves the quality of variable-density sampling.

2. Previous Work

Poisson-disk sampling was introduced to computer graphics by Dippé and Wold [DW85] and Cook [Coo86] through dart-throwing methods, which are inefficient for maximal sets as described earlier. Several algorithms have been proposed to improve the speed of dart-throwing while maintaining the quality of the produced samples. For example, Jones [Jon06] uses Voronoi tessellation to guide new samples to the free space. This method still rejects some samples, but less than naïve dart-throwing. Dunbar and Humphreys [DH06]

present a novel data structure called a scalloped sector which guarantees insertion of the sample into an available region, thereby avoiding rejection of samples. White et al. [WCE07] and Gamito and Maddock [GM09] use a quadtree to keep track of available regions. White's method also uses a uniform grid to store information about neighboring samples, but this consumes a significant amount of memory, making the algorithm impractical for a large number of samples. Gamito and Maddock use a subdivision refinement strategy instead, which also allows them to sample in n-D space. Although rejection is not completely avoided in either of these methods, its probability is significantly reduced. However, none of these approaches have been demonstrated for variable-density sampling.

Another approach for producing blue noise samples is Lloyd's method [Llo82, MF92]. Secord [Sec02] uses a weighted Voronoi tessellation version of Lloyd's method for density adaption for non-photorealistic rendering applications. Balzer et al. [BSD09] improves on several shortcomings of Lloyd's method by utilizing the concept of capacity. Although the quality of their results is good, the computational time and memory usage are very high which makes it impractical for a large number of samples.

Finally, there are several methods for fast generation of Poisson-disk samples. Most of them use special tiles to fill the space [HDK01, CSHD03, LD05, KCODL06, ODJ04]. These methods have blue noise properties, but do not produce the same result as dart-throwing. Wei [Wei08] proposes a parallel Poisson-disk sampling implementation which can be run on a GPU. This method is very fast, but violates the uniform sampling condition [GM09] and is not equivalent to dart-throwing. Since these methods are all approximate and do not mimic the dart-throwing process, we will not focus on them in this paper. Note that although our method is technically also an approximate method due to our discretization of the sampling space, it is equivalent to dart-throwing for a reasonable grid resolution as shown in Section 5.

3. Background Theory

Our basic observation is that we can model the process of dart-throwing as an importance sampling problem with a 2D probability density function (pdf) $f_{X,Y}(x,y)$. For example, suppose we are sampling the 2D space $D = [0,1]^2$. When the space is empty we have a constant pdf equal to one for the entire space: $f_{X,Y}(x,y) = 1$. This means that a sample can be placed anywhere in the space with equal probability. After inserting the first sample, a rejection disk with radius r (the minimum distance between samples) centered on the sample will have zero value in the pdf. The remaining space will now have a pdf equal to $1/(1-\pi r^2)$. For subsequent samples the analysis is the same, except that some disks may overlap. In that case, only the area of the disks which falls into the available region affects the pdf. As the number of samples increases, the available space shrinks, making it harder for naïve dart-throwing to place a sample.

We propose a way to produce maximal point sets while avoiding sample rejection by leveraging this pdf formulation. We use the property of joint distributions to break down the 2D pdf $f_{X,Y}(x,y)$ into a 1D conditional pdf and a 1D pdf: $f_{X,Y}(x,y) = f_{Y|X}(y|x)\, f_X(x)$, where $f_X(x)$ is the marginal density function of x found by integrating $f_{X,Y}(x,y)$ over y:

$$f_X(x) = \int_0^1 f_{X,Y}(x,y)\, dy. \tag{1}$$

We can then compute the coordinates (x_i, y_i) of a new sample i in two steps: 1) random generation of x_i based on $f_X(x)$ and then 2) selecting y_i randomly according to $f_{Y|X}(y|x_i)$. The first part of the problem involves selecting a sample from a probability density distribution that is typically not uniform, a problem encountered in importance sampling. As in importance sampling, we first generate a value c uniformly in the range $[0,1]$ using a random number generator and then compute x_i by inverting the corresponding cumulative distribution function (cdf) [Shi92]:

