Computers & Graphics 46 (2015) 72–79

Contents lists available at ScienceDirect

Computers & Graphics

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

SMI 2014

Efficient maximal Poisson-disk sampling and remeshing on surfaces

Jianwei Guo [a], Dong-Ming Yan [b,a,*], Xiaohong Jia [c], Xiaopeng Zhang [a]

[a] NLPR-LIAMA, Institute of Automation, Chinese Academy of Sciences, Beijing, China
[b] Visual Computing Center, King Abdullah University of Science and Technology, Thuwal, Saudi Arabia
[c] KLMM, NCMIS, Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing, China

ARTICLE INFO

Article history:
Received 3 July 2014
Received in revised form
28 August 2014
Accepted 17 September 2014
Available online 2 October 2014

Keywords:
Maximal Poisson-disk sampling
Remeshing
Blue-noise

ABSTRACT

Poisson-disk sampling is one of the fundamental research problems in computer graphics that has many applications. In this paper, we study the problem of maximal Poisson-disk sampling on mesh surfaces. We present a simple approach that generalizes the 2D maximal sampling framework to surfaces. The key observation is to use a subdivided mesh as the sampling domain for conflict checking and void detection. Our approach improves the state-of-the-art approach in efficiency, quality and the memory consumption.

1. Introduction

Poisson-disk sampling is a fundamental research topic in computer graphics that has many applications such as rendering, digital half-toning and object distribution.

Given a sampling domain Ω and a sizing function $\rho(\mathbf{x})$ defined over Ω that indicates the sampling radius r at each point, an ideal Poisson-disk set $P=\{(\mathbf{x}_i, r_i)\}$ should satisfy three properties: (1) *minimal distance property* requires that the distance between any two disk centers should be larger than the sampling radius, i.e., $\forall \mathbf{x}_i, \mathbf{x}_j \in P,\ \|\mathbf{x}_i, \mathbf{x}_j\| \geq \min(r_i, r_j)$; (2) *unbiased sampling property* requires that each point in the domain has the probability that is proportional to the sizing at this point to receive a sampling point; and (3) *maximal sampling property* requires that the union of the disks covers the entire sampling domain, i.e., $\bigcup(\mathbf{x}_i, r_i) \supseteq \Omega$. If the sizing function is a constant, then the sampling is the classic uniform sampling. Otherwise, it becomes the adaptive sampling as discussed in [1]. If a sampled Poisson-disk set fulfills the above three requirements, then it is called a *Maximal Poisson-disk Sampling* (MPS). Except the aforementioned applications, it has been shown that the triangulation of a MPS point set also exhibits excellent geometric properties, such as angle bounds and edge length bounds, which are important to applications such as physical simulation [2,1].

The classic method for generating the Poisson-disk sampling is called dart-throwing [3]. However, it is known that dart-throwing cannot achieve the maximal sampling property. Different data structures have been proposed for accelerating the efficiency and improving the sampling maximality, e.g., active front of disks [4], Voronoi diagram [5], hierarchical quadtree [6,7], uniform grid [8], power diagram and regular triangulation [1].

More recently, the research of MPS has been generalized to mesh surfaces. Yan and Wonka [1] show that the remeshing of the uniform MPS can preserve the same geometric bounds on surfaces as that of the 2D domains [2]. They further generalize the MPS to adaptive sampling and remeshing on surfaces. In this work, a theoretical analysis of the gap existence is provided based on the restricted power diagram on mesh surfaces, and a uniform 3D grid is used to assist the conflict checking of the dart-throwing. Although high quality remeshing can be generated, the approach of [1] is both time and memory consuming. Furthermore, since the Euclidean distance for conflict checking in a global grid structure is used, the sampling has the problem in thin-sheet regions or nearby regions that have small Euclidean distance but large geodesic distance (An example of our algorithm is shown in Fig. 1).

