Computer-Aided Design 46 (2014) 37–44

Contents lists available at ScienceDirect

Computer-Aided Design

journal homepage: www.elsevier.com/locate/cad

# A parallel algorithm for improving the maximal property of Poisson disk sampling

Xiang Ying, Zhenhua Li, Ying He *

*School of Computer Engineering, Nanyang Technological University, Singapore*

## HIGHLIGHTS

- A simple algorithm for improving the maximal property of Poisson disk sampling.
- The algorithm is fully parallel and can be implemented on the GPU.
- It works for 2D and 3D, and can also be extended to surface in an intrinsic manner.

## ARTICLE INFO

*Keywords:*
Poisson disk sampling
Maximal sampling
Parallel algorithm
GPU
Exponential map

## ABSTRACT

This paper presents a simple yet effective algorithm to improve an arbitrary Poisson disk sampling to reach the maximal property, i.e., no more Poisson disk can be inserted. Taking a non-maximal Poisson disk sampling as input, our algorithm efficiently detects the regions allowing additional samples and then generates Poisson disks in these regions. The key idea is to convert the complicated plane or space searching problem into a simple searching on circles or spheres, which is one dimensional lower than the original sampling domain. Our algorithm is memory efficient and flexible, which generates maximal Poisson disk sampling in an arbitrary 2D polygon or 3D polyhedron. Moreover, our parallel algorithm can be extended from the Euclidean space to curved surfaces in an intrinsic manner. Thanks to its parallel structure, our method can be implemented easily on modern graphics hardware. We have observed significance performance improvement compared to the existing techniques.



## 1. Introduction

Poisson disk sampling is a process to distribute all samples that are uniformly and randomly located. It also requires a minimum distance apart from any two samples. Let $\mathcal{D}$ denote the sample domain. Poisson disk sampling is a set $X = \{x_i \in \mathcal{D};\ i = 1, 2, \ldots, N\}$ of $N$ samples having the properties

$$\forall x_i \in X, \quad \forall M \subseteq \mathcal{D} : P(x_i \in M) = \int_M dx, \tag{1}$$

$$\forall x_i, \quad x_j \in X : \|x_i - x_j\| \geq 2r, \tag{2}$$

where $P(\cdot)$ is a probability. A Poisson disk sampling is called *maximal* if there is no room to insert an additional sample, i.e.

$$\forall x \in \mathcal{D}, \quad \exists x_i \in X : \|x - x_i\| \leq 2r. \tag{3}$$

Due to its excellent spatial and spectral properties, Poisson disk sampling is widely used in computer graphics and visualization, such as anti-aliasing [1], global illumination [2], non-photorealistic rendering [3], remeshing [4], texture synthesis [5], vector field visualization [6], etc.

Dart throwing, proposed by Dippé and Wold [7], is the first accurate approach to generate Poisson disk patterns in $\mathbb{R}^n$. However, this brute-force approach is impractical and inefficient, since a large number of samples are involved in the attempt but only a small percentage of them are eventually inserted into the distribution. Since then, many approaches have been proposed to improve the performance of the dart throwing algorithm, e.g., jittered sampling [8], spatial data structures [9,10], procedure tiling [11–13], and hierarchical sampling [14,15]. Some algorithms [16,10,17] are able to generate maximal Poisson disk sampling. Although they are faster than the dart throwing algorithm, these algorithms are still sequential in the sense that the sample or dart is drawn one by one.

In contrast to many sequential algorithms, there are relatively few works for computing Poisson disk sampling in parallel. Wei [18] pioneered the phase group method, which subdivides the sample domain into grid cells and draws samples concurrently from multiple cells that are sufficiently far apart to avoid conflicts. Ying et al. [19,20] presented a technique to parallel the dart throwing by assigning each sample candidate a random and unique priority that is unbiased with regard to the distribution. Hence,

* Corresponding author. Tel.: +65 6514 1008.
*E-mail addresses:* yingxiang@ntu.edu.sg (X. Ying), lizh0021@e.ntu.edu.sg (Z. Li), yhe@ntu.edu.sg (Y. He).