$$x_i = F_X^{-1}(c). \tag{2}$$

Once x_i has been selected, we need to select y_i. Since $f_{Y|X}(y|x_i)$ is zero in the unavailable regions and constant in the available regions, we do not need to compute the conditional cdf to generate y_i. Rather, y_i can be easily generated by randomly sampling the parts where $f_{Y|X}(y|x_i)$ is non-zero. Note that this formulation never rejects any samples. If a region in the x dimension is entirely filled up, the cdf for that region will have a constant value so the inversion process of Eq. 2 will not allow any samples to be placed in the unavailable region. In the case of y, sampling according to $f_{Y|X}(y|x)$ guarantees we do not sample the unavailable regions as well. Effectively, one can think of our algorithm as mimicking the rejection process of dart-throwing by setting the pdf to zero in the unavailable regions. We continue to place samples until the pdf becomes zero for the entire space. This way, we achieve a maximal point set without rejection. This simple formulation enables us to develop the novel algorithm described in the next section.

4. Algorithm

4.1. Uniform Sampling

We now describe a uniform-sampling algorithm based on the theoretical background presented in the last section. The two key steps are the calculation of $f_X(x)$ and $f_{Y|X}(y|x)$. We might first consider computing $f_X(x)$ in continuous space. However, since the rejection disks for different samples are likely to overlap, computing $f_X(x)$ using Eq. 1 in analytical form is difficult. Also $f_{Y|X}(y|x_i)$ should be calculated for each x_i during runtime which is not efficient. Therefore, we discretize the 2D pdf $f_{X,Y}(x,y)$ in one dimension across x axis (not the y) so that we can quickly compute the required pdf's for our algorithm.

To do this, we discretize $f_{X,Y}(x,y)$ in the space $D = [0,1]^2$

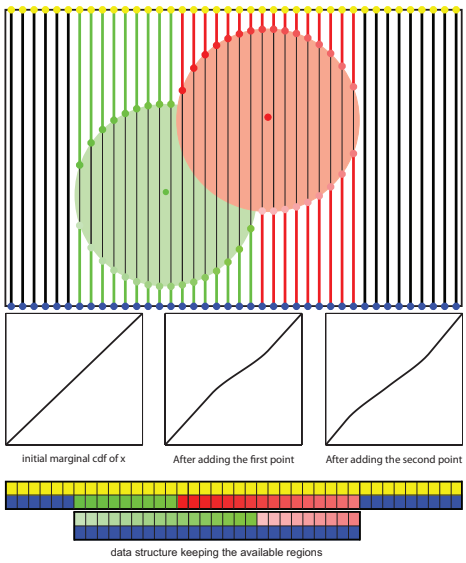


Figure 1: ***Visualization of the sampling process.*** *The first sample is shown with a red dot with the rejection disk around it in light red. The black lines inside the red disk represent regions where the pdf has been zeroed out and are subtracted from the marginal cdf. The red lines indicate available regions which are added to our availability data structure representing $f_{Y|X}(y|x)$, shown at the bottom. We have color coded the start and end points of each span to show the entry in the data structure. For the next sample, shown with a green dot, we update the cdf by using the black lines that are inside the green circle but outside the red one. The available span data structure now contains two spans in the middle positions. The cdf plot has been exaggerated for illustration purposes so the effect of subtracting the regions inside the circles can be seen.*

with $\lceil m/r \rceil$ discrete vertical lines along x ($r < 1$), where m sets the number of lines in a rejection disk radius r (empirically, we found a value of $m = 30$ to be sufficient, as discussed in Section 5.1). This means that our cdf is stored as a 1D array of size $\lceil m/r \rceil$. After placing the first point, we take a disk with radius r around it and rasterize the discrete lines crossing the disk. We subtract the length of each line span which falls inside the disk from our cdf $F_X(x)$ from that discrete position onward, effectively zeroing out the pdf in this region. For $f_{Y|X}(y|x)$, we only need to know the available regions for sampling, so we use a data structure to store the start and end points of spans that indicate availability in the continuous domain.