In this paper, we propose a simple method for maximal Poisson-disk sampling on mesh surfaces. Our approach follows the definition of [1] for the adaptive MPS on surfaces. We generalize the fast uniform MPS in Euclidean space [9] to mesh surfaces. The key idea is to use a subdivided mesh for conflict checking, which is analogous to the grid structure used for MPS in Euclidean space. As a consequence, the proposed algorithm improves both the sampling quality and the efficiency upon the state-of-the-art approach. The main contributions of this paper include (1) an

* Corresponding author at: Visual Computing Center, King Abdullah University of Science and Technology, Thuwal, Saudi Arabia.
E-mail addresses: jianwei.guo@nlpr.ia.ac.cn (J. Guo), yandongming@gmail.com (D.-M. Yan), xhjia@amss.ac.cn (X. Jia), xiaopeng.zhang@ia.ac.cn (X. Zhang).

http://dx.doi.org/10.1016/j.cag.2014.09.015
0097-8493/

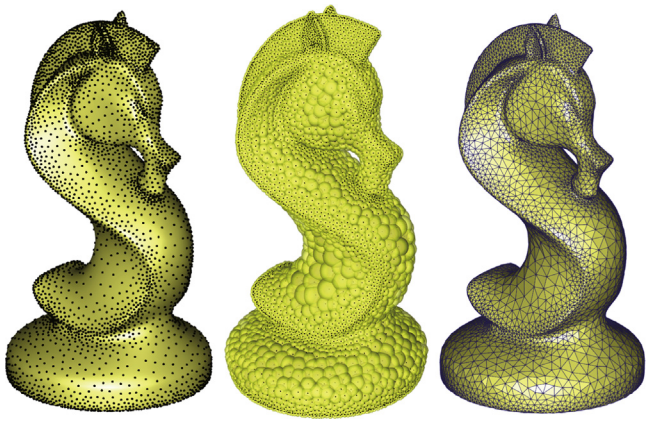


Fig. 1. An example of the maximal Poisson-disk sampling and remeshing on the Chess model. Left: the sampled points (7500 samples); middle: the spheres centered at the samples that cover the input surface; and right: the remeshing result. The angles are bounded between $[34^\circ, 112^\circ]$.

efficient framework for MPS on surfaces, which is a generalization of the fast 2D sampling [9], and (2) several improvements upon the state-of-the-art surface MPS algorithm [1], including smaller memory footprint, faster conflict checking, and remeshing with valid topology.

2. Related work

Poisson-disk sampling: Cook [3] first proposes an algorithm for 2D Poisson-disk sampling, called dart-throwing. Given a sampling domain and a sampling radius, the algorithm keeps generating disks (darts) in the sampling domain randomly. If the current generated disk conflicts with any previous sampled disk, then it is rejected, otherwise it is accepted. This process is repeated until a continuous number of rejection is observed. The algorithm complexity of the original dart-throwing algorithm is $O(n^2)$ [3]. Because of the existence of tiny uncovered regions that are difficult to receive a random point, the result of dart-throwing is not maximal. To achieve the sampling maximality, different data structures are designed to trace and sample the empty regions. Dunbar and Humphreys [4] use a data structure called scalloped sectors that records the active front of the sampled disk set for maximal sampling. The algorithm complexity becomes $O(n \log(n))$, but the sampling is biased. Jones [5] first introduces a unbiased MPS algorithm. It builds the Voronoi diagram of the already sampled point set, and compute the empty regions by subtracting the disk from each Voronoi cell. Then the empty regions are filled until the maximality is reached. The algorithm complexity is also $O(n \log(n))$. Wei [10] presents a parallel method for Poisson-disk sampling on GPUs. The speed is much faster than previous work, but the sampling is not maximal and is biased. Ebeida et al. [8] propose a two-step unbiased sampling strategy. They first perform the classic dart-throwing on a uniform grid, and then switch to computing and filling empty regions by subtracting the incident disks of each grid cell. Other approaches use hierarchical quadtrees to track the empty regions [6,7]. Ebeida et al. [9] further accelerate the sampling process by sampling a flat fragment array instead of using the hierarchical quadtree. In this paper, we show how to extend this fast sampling scheme to 3D surfaces. Mitchell et al. [11] study 2D Poisson-disk sampling with various radii, and they analyze the conflict condition under which a maximal sampling can be achieved.

Poisson-disk sampling on surfaces: Cline et al. [12] first extend the dart-throwing algorithm to mesh surfaces using a hierarchical quadtree data structure. Bowers et al. [13] propose a parallel surface sampling method with the GPU acceleration. Corsini et al. [14] first perform a pre-sampling stage, and then carefully choose samples from the pre-sampled point set. Ying et al. [15] propose a GPU-based approach that generates Poisson-disks on surfaces using the geodesic distance. The recent work of Ying et al. [16] proposes a parallel approach to improve the maximal property of the Poisson disk sampling. The discrete exponential map is used for the surface sampling. However, these approaches are not maximal.