0010-4485/$ – see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.cad.2013.08.016

multiple threads can process the candidates simultaneously and resolve conflicts by checking the given priority values. By taking advantage of the modern GPU, these parallel techniques significantly outperform the sequential algorithms. However, none of these parallel algorithms is maximal, i.e., when the algorithm terminates, there is still room for additional samples.

Ebeida et al. [21] presented a parallel algorithm to generate maximal Poisson disk sampling in $\mathbb{R}^d$. Their idea is to maintain a flat quadtree structure to keep track of the uncovered region. Their algorithm is theoretically sound and also practical for dimension up to 3 on the GPU. However, compared to the other parallel algorithms, Ebeida et al.'s approach requires large memory due to maintaining all the active cells, which diminishes its application to large scale models due to the limited memory on graphics card. Furthermore, it is not clear whether their flat quadtree data structure can be extended to curved surfaces. Recently, Yan and Wonka [22] studied the geometry of gaps in disk sets and then proposed efficient algorithms to detect and update these gaps when the Poisson disk distribution is updated (e.g., disks are inserted, deleted, moved or when their radii are changed). Yan and Wonka's algorithm can fill in the gaps in parallel. Moreover, their algorithm is theoretically sound and can generate maximal Poisson disk distributions with varying radii in the Euclidean space of arbitrary dimension or on a manifold.

On the other hand, there are many Poisson disk sampling algorithms on surfaces. However, very few of them could produce the *maximal* distribution. Due to the fundamental difference between the Euclidean space and the curved surface, efficiently computing a maximal Poisson disk sampling on arbitrary surfaces is still a challenging problem. Fu et al.'s method [4] produces the maximal distribution on surfaces at a very low performance (only 30–50 samples per second). Bowers et al.'s parallel algorithm [23] is able to generate 180k samples per second on the Nvidia GTX580, but their algorithm is extrinsic and approximate. Ying et al. [20] proposed an intrinsic, parallel and accurate algorithm which can produce 350k samples per second. However, neither [23] nor [20] could generate a maximal distribution.

This paper presents a simple yet effective technique to compute maximal Poisson disk sampling in $\mathbb{R}^2$ and $\mathbb{R}^3$. Taking a non-maximal Poisson disk sampling as input, our algorithm efficiently detects the regions allowing additional samples and then generates Poisson disks in these regions. The key idea is to convert the complicated plane or space searching problem into a simple searching on circles or spheres, which is one dimensional lower than the original sampling domain. Compared to the existing maximal Poisson disk sampling algorithm, our method is memory efficient, very fast and fully parallel, which can generate more than 8 million samples per second in $\mathbb{R}^2$ and 0.55 million samples per second in $\mathbb{R}^3$ on the Nvidia GTX580. Furthermore, our algorithm can also be extended to surfaces in an intrinsic manner.

The remaining of this paper is organized as follows: we first describe an interesting geodesic problem in Section 2. The solution of such a problem lays the foundation of our algorithm. Then we present our parallel algorithm in Section 3 followed by the experimental results and analysis in Section 4. We compare our method to the existing techniques in Section 5 and conclude the paper in Section 6.

## 2. A geometric problem

Before introducing our algorithm, we discuss an interesting geometric problem, which is closely related to maximal Poisson disk sampling.

*Problem statement.* Consider a set of circles $C_i = (c_i, r_i)$ in $\mathbb{R}^2$ (resp. spheres $S_i$ in $\mathbb{R}^3$) covering a finite domain $\mathcal{D} \subset \mathbb{R}^2$ (resp. $\mathbb{R}^3$), where $c_i$ and $r_i$ are the center and radius respectively. Any two circles (resp. spheres) do not intersect. Can we insert a circle (resp. sphere) of radius $r$ into $\mathcal{D}$ without intersecting the existing circles (resp. spheres)?