To place the next sample, we compute its x coordinate according to Eq. 2 by picking a random number c from [0,1] and inverting the cdf by scanning the cdf array until we find the two values that straddle c. The exact value of x_i is computed by interpolating the two cdf values based on c. This avoids complications in computing the inverse of the cdf to select x_i. The next step is to simply pick the y_i value by looking up the available regions at the position nearest to x_i in the data structure. After placing a new sample, we update $F_X(x)$ and the data structure using the same process described for the first sample except that we only subtract from the cdf the length of the line in the disk that lies in the free space. This avoids double-counting the reduction of the pdf when the rejection disk of the new sample overlaps existing rejection disks. Fig. 1 shows an example of this process.

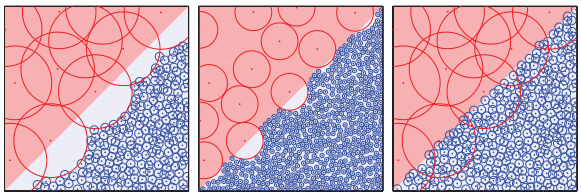
This procedure can be continued up to N_{max} times where N_{max} is the maximum number of circles with radius $r/2$ which can be packed inside the 2D space. The maximum packing is obtained when the samples are placed on a hexagonal grid, so $N_{max} = 2/(\sqrt{3}r^2)$. However, in practice the process is not continued N_{max} times but is terminated when there is no remaining space to put samples. This condition can easily be checked by looking at our data structure, which shows the available regions. When it is empty, we are finished sampling and a maximal point set has been obtained. Note that we use the term *maximal* to refer to the maximum packing of samples based on the discretization of our space, so if the resolution of discretization is low, the packing density is smaller than we would theoretically obtain.

For our algorithm, we need a 1D array of length $\lceil m/r \rceil$ for $F_X(x)$ and a 2D array for $f_{Y|X}(y|x)$, with length $\lceil m/r \rceil$ in x and variable length for the y dimension, with a theoretical peak on the order of $\sqrt{N}$. However, we find in practice that this length is typically smaller than $\sqrt{N}$ due to the randomness of the samples. Since $\sqrt{N} \approx 1/r$, the cdf size is on the order of $m\sqrt{N}$ and cost of updating it for each sample is $O(m\sqrt{N})$. The complexity of updating the data structure is also $O(m\sqrt{N})$ since we update a data structure of size $O(\sqrt{N})$ for $2m$ vertical lines. Since we repeat this process for N samples, the complexity of our method is $O(mN^{3/2})$.

At this point, we should mention that the proposed uniform-sampling algorithm can be simplified if all that is desired is a Poisson-disk distribution of the maximal set of samples. Instead of picking the x coordinate of the samples based on the cdf using Eq. 2 (which samples the space uniformly with every sample), we can simply pick the x coordinate uniformly from the available spans. This means that we are no longer uniformly sampling the space during the sampling process, because we give spans with less available space equal chance as those that have more. However, if we let the algorithm run until the sample set is maximal, the entire space would be filled up and the final sample distribution would have all the properties of Poisson-disk samples. We discuss the results of this simplification in more detail in Section 5.

4.2. Variable-density Sampling

Because our algorithm is based on importance sampling, the extension to variable-density sampling is straightforward. In this case, the radius at each position (x,y) should be selected according to an importance field $\mathcal{I}(x,y)$. We calculate the radius map as proposed Wei [Wei08]: $r(x,y) = g\mathcal{I}(x,y)^{-1/l}$, where g is a global parameter that controls the magnitude of



(a) Max-conflict **(b)** Mean-conflict **(c)** Min-conflict

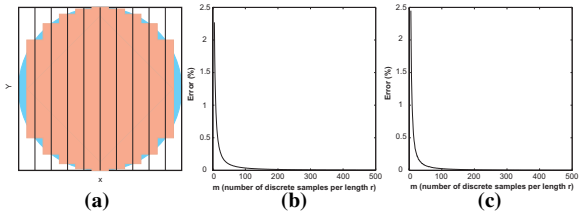
Figure 2: ***Illustration of max, mean, and min-conflict metrics for variable-density sampling.*** ***(a)*** *Traditional max-conflict metric. The circles indicate the rejection disks where other samples cannot be placed. The red region is sparse but its samples block those in the dense region shown in blue.* ***(b)*** *Traditional mean-conflict metric. In this illustration, the circles are not rejection disks, but are rigid bodies that cannot overlap. Because of the mean metric, the radius of these circles is half of the original. The samples in the red region block those in the blue region.* ***(c)*** *Our proposed min-conflict metric. The sparse samples do not block samples in the dense region, producing a better result.*

