# Parallel and Accurate Poisson Disk Sampling on Arbitrary Surfaces

Ying Xiang Shi-Qing Xin Qian Sun Ying He
School of Computer Engineering, Nanyang Technological University, Singapore

## 1 Introduction

Sampling plays an important role in a variety of graphics applications. Among existing sampling methods, Poisson disk sampling is popular thanks to its useful statistical property in distribution and the absence of aliasing artifacts. Although many promising algorithms have been proposed for multi-dimensional sampling in Euclidean space, very few research studies have been reported with regard to the problem of generating Poisson disks on surfaces due to the complicated nature of the surface. This still remains a challenge due to the following reasons: first, a surface is a two-dimensional manifold that has arbitrary topology and complicated geometry, and is embedded in $\mathbb{R}^3$ or even higher dimensional space. Second, the exact geodesic distance should be used to enforce the minimum distance constraint between any pair of samples. Third, the algorithm should be parallelized such that it can make full use of all available threads. Last but not least, the generated samples should be randomly and uniformly distributed on surfaces, and exhibit the blue noise pattern without bias. Wei [2008] pioneered a parallel Poisson disk sampling algorithm by subdividing the sample domain into grid cells and drawing samples concurrently from multiple cells that are sufficiently far apart to avoid conflicts. Bowers et al. [2010] extended Wei's algorithm to 3D surfaces. Their method is highly efficient, allowing sampling on large-scale models at interactive speed. However, the generated distribution is not fully random since the sequence of processing the phase groups follows a predefined order. Moreover, the approximate geodesic computation in their approach results in large errors in models with rich features and thus compromises the sampling quality.

This paper presents a parallel and unbiased algorithm for direct Poisson disk sampling on arbitrary surfaces. Our contributions are twofold. First, we propose a new technique for parallelizing the Poisson disk sampling. Rather than the conventional approaches that explicitly partition the spatial domain to generate the samples in parallel, our approach assigns each sample candidate a random and unique priority that is unbiased with regard to the distribution. Hence, multiple threads can process the candidates simultaneously and resolve conflicts by checking the given priority values. Second, our algorithm is unbiased as it uniformly and randomly distributes the Poisson disks on arbitrary surfaces. All the computations of our algorithm are purely based on the intrinsic metric and are independent of the embedding space. Therefore, it works for arbitrary surfaces in $\mathbb{R}^n$. To our knowledge, this is the first *unbiased* and *parallel* algorithm for distributing Poisson disks on arbitrary surfaces.

## 2 Sampling Algorithm

It is observed that samples have a uniform random position and a uniform random birth time in the sequential dart throwing algorithm. As time progresses, samples are accepted or rejected based on their distance from the samples that have already been born. We parallelize the classic dart throwing algorithm by assigning a *unique random* value to each sample, which represents the priority or order of the sample. The candidate samples can be processed by multiple threads simultaneously. Each thread checks the collision of the processed sample against its neighbors. When conflicts occur, the samples with the higher priority win and are accepted as Poisson disks. We prove that our priority based parallel sampling is equivalent to the dart throwing algorithm, thus, our algorithm generates unbiased Poison disk distribution.

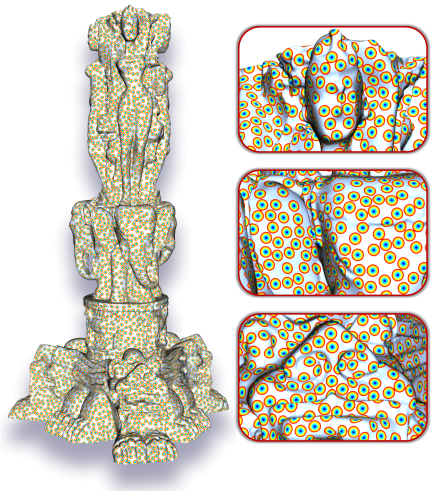

**Figure 1:** *Our parallel algorithm generates unbiased Poisson disk distribution on arbitrary surfaces. Our implementation on NVIDIA GTX 580 allows interactive sampling (350,000 samples/second) on the large-scale models.*