Yan and Wonka [1] first introduce a theoretical framework for analyzing and extracting the empty regions on surfaces, using the restricted power diagram and the restricted regular triangular. Then this framework is used for maximal Poisson-disk sampling on surfaces. While high-quality sampling/remeshing results can be obtained by this approach, the performance is limited due to the costly computation. In this paper, we present an alternative simpler approach that is able to achieve the better quality, higher performance, and lower memory footprint. Later, this MPS framework is extended to handle isosurfaces [17].

Other approaches: There are quite a lot other approaches that aim at high quality sampling on surfaces. Fu and Zhou [18] extend the active front approach of [4] to mesh surfaces. They use geodesic distance to trace the disk borders on surfaces in sampling stage. Then, they perform the Lloyd relaxation [19] for high quality remeshing. Yan et al. [20] present an efficient algorithm for computing the restricted Voronoi diagram on mesh surfaces, and then they use an accelerated version of the Lloyd relaxation [21] to compute the so-called *Centroidal Voronoi tessellation* (CVT) for isotropic surface remeshing. Chen et al. [22] combine the CVT and the capacity constrained Voronoi tessellation for surface blue-noise sampling. Xu et al. [23] generalize the capacity constrained Voronoi tessellation to surfaces. More recently, Chen et al. [24] introduce *Bilateral Blue-noise Sampling* that is suitable for dense point set sub-sampling. Yan et al. [35] generalize the 2D *Farthest Point Optimization* [36] for surface sampling while maintain the blue-noise properties. However, all these methods rely on the iterative optimization/relaxation, which are not suitable for real-time applications.

3. Overview

In this paper, we present a simple algorithm for maximal Poisson-disk sampling on mesh surfaces. There are two key issues of generating MPS on surfaces, i.e., (1) conflict checking for dart-throwing, and (2) empty regions detection and filling for maximal sampling. Our approach aims at improving both the sampling quality and the efficiency upon the state-of-the-art method in these two aspects. First, we improve the sampling quality by using a local region-growing-based approach for conflict checking that avoids the misclassification caused by using a global grid. Second, we improve the efficiency of surface MPS by generalizing the simple 2D MPS algorithm to mesh surfaces that avoids the computation of the restricted power diagram on surfaces as in [1].

4. Methodology

The input of our algorithm is a 2-manifold triangular mesh surface $\mathcal{M}$, which consists of a set of triangles $\{t_j\}_{i=1}^{n}$. The minimal and the maximal sampling radius r_{min} and r_{max} are specified by the user. Typically we set $r_{max} = \lambda r_{min}, \lambda \geq 2$ (we choose $\lambda = 8$ for adaptive sampling in this paper). If $\lambda = 2$, then it becomes the uniform sampling. Supposing that a sizing function $\rho(\mathbf{x})$ is defined over the mesh surface that indicates the local sampling radius which is mapped to $[r_{min}, r_{max}]$. Either local feature size [25] or curvatures [13] can be used for defining the sampling radius.