the radii and l defines the contrast of the final sampled result. We would like the density of samples to match the importance field, so if the importance field is 10 times larger in one region than another, it should have 10 times more samples. Therefore we set $l = 2$, making the number of samples proportional to $1/r^2$. Like the uniform sampling case where we discretized the x coordinate with $\lceil m/r \rceil$ values, in the variable-density sampling algorithm we discretize $f_{X,Y}(x,y)$ along the x dimension with $\lceil m/r_{min} \rceil$ values, where r_{min} is the minimum value in the radius map, to have sufficient resolution for the smallest circles in the high density regions. To calculate r_{min}, we use the maximum value in the importance field i_{max}: $r_{min} = g\, i_{max}^{-1/2}$.

Unlike the uniform sampling case, however, we discretize the space in *both* the x and y directions because of two key differences in calculating availability between variable-density and uniform sampling. First, for each new sample the radius of its conflict disk must be set based on the value in the radius map. Second, in order to block out the unavailable regions after placing a sample, we need to do "two-way" conflict checking, which means that we must not only take the radius of the new sample into account, but also the radii at the points inside the conflict disk that might be blocked out, since they might be of different size. Therefore, we cannot have a continuous domain in the y direction, because we would have to check an infinite number of points inside the conflict disk to determine availability since the radius map changes continuously. Because of this, we discretize $f_{X,Y}(x,y)$ along the y dimension as well, using the same resolution described above. The fact we have to perform a conflict check also raises the question as to what metric should be used, and we describe ours in the next section.

4.2.1. A New Min-Conflict metric

Certain conflict metrics can cause problems at the boundary of regions with different densities. Wei [Wei08] and Li et al. [LWSF10] proposed the *max-conflict* metric for these situations, where the maximum radius of the two positions is used to perform the conflict check. This metric has problems as shown in Figs. 2 and 6, where points near the boundary in the sparse region block samples near the boundary in the dense region, resulting in undesirable "cut out" artifacts. These two papers also introduced a *mean-conflict* metric in which the average of two points' radii is used to perform conflict check. Although this metric produces samples with slightly higher quality, it still has the same issue as the max-conflict metric but with a higher computational cost. To address these problems, we propose to use instead the *minimum* radius between two samples for the conflict check, which we call the *min-conflict* metric. With this simple change, we do not have visible artifacts at the density boundaries since the samples in the sparse region do not block the samples in the dense region.



(a) **(b)** **(c)**

Figure 3: ***Accuracy of disk approximation.*** ***(a)*** *Approximation of disk discretized in x when the sample is on the grid.* ***(b)*** *Percentage of absolute error between the exact and approximate circle areas when the disks are only discretized in x (as in the uniform sampling case) for different values of m, averaged over 10,000 samples.* ***(c)*** *The same plot for variable-density sampling discretized in both x and y directions. Both plots are similar, so the discretization in y does not affect the accuracy much.*

To apply this metric, we must examine all of the discrete positions within the conflict disk of a new sample to see if they are still available. This is different than in the uniform case, where we simply blocked out all the positions under a new sample's rejection disk. We can no longer do this for variable-density sampling, since as shown in Fig. 2c there may be valid sample positions within the dense region that fall inside the conflict disk of one of the sparse samples. To do these tests, we loop over all the discrete positions inside the rejection disk and perform a two-way min-conflict check between the radius of the new sample and the radius of the position in question, removing the positions from the cdf that fail this test.