**Algorithm Overview.** Let $M$ be the input 2-manifold surface embedded in $\mathbb{R}^n$, and $r$ denote the radius of Poisson disks. Our algorithm contains the following four steps:
Step 1. Generate a dense point set $\mathcal{P}$ on $M$ and initialize each point's status to *IDLE*.
Step 2. Each thread randomly picks an *IDLE* point $p_i$ from $\mathcal{P}$, assigns a unique priority value and updates $p_i$'s status to *ACTIVE*. Then the thread computes a geodesic disk centered at $p_i$ with radius $2r$ and collects the *ACTIVE* and *IDLE* points inside the disk.
Step 3. If collision is found among *ACTIVE* points, the ones with lower priority are *REJECTED*.
Step 4. The *ACTIVE* points are accepted as Poisson disk samples and all the collided *IDLE* are *REJECTED*. Goto Step 2. until no more *IDLE* points can be found in set $\mathcal{P}$.

**Pre-defined Point Set.** We followed Osada et al. [2002]'s algorithm to generate a highly dense set of random points on $M$, which are uniformly distributed and insensitive to mesh tessellation. Each point has a time-dependent status, which could be *IDLE, ACTIVE, ACCEPTED*, or *REJECTED*.

```
enum PointStatus{
    IDLE,     //not processed
    ACTIVE,   //being processed by a thread
    ACCEPTED, //accepted as a Poisson disk
    REJECTED, //rejected
};
```

Each *ACTIVE* point has a unique random number to represent its priority. During the run time, each thread takes one active sample. Let $p_i$ be the active sample that is being processed by the $i$-th thread. To ensure that no two active samples have the same priority, the priority of $p_i$ is given by

$$priority(p_i) = \frac{rand() * T + i}{RAND\_MAX * T},$$

where $T$ is the total number of threads, and the function rand(),


SIGGRAPH Asia 2011, Hong Kong, China, December 12 – 15, 2011.
ISBN 978-1-4503-0807-6/11/0012

implemented by the parallel Multiply-With-Carry generator, returns a pseudo-random integer number in the range 0 to RAND_MAX-1.

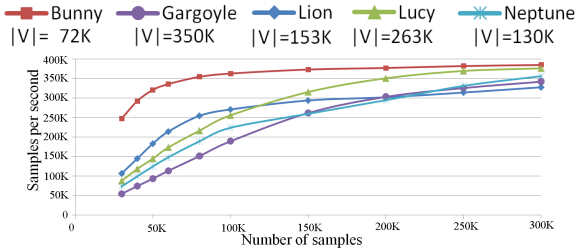


**Figure 2:** *Sampling speed* ***vs*** *number of samples.*

**Geodesic disk.** To enforce the minimal distance constraints, we implemented the Improved Chen-Han (ICH) algorithm [Xin and Wang 2009] on CUDA to compute the exact geodesic disks. The original ICH algorithm computes the "single-source all-destination" discrete geodesic on triangle mesh by maintaining the geodesic windows from near to far in a priority queue. It can be easily adapted to compute a geodesic disk with a user-specified radius. In our framework, each GPU thread initializes the CUDA-based ICH algorithm, which starts from the corresponding active sample, propagates the wavefronts and terminates when the geodesic distance exceeds $2r$. The memory cost of each ICH thread is linear to the number of mesh vertices in the geodesic disk of radius $2r$.

**Resolving conflicts.** For each *ACTIVE* sample, the corresponding thread computes a local geodesic disk of radius $2r$. If there is an *ACTIVE* point with higher priority inside the geodesic disk, the sample's status is changed to *REJECTED*.

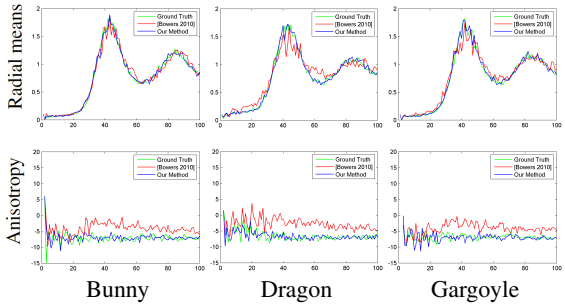


**Figure 3:** *Spectral analysis.*

## 3 Results

**Performance.** We implemented our algorithm and tested it on a PC with an Intel Xeon 2.66GHz CPU, 12GB memory and NVIDIA GTX 580. Our program is compiled using CUDA 4.0 RC2. Fig. 2 shows the performance of our algorithm on real-world models.