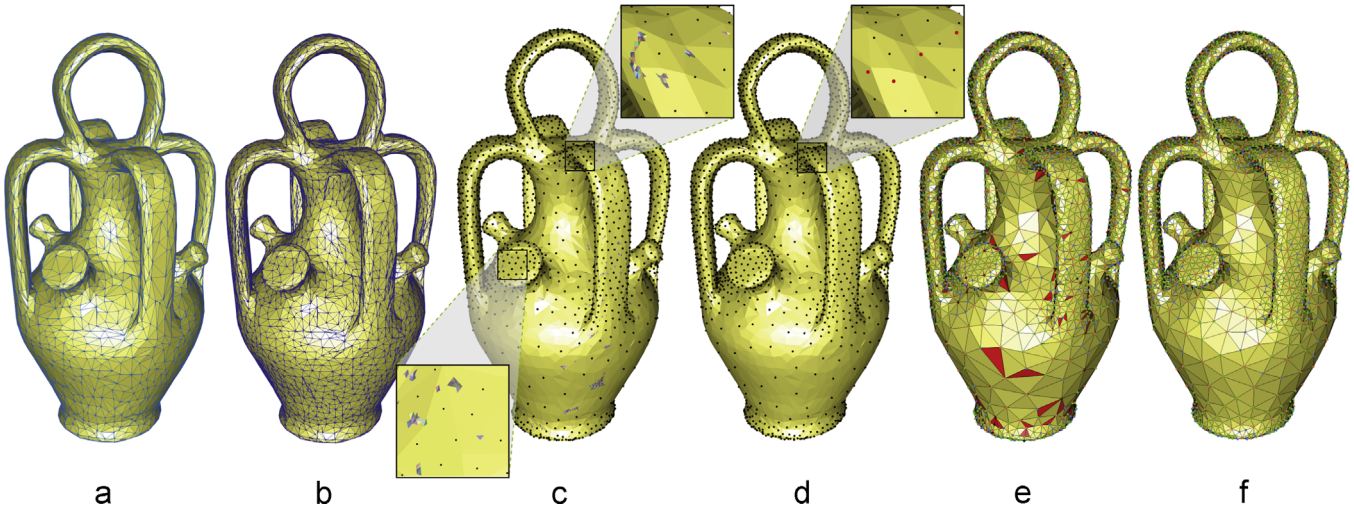


Fig. 2. The pipeline of our framework. (a) Input surface. (b) Subdivided surface that serves as the sampling domain. (c) Initial sampling. The zoomin view shows the triangles that are not fully covered by disks. (d) Maximal sampling. The red points in the zoomin view are sampled by filling the empty regions. (e) Extracted mesh from the MPS. The vertices are color-coded by the valence. Blue indicates the valence 5, green is valence 6 and orange is valence 7. The red triangles are those with minimal angle smaller than a user specified value, i.e., 32° in this example. (f) Optimized sampling and remeshing. The vertex valences are pure 5, 6, and 7, and the 'bad' triangles are eliminated. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

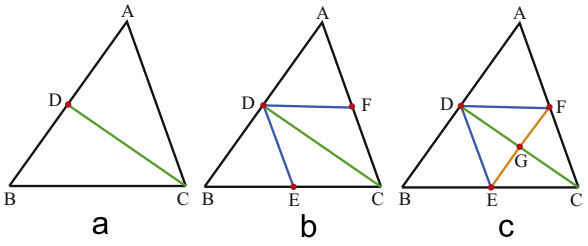


Fig. 3. The subdivision process for an input triangle.

In this paper, we use the local feature size (lfs) as the sizing function. For each point $\mathbf{p}_i$ on the surface, we define its radius $r_i = k\rho(\mathbf{p}_i)$, thus r_i is proportional to its local edge length (k is a constant).

We follow the recent work [1] for the definition of the adaptive MPS on surfaces, i.e., the sampling process should follow the three properties as discussed in Section 1. The five main steps of our framework are the following: (1) initialization, (2) initial sampling, (3) maximal sampling, (4) mesh extraction, and (5) mesh optimization. Note that the mesh optimization step is optional. Fig. 2 illustrates these steps of the proposed approach. In the following, we will explain the details of each step.

4.1. Initialization

The purpose of this step is to construct a suitable structure for Poisson-disk sampling on the surface that is similar to the flat quad array used in [9]. Toward this aim, we simply split the edges of the input mesh $\mathcal{M}$ whose length is larger than the minimal sampling radius r_{min}. We store the edges in a priority-queue, with the edge length as the sorting score, such that the longest edge is in the front of the queue. We iteratively pop the first edge from the top of the queue, and split the edge by adding a new vertex in the middle point of the edge. Four new triangles are generated by adding a new edge that connects the new point and the vertex that is opposite to the edge that splits. Then, the new added edges are inserted in the queue and the weight of the effected edges is updated. This step terminates until all the edge lengths are smaller than r_{min}. This splitting process is illustrated in Fig. 3. In sub-figure (a), ΔABC is an input triangle. We first find that the edge length of AB is lager than r_{min}, and then we subdivide ΔABC into two triangles: ΔADC and ΔDBC. Here D is selected as the middle point of edge AB. In sub-figures (b) and (c), we use the same strategy to handle the long edges BC, AC and CD. Finally, we subdivide the input triangle into six small triangles and each edge length is smaller than r_{min}.

Once the splitting is done, we get a subdivided mesh $\mathcal{M}'$, which will be used as the sampling domain in later steps. The function of $\mathcal{M}'$ is similar to that of the grid structure used for conflict checking in Euclidean space [9,1], such that each triangle of $\mathcal{M}'$ can receive at most one sample point. Each triangle of $\mathcal{M}'$ is equipped with a buffer to record the indices of samples that fully cover it or intersect with it, as well as a label to indicate whether the triangle is fully covered by any sample point or not. This flag is initialized as *false*.