Note that the size of the conflict disk is selected by the importance field at the new sample position, which is possible because the minimum of the two radii is guaranteed to be less than or equal to the radius of the new sample. Since we remove unavailable positions after positioning a new sample, these are taken into account in our cdf for placing future samples and so there is no rejection in our variable-density sampling algorithm either. In this case, the cost of updating data structure is now $O(m^2\sqrt{N})$ because of the additional

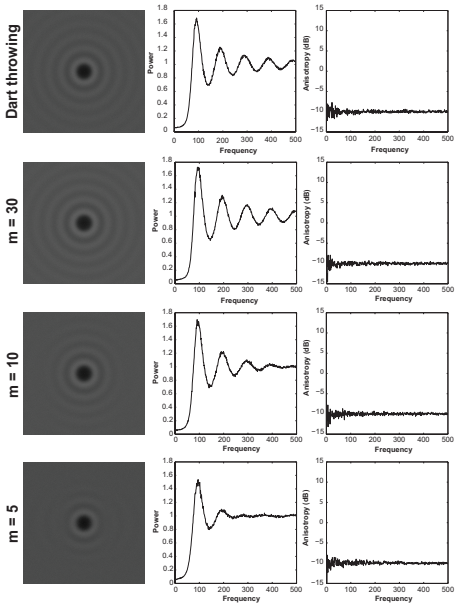


Figure 4: ***Quality of our Poisson-disk distribution.*** *Here we show how the periodogram, radial power, and radial anisotropy of the Poisson-disk distribution are affected by m. For reference, the result of dart-throwing is shown at the top. The quality is reasonable until $m = 10$, but degrades when m is too small. To do this experiment, we set $r = 0.01$ and averaged over 10 runs. Note that a radial anisotropy of -10dB indicates perfect radial symmetry [LD08].*

discretization in the y direction, making the overall complexity of variable-density algorithm $O(m^2N^{3/2})$.

5. Results

5.1. Uniform sampling

Our uniform sampling algorithm was implemented in C, compiled with the gcc compiler, and run under the Ubuntu Linux operating system. All timings are obtained on an Intel Core i7 2.8GHz machine with 4GB of memory.

We begin by studying the effect of discretization on the quality of the results. First, we examine the accuracy of the approximation of the rejection disk area with different m values in Fig. 3. As expected, the error is reduced as the resolution m increases, but the rate of improvement also decreases as m becomes large. To study the effect of m on the quality of the generated point set, we compare its periodogram and mean radial power and radial anisotropy [Uli87] to that of dart-throwing in Fig. 4. We find that these are similar to dart-throwing for values of m as low as 10. However, since our goal is to generate high-quality point sets, we use $m = 30$ for our experiments. For this m value, our method generates 6,900 samples which is denser than the 6,400 samples generated for dart-throwing because it is maximal point set.

To evaluate the performance of our algorithm, we tested our method against those of Gamito and Maddock [GM09], Dunbar and Humphreys [DH06] and Jones [Jon06] with their available implementation code. We generated Poisson-disk samples both with and without the toroidal boundary condition, which wraps the rejection disks for samples on the edge of the space around to the opposite edge. We also generated Poisson-disk samples using the simpler method described at the end of Section 4.1 which uniformly selects the samples in the x dimension instead of using the cdf. First, we consider the case without the toroidal boundary condition. Here, we compare against all methods listed except Dunbar and Humphreys, which cannot generate samples without this condition. The simulation was done for different radii and the results shown in Table 1 are averaged over 100 runs. We do not show the results of dart-throwing since it does not guarantee termination for maximal point sets. For example, it takes 23 seconds to generate 6,946 samples with $r = 0.01$, but we are unable to tell whether the point set is maximal.

Methods		Radius (r)			
		0.05	0.01	0.005	0.001
Ours	Time	0.0056	0.0913	0.391	26.7
	Samples	296.2	7,030.3	27,952	695,154
Uniform-x	Time	0.0066	0.0880	0.337	21.8
	Samples	298.6	7,058.3	28,019	695,599
Jones	Time	0.0423	0.813	3.207	100.4
	Samples	297.6	7,059.4	28,044	697,463
Gamito	Time	0.0092	0.101	0.493	16.7
	Samples	297.0	7,058.8	28,047	697,451

Table 1: *Comparison of timing and number of samples for different methods* ***without*** *the toroidal boundary condition. Results are averaged over 100 runs and timings are in seconds. "Uniform-x" is the method that uniformly samples in x instead of using the cdf.*

The results of the algorithms with the toroidal boundary condition are shown in Table 2. Here the Jones method is excluded since it cannot generate samples with toroidal boundary condition because its Voronoi decomposition cannot enforce periodicity. For the Gamito method, we used the code provided on their website which does not produce maximal point sets for this case. Also, Dunbar and Humphreys presented three algorithms in their paper: one that exactly mimics the dart-throwing process and two others that are approximate. The results presented in Table 2 are obtained using the accurate method to generate the samples and then we maximize the point sets using their approximate method.