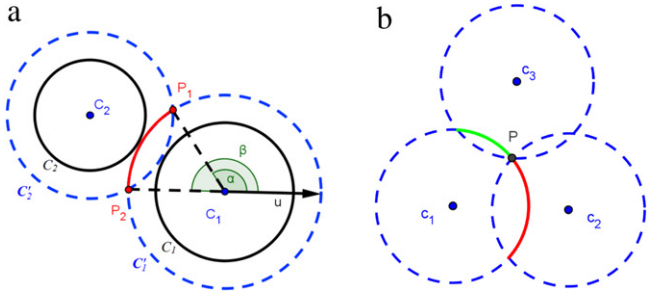


**Fig. 1.** (a) A "forbidden" arc $\widehat{p_1p_2}$ (in red) with two ending angles $[\alpha, \beta]$. (b) When multiple circles $C_i'$ intersect at one point $p$, such a point $p$ is a degenerate but "permitted" arc, which allows insertion of a circle with radius $r$. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

*Solution in 2D.* For each circle $C_i = (c_i, r_i)$, we draw a new circle $C_i' = (c_i, r_i + r)$ with radius $r_i + r$. Note that these new circles $C_i'$ may intersect due to the increased radius.

Now consider two circles $C_1'$ and $C_2'$ intersecting at $p_1$ and $p_2$ (see Fig. 1, where $C_i$ is in solid black and $C_i'$ in dashed blue). Clearly, we cannot insert an additional circle of radius $r$ in the region formed by $C_1' \bigcap C_2'$, since such a circle will intersect $C_1$ or $C_2$. Observe that the red arc $\widehat{p_1p_2}$ of circle $C_1'$ is completely inside circle $C_2'$. We call this arc "forbidden" since no circle of radius $r$ can be inserted into $C_1' \bigcap C_2'$. Other circles $C_i'$ can also generate forbidden arcs on $C_1'$ if they intersect $C_1'$. Then we define the "permitted" arc(s) on $C_1'$ as the complement of all forbidden arcs.

Thus, in order to find the room to insert an additional circle near $C_1$, it is equivalent to test whether the circle $C_1'$ has at least one permitted arc.

Take an arbitrary unit vector $\overrightarrow{u}$ as the reference axis for circle $C_1'$. Then we compute the angles $\alpha$ = angle between $\overrightarrow{u}$ and $\overrightarrow{c_1p_1}$ and $\beta$ = angle between $\overrightarrow{u}$ and $\overrightarrow{c_1p_2}$. The forbidden arc $\widehat{p_1p_2}$ is given by $(\alpha, \beta)$ if the reference axis $\overrightarrow{u}$ does not intersect the arc. Otherwise, we split it into two small arcs, i.e., $(-\epsilon, \beta)$ and $(\alpha, 2\pi + \epsilon)$, where $\epsilon = 1 \times 10^{-8}$ is a small value. Fig. 1(b) shows a special case where multiple circles intersect at one point $p$. In such a situation, a circle of radius $r$ can be inserted at $p$; thus, the point $p$ is a degenerate but permitted arc. The detailed algorithm to find all permitted arcs is documented in Algorithm 1.

*Solution in 3D.* The 3D problem can be solved by using the above 2D results. For each sphere $S_i = (c_i, r_i)$, we make a new sphere $S_i' = (c_i, r_i + r)$. Due to the increased radius, the spheres may intersect each other. Similar to the 2D case, if two spheres $S_i'$ and $S_j'$ intersect, we cannot insert a sphere with radius $r$ in between $S_i$ and $S_j$.

Observe that the intersection of two spheres is a circle, which bounds a disc on each sphere. Similar to the 2D case, we call such a disc forbidden, since it does not allow inserting an additional sphere. Also, the complement of all forbidden discs is called "permitted". Note that the boundary of the permitted region is a set of permitted arcs. Thus, in order to find the room to insert an additional sphere near $S_i$, it is equivalent to test whether the permitted region on $S_i'$ is non-empty.

This permitted region searching problem, on the other hand, can be formulated as finding permitted arcs. As shown in Fig. 2, let $C_1 = S_i' \bigcap S_1'$ and $c_2 = S_i' \bigcap S_2'$ be two intersecting circles on $S_i'$. And $C_1$ and $C_2$ also intersect at points $p_1, p_2$. Thus, we can use the permitted arcs technique to find the permitted region, i.e., a region which is not enclosed by any circles. Therefore, by searching all permitted regions on all spheres, we can find the available spaces for inserting new spheres.