4.2. Initial sampling

In this step, we run the dart-throwing algorithm on the subdivided mesh. The main idea is that we iteratively sample a random point in the mesh, and perform the conflict checking to determine whether this new point can be accepted. The new point is accepted only if it is not covered by any previous samples and it does not cover any other existing samples in a locally defined neighborhood. Then the index of the new sample is recorded by the triangles that are fully covered by this sample or intersect with it. In detail, to generate a new point on the mesh, we first randomly select a triangle $t'_j \in \mathcal{M}'$ with the probability proportional to its weighted area (e.g., $\rho(\mathbf{c}_{t'_j})|t'_j|$, where $\mathbf{c}_{t'_j}$ is the barycenter, and $|t'_j|$ is the area of the triangle). Then we randomly generate a point $\mathbf{p}$ inside the selected triangle t'_j (we use the random point generator proposed by Turk [26]), as well as computing its corresponding sampling radius r_p from the sizing function.

Algorithm 1. Conflict checking.

1 Generate a random point $\mathbf{p}$ in a random triangle t'_j;

Queue $Q \leftarrow \emptyset$;
if t'_j *has been fully covered* **then**
  p is rejected. Return.
else
  push t'_j into Q;
  while $Q \neq \emptyset$ **do**
    pop the front element t'_i of Q;
    check if **p** conflicts with the previous samples recorded by t'_i;
    if *not* **then**
      index of **p** is recorded by t'_i;
      check if **p** fully covers or intersects with the neighboring triangle t'_k of t'_i;
      if *yes* **then**
        push t'_k into Q;
      else
        p is rejected. Return.

Conflict checking: Next, we propose a local region-growing-based method for conflict checking (Algorithm 1). In our approach, a FIFO queue Q is used to facilitate the propagation process. First, if t'_j has been fully covered by previous samples, then the new point **p** is rejected. Otherwise, we push t'_j into the queue. The region growing process is then performed by repeatedly popping triangles in the front of Q until the queue becomes empty. For each popped triangle t'_j, we traverse all the samples that intersect with this triangle and compute the distance between each sample **x** and the new point **p**. Once the distance is smaller than the radius of **x** or **p**, the point **p** is also rejected; otherwise, the index of **p** is recorded by the triangle t'_j. Then we detect each neighboring triangle t'_k (if it has not been visited) of t'_j and push t'_k into Q as long as **p** fully or partially covers it. Fig. 4 uses a 2D illustration to display our conflict checking process.

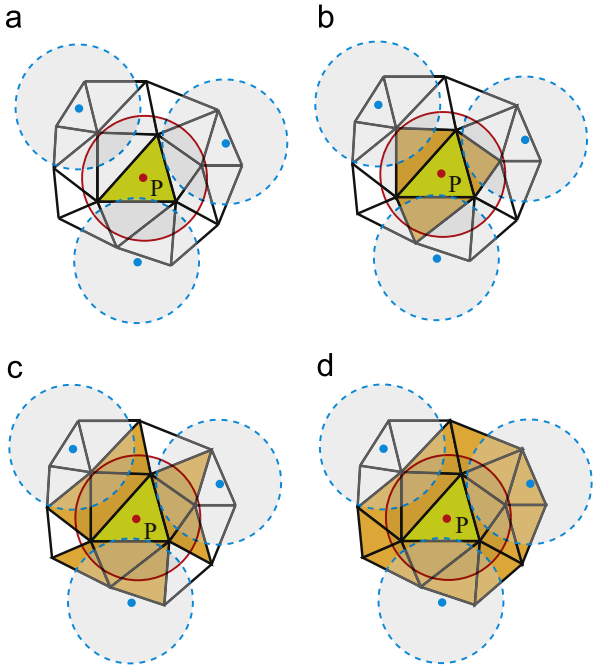


Fig. 4. (a) The yellow triangle is the randomly selected triangle t'_j, **p** is the randomly generated point in t'_j and the blue points are the previous samples. (b) Check for the neighboring triangles of t'_j. (c) and (d) propagate to other neighboring triangles. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

The initial sampling step terminates when k consecutive rejections are observed or the number l of throwing darts is larger than a threshold. In our implementation, we empirically set $k=300$, and $l=8m$ where m is the number of triangles in $\mathcal{M}'$.