These results show that our method is better than previous methods, except for Gamito's when the number of samples is large. However, none of these methods can do variable-density sampling, an advantage of our technique. Also note that the number of samples of our method is slightly less than the other methods (less than 0.3%) due to the discretization.

We also compare our method with the simpler approach which uniformly selects the x coordinate among the available vertical spans. As shown in Tables 1 and 2, this simpler method tends to pack more samples because it biases them toward spans that are more full. Therefore, if the algorithm can be run up to maximal sampling, this simpler method would produce reasonable results. However, in many appli-

Methods		Radius (r)			
		0.05	0.01	0.005	0.001
Ours	Time	0.0052	0.090	0.374	25.9
	Samples	277.9	6,938.1	27,762	694,243
Uniform-x	Time	0.0059	0.087	0.347	21.3
	Samples	280.7	6,965.7	27,825	694,646
Dunbar	Time	0.0952	1.747	6.726	162.6
	Samples	278.7	6,973	27,889	696,507
Gamito	Time	0.0066	0.120	0.502	16.2
	Samples	231.2	6,694.7	27,316	693,760

Table 2: *Comparison between different methods* ***with*** *the toroidal boundary condition. Results are averaged over 100 runs and timings are in seconds. "Uniform-x" is the method that uniformly samples in x instead of using the cdf.*

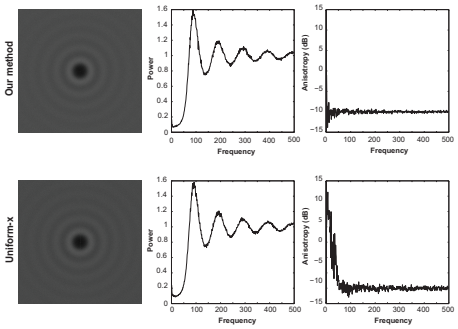


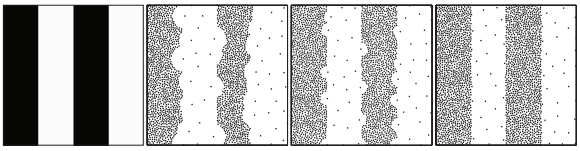
Figure 5: ***Comparison of uniform sampling in the*** *x* ***dimension with our algorithm.*** *Results are obtained by setting the radius equal to 0.01 and the desired number of samples to 6,000. Uniform sampling in the x dimension produces a result with a biased spectrum.*

cations (e.g., rendering) we cannot always utilize a maximal set of samples. Because this approach does not uniformly distribute the samples in the space during the sampling process, we can see a bias in the distribution for sets that are not maximal. To show this, Fig. 5 shows the frequency analysis for a set of 6,000 samples with toroidal boundary condition (where the maximal number of samples is approximately 6,900). The proposed algorithm has a reasonable periodogram, radial power, and radial anisotropy, while the uniform-x algorithm shows a significant bias as can be seen by the vertical line in the periodogram.

5.2. Variable-density Sampling

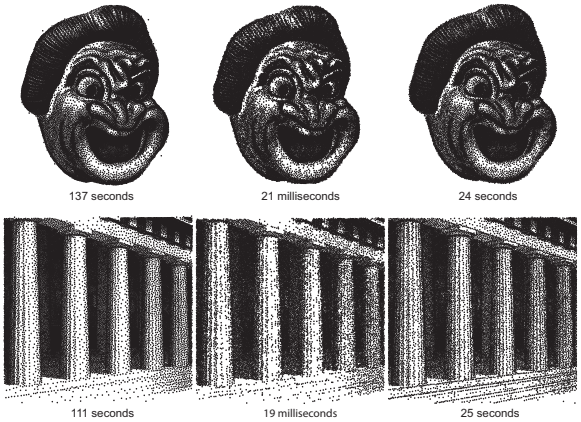
We begin by demonstrating the improvement of the proposed min-conflict metric over the traditional max and mean-conflict metrics. Fig. 6 shows a comparison of the three approaches given a specific importance field. To evaluate the performance of our variable-density algorithm, we compare it against the methods of Balzer et al. [BSD09] and Kopf et al. [KCODL06] for image stippling. We chose Balzer's method because of its high quality results, while Kopf's method was chosen because it is a fast method. The results are shown in Fig. 7 with 20,000 dots for all images.