**Quality.** Our algorithm distributes Poisson disks randomly and uniformly on surfaces. We applied the spectrum analysis method in [Bowers et al. 2010] to check the quality of the generated distribution. As expected, the plots of both the radial mean and anisotropy of our method are highly consistent with the brute-force dart throwing in all examples (see Figure 3). The phase group algorithms [Wei 2008] and [Bowers et al. 2010] generate biased Poisson disk distributions, since the sequence of processing the phase groups follows a predefined order. As a result, the anisotropy plots of [Bowers et al. 2010] deviate from the ground truth. Further, our algorithm measures the exact distance between two samples, while Bowers et al. [2010] approximates the geodesic distance by a smooth curve interpolating the normals of two endpoints. Thus, Bowers et al.'s method may fail on models with rich features (see Fig. 4).

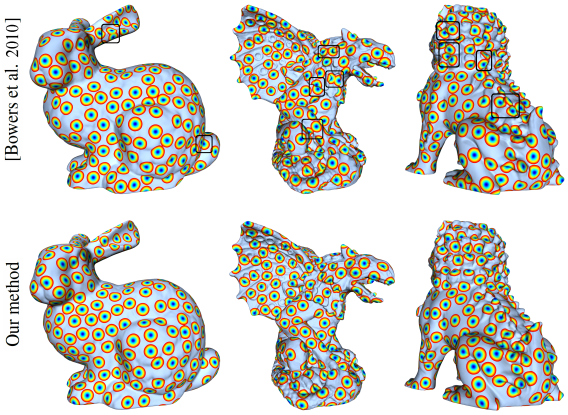


**Figure 4:** *To visualize the distribution, we compute the geodesic disks centered at each sample with radius $r$. Due to the approximate geodesic distance used in [Bowers et al. 2010], some samples violate the minimal distance constraints, as highlighted in the black boxes. Our method measures the exact geodesic distance, thus, can guarantee the generated distribution is free of conflicts.*

**Intrinsic sampling.** Our algorithm is intrinsic in the sense that it depends on the metric rather than the surface embedding, since all distances are measured intrinsically by the exact geodesic algorithm [Xin and Wang 2009]. Further, our method does not require the spatial partition of the embedding space, which is fundamentally different from the phase group algorithm [Wei 2008]. Therefore, our algorithm works for surfaces in the arbitrary dimension. Figure 5 shows examples of intrinsic sampling on surfaces in $\mathbb{R}^4$ and the pose invariant sampling of a lion in $\mathbb{R}^3$.

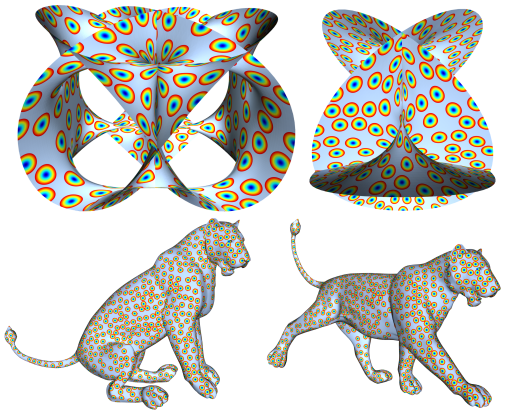

**Figure 5:** *Row 1: Poisson disk sampling on 4D surfaces. For rendering purpose, the surfaces are projected into $\mathbb{R}^3$; Row 2: the generated sampling is intrinsic and invariant to the pose.*

**Acknowledgements.** This work was supported by NRF2008IDM-IDM004-006.

## References

BOWERS, J., WANG, R., WEI, L.-Y., AND MALETZ, D. 2010. Parallel poisson disk sampling with spectrum analysis on surfaces. *ACM Trans. Graph. 29*, 166:1–166:10.

OSADA, R., FUNKHOUSER, T., CHAZELLE, B., AND DOBKIN, D. 2002. Shape distributions. *ACM Trans. Graph. 21*, 807–832.

WEI, L.-Y. 2008. Parallel poisson disk sampling. *TOG 27*, 20.

XIN, S.-Q., AND WANG, G.-J. 2009. Improving Chen and Han's algorithm on the discrete geodesic problem. *TOG 28*, 4, 1–8.