4.3. Maximal sampling

As discussed in Section 2, the key issue of the maximal sampling is to track and fill the small regions that cannot be sampled by the classic dart-throwing. For this aim, we apply an iterative sampling approach to track/fill the empty regions, which is similar to [9].

Algorithm 2. Fragments filling.

Initially sampled point set P, sampling domain $\mathcal{M}'$, $i \leftarrow 1$;
repeat
  Subdivide each valid triangle in the fragment array of $level_i$ into 4 small triangles;
  Re−throw darts in the still valid subdivided triangles to get a new point set P';
  $i \leftarrow i+1$, update the fragment array;
until *achieve maximal sampling* P'

After the initial sampling stage, the triangles that are not fully covered are identified as valid and collected in a flat array. Each such triangle is called a "fragment". Then, at each iteration, we first traverse all the valid triangles in the fragment array of current $level_i$, and subdivide each of such triangle into four small triangles using the mid-point-insertion subdivision (see Fig. 5(b)). Next, we check the validness of these subdivided triangles (i.e., whether fully covered by neighboring disks) and collect all the valid triangles in a flat fragment array in the next $level_{i+1}$. Now we can re-throw darts in the fragments, where we use the same method as in the initial sampling stage to generate new points. In the worst case, in a fragment, the newly generated points which are not covered by existing samples may cover some existing samples. It will lead to non-maximal sampling. To overcome this problem, we use the method in [1] to recompute the radius of the new point by setting its radius as the distance to the nearest sample. By this approach, we will always sample an empty region. These two steps are repeated until all the fragments are fully covered or the subdivision process reaches a maximal level (e.g., 64). However, the latter is never happened in our experiments. In Fig. 5, we use the same example in Fig. 4(d) to demonstrate one iteration of filling fragments. We refer the reader to [9] for more details of the algorithm complexity analysis, which is similar to our case.

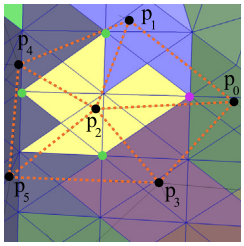
4.4. Mesh extraction

Once we have a maximal sampled Poisson-disk set, we are ready to extract the high-quality mesh from the samples. Instead of computing the restricted Power diagram [1], we propose a much simpler approach by using the discrete clustering [27,28] technique. Note that Guo et al. [29] use a similar clustering approach for triangulating Poisson-disk sampled point sets in 2D domain. Our approach includes two stages described as follows:

Initial mesh extraction: We first cluster the triangles in the subdivided mesh $\mathcal{M}$ into different patches using the variational shape approximation [27]. Each patch corresponds to one sample point. A priority queue PQ is maintained to store the triples $(t'_i, \mathbf{p}_j, d_{ij})$, where t'_i is a triangle of $\mathcal{M}'$, $\mathbf{p}_j$ is a sample point, d_{ij} is

the Euclidean distance between the triangle center and the sample. d_{ij} is used as the key for the priority queue, and the least distance has the highest priority. First, we push all the triangles that contain sample points as the initial seeds into the priority queue. While PQ is not empty, the seed element $(t_i', \mathbf{p}_j, d_{ij})$ in the front of PQ is popped out. For each popped $(t_i', \mathbf{p}_j, d_{ij})$, if t_i' has been assigned to a sample, then we discard it and pop next seed element; otherwise, we assign the index of $\mathbf{p}_j$ to t_i', and push the new element $(t_k', \mathbf{p}_j, d_{kj})$ into PQ, where t_k' is the neighboring triangle of t_i'. This process is performed repeatedly until the priority queue PQ becomes empty. Fig. 6(a) shows an example of the clustering result.

Next, we extract the mesh from the clustering. We consider each vertex $\mathbf{v}_i'$ of $\mathcal{M}'$, and check the 1-ring triangles around this vertex in a counterclockwise direction. If the number of different clusters that these triangles belong to is larger than two, we can extract a triangle fan using the corresponding samples. For example, in the embedded figure, black points are the samples, colored points are the input vertices. For the purple vertex, there are four clusters around it, so the samples $\mathbf{p}_0$, $\mathbf{p}_1$, $\mathbf{p}_2$ and $\mathbf{p}_3$ can form a triangle fan (dashed lines), including $\Delta\mathbf{p}_0\mathbf{p}_1\mathbf{p}_2$ and $\Delta\mathbf{p}_0\mathbf{p}_2\mathbf{p}_3$. After processing each vertex like this, we extract an initial triangular mesh.