When comparing to Kopf's method, we see that their method is fast but the quality of the produced images is reduced. The quality of our images is comparable to that of Balzer's method, but our algorithm is faster. In some parts of the image, such as in the steps of the bottom image, our method also performs better. The reason for this improvement is that our method produces maximal point sets which do not miss small details, while at the same time our proposed min-conflict check does not let the samples in sparse regions "cut out" the thin dense regions. Fig. 8 presents the breakdown of the influence of maximality and the min-conflict metric, and shows that most of the quality improvement is due to use of the min-conflict metric.



(a) Original image **(b)** Max-conflict **(c)** Mean-conflict **(d)** Min-conflict

Figure 6: ***Results using the max, mean and min conflict metrics.*** ***(a)*** *Density function.* ***(b,c)*** *With the max or mean-conflict metrics, samples in the sparse region block those in the dense region, producing artifacts.* ***(d)*** *Stippling using the min-conflict metric.*



(a) Balzer et al. **(b)** Kopf et al. **(c)** Ours

Figure 7: ***Comparison with the method of Balzer et al. and Kopf et al. for generating stippled images.*** *All stippled images use 20,000 dots of the same size. The grayscale images from Kopf et al. [KCODL06] are used as the density function.*

In terms of memory consumption, Balzer's method requires more memory than our approach. For example, to generate the bottom image of Fig. 7 the Balzer method needs 61MB of memory compared to 26MB for ours. The timing and memory gets bigger for larger number of samples. For example, if we generate the bottom image with 250,000 samples (instead of 20,000), the timings would be 311 mins for Balzer's method and 13 minutes for ours, while memory usage were 655MB for Balzer's method and 107MB for ours.

6. Limitations and future work

The method presented in this paper is only one way to implement the core idea of using importance sampling to gener-

(a) our method **(b)** max-conflict **(c)** not maximal

Figure 8: ***Breakdown of the influence of maximality and the min-conflict metric.*** *All stippled images use 20,000 dots of the same size.* ***(a)*** *Our proposed approach.* ***(b)*** *Substituting the max-conflict metric in our approach produces an image with some cut-outs, even though it is maximal.* ***(c)*** *Image obtained by adjusting the radii so that the maximal set has around 22,000 samples, while fixing the number of samples to 20,000. This shows the effect of the min-conflict metric when the set is not maximal.*

ate Poisson-disk samples. Although it has reasonable speed and memory requirements, the complexity of the proposed method is $O(mN^{3/2})$ and thus its speed is worse than Gamito and Maddock [GM09] for a large number of samples, although that algorithm is not adaptable to variable-density sampling. The use of more complex data structures to improve the computational complexity of the method should be investigated in the future. Furthermore, there is the possibility of updating the data structures in parallel using a GPU, which would make the algorithm faster. Finally, our method could be extended to anisotropic blue noise sampling by excluding rejection-ellipses instead of disks from our pdf's.

7. Conclusion

We have presented a novel algorithm based on importance sampling for generating Poisson-disk samples with similar quality to dart-throwing, and which can generate maximal point sets without rejecting samples. Our approach can not only generate high-quality uniform point sets competitive with all existing uniform algorithms, but unlike these we can also generate high-quality variable-density point sets that are comparable in quality to state-of-the-art methods such as Bazler et al. [BSD09] in less time. Moreover, we proposed a new min-conflict metric that produces better results along density edges. We note that this is an initial effort to develop an importance sampling algorithm for blue noise sampling. In the future, more advanced data structures might speed up the performance of the approach.

Acknowledgements

The authors thank the anonymous reviewers for their thoughtful comments and feedback which greatly improved the final version of the paper. The implementations for the algorithms of Jones [Jon06], Gamito et al. [GM09], Dunbar and Humphreys [DH06], Balzer et al. [BSD09], and Kopf et al. [KCODL06] were provided by the respective authors. This work was supported by the National Science Foundation under the NSF CAREER award #0845396.

References

[BSD09] BALZER M., SCHLÖMER T., DEUSSEN O.: Capacity-constrained point distributions: a variant of Lloyd's method. *ACM Trans. Graph. 28* (July 2009), 86:1–86:8. 2, 6, 7

[Coo86] COOK R. L.: Stochastic sampling in computer graphics. *ACM Trans. Graph. 5* (January 1986), 51–72. 1

[CSHD03] COHEN M. F., SHADE J., HILLER S., DEUSSEN O.: Wang tiles for image and texture generation. *ACM Trans. Graph. 22* (July 2003), 287–294. 2

[DH06] DUNBAR D., HUMPHREYS G.: A spatial data structure for fast Poisson-disk sample generation. *ACM Trans. Graph. 25* (July 2006), 503–508. 1, 5, 7

[DW85] DIPPÉ M. A. Z., WOLD E. H.: Antialiasing through stochastic sampling. *SIGGRAPH Comput. Graph. 19* (July 1985), 69–78. 1

[GM09] GAMITO M. N., MADDOCK S. C.: Accurate multidimensional Poisson-disk sampling. *ACM Trans. Graph. 29* (December 2009), 8:1–8:19. 1, 2, 5, 7

[HDK01] HILLER S., DEUSSEN O., KELLER A.: Tiled blue noise samples. In *Proceedings of the Vision Modeling and Visualization Conference 2001* (2001), VMV '01, pp. 265–272. 2

[Jon06] JONES T. R.: Efficient generation of Poisson-disk sampling patterns. *journal of graphics, gpu, and game tools 11*, 2 (2006), 27–36. 1, 5, 7

[KCODL06] KOPF J., COHEN-OR D., DEUSSEN O., LISCHINSKI D.: Recursive wang tiles for real-time blue noise. *ACM Trans. Graph. 25* (July 2006), 509–518. 2, 6, 7

[LD05] LAGAE A., DUTRÉ P.: A procedural object distribution function. *ACM Trans. Graph. 24* (October 2005), 1442–1461. 2

[LD08] LAGAE A., DUTRÉ P.: A Comparison of Methods for Generating Poisson Disk Distributions. *Computer Graphics Forum 27*, 1 (2008), 114–129. 5

[Llo82] LLOYD S.: Least squares quantization in PCM. *IEEE Trans. on Information Theory 28*, 2 (Mar. 1982), 129–137. 2

[LWSF10] LI H., WEI L.-Y., SANDER P. V., FU C.-W.: Anisotropic blue noise sampling. *ACM Trans. Graph. 29* (December 2010), 167:1–167:12. 4

[MF92] MCCOOL M., FIUME E.: Hierarchical Poisson disk sampling distributions. In *Proceedings of the conference on Graphics interface '92* (San Francisco, CA, USA, 1992), Morgan Kaufmann Publishers Inc., pp. 94–105. 2

[ODJ04] OSTROMOUKHOV V., DONOHUE C., JODOIN P.-M.: Fast hierarchical importance sampling with blue noise properties. *ACM Trans. Graph. 23* (August 2004), 488–495. 2

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

[Shi92] SHIRLEY P.: *Nonuniform random point sets via warping*. Academic Press Professional, Inc., San Diego, CA, USA, 1992, pp. 80–83. 2

[Uli87] ULICHNEY R.: *Digital Halftoning*. MIT Press, Cambridge, MA, 1987. 5

[WCE07] WHITE K. B., CLINE D., EGBERT P. K.: Poisson disk point sets by hierarchical dart throwing. In *Proceedings of the 2007 IEEE Symposium on Interactive Ray Tracing* (Washington, DC, USA, 2007), IEEE Computer Society, pp. 129–132. 2

[Wei08] WEI L.-Y.: Parallel Poisson disk sampling. *ACM Trans. Graph. 27* (August 2008), 20:1–20:9. 2, 3, 4