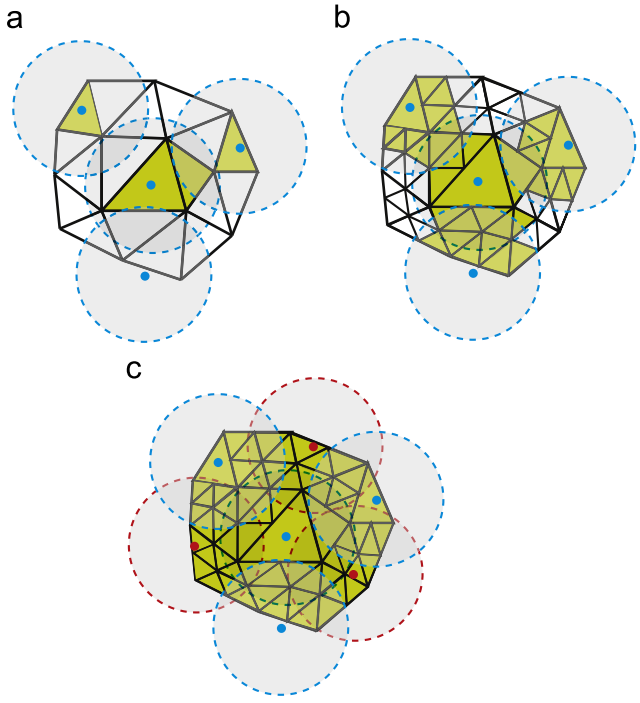


Fig. 5. (a) The yellow triangles are fully covered by samples and the light triangles are valid. (b) Each valid triangle is subdivided into 4 small triangles, and the fragments are in light color. (c) Generate three new samples (red) in the fragments then there are no valid triangles. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

Edge flipping: However, the initial remeshing is not guaranteed to be a Delaunay/regular triangulation. In this step, we use the algorithm that similar to [30,31] for edge flips. We omit the details here since this is not our contribution.

Note that using our mesh extraction algorithm, each clustered patch is singly connected and any two patches are not overlapping. As a result, there are no intersecting triangles in the extracted mesh.

4.5. Mesh optimization

In the remeshing literature, the most important characteristic of a triangle mesh is the angle bounds, which are required in many applications [32]. On the other hand, the valences of vertices often have an impact on how certain mesh processing algorithms perform, and irregular vertices are undesirable [33]. In this part, we provide an optional step to improve the remeshing quality, especially the properties of angle bound and vertex valence. In angle bound optimization, the user is required to specify the desired lower/upper angle bound. The vertices with one triangle angle that is bad (here "bad" means that this angle is not within the required angle bound) are removed. In valence optimization, the vertices whose degrees are less than 5 or larger than 7 are removed. Then we keep tracking of the new generated fragments and fill them using the method described in Section 4.3. The optimization step terminates when there are no bad angles and each vertex valence is 5, 6 or 7.

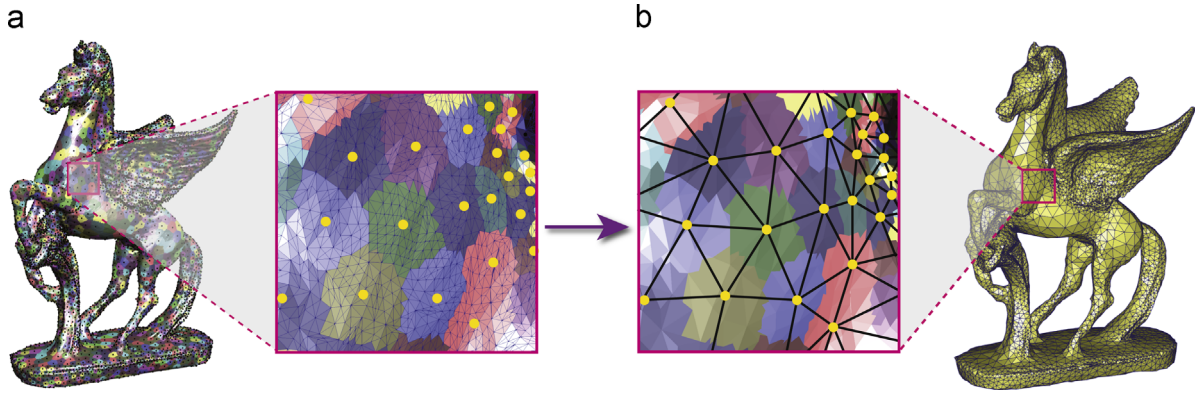


Fig. 6. (a) Clustering of the triangles in the subdivided mesh $\mathcal{M}'$. Each patch is encoded in a single color. (b) Remeshing result extracted using the clustering information. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

5. Experimental results

We demonstrate the experimental results to verify the effectiveness and the validity of the proposed algorithm. First, we perform the spectral analysis of the sampling results, as well as the remeshing quality of the extracted meshes. Next, we evaluate the performance and the memory usage of the proposed method.

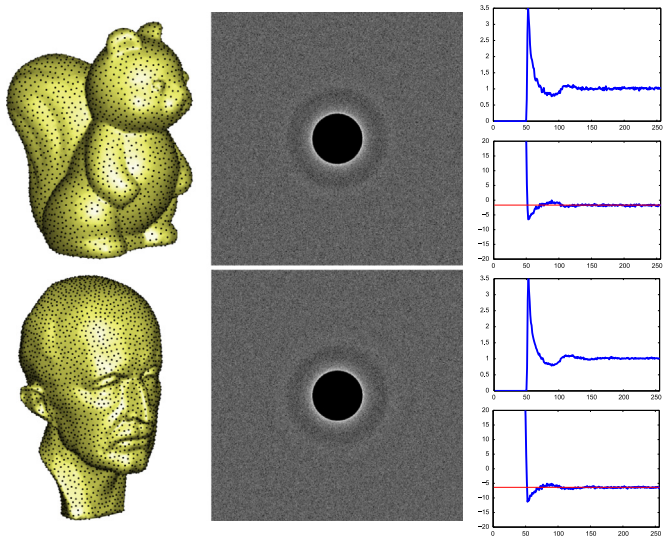


Fig. 7. Spectral analysis of the uniform sampling (with ~3500 samples) on Squirrel and Max-Planck.

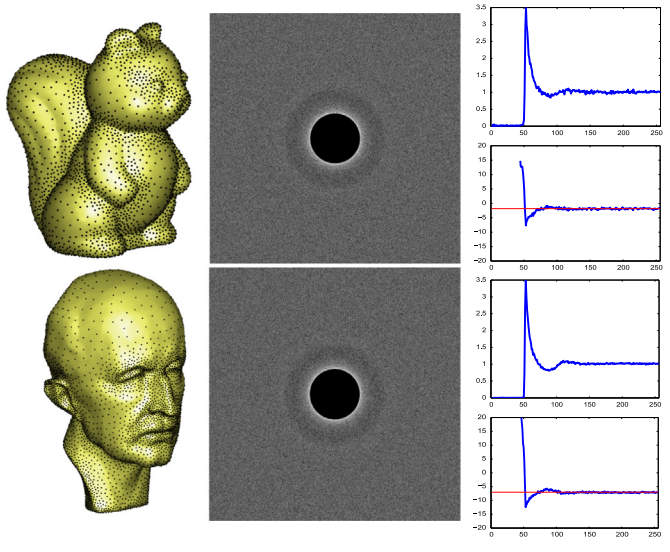


Fig. 8. Spectral analysis of the adaptive sampling (with ~5200 samples) on Squirrel and Max-Planck.

To the best of our knowledge, [1] is the only approach that can achieve adaptive maximal sampling on surfaces. Therefore, here we focus the comparisons of our algorithm with [1]. All the results presented in this paper are conducted on a PC with Intel i7-3770, 3.40 GHz CPU, 16 GB memory, and a 64-bit Windows 7 operating system.

Spectral analysis: We apply the technique of differential domain analysis [34] to analyze the spectral properties of the point sets sampled by our algorithm, including power spectrum, radially average and anisotropy. Figs. 7 and 8 illustrate these standard spectral measurements for uniform and adaptive samplings, respectively. Note that for the Squirrel model, the MPS point sets used for spectral analysis are generated by our algorithm without optimization. For the Max-Planck model, the MPS point sets have been optimized using our optimization framework. From this figure, we can see that our approach can generate high-quality samples that exhibit the blue noise characteristics. Furthermore, our optimization framework preserves the blue noise properties of the sampled point sets very well.

Remeshing quality: Our mesh extraction step accompanied with the mesh optimization framework can generate high-quality remeshings as the same as [1], shown in Table 1. We have verified that for uniform remeshing, our method produces meshes with the same angle bounds $[30^\circ, 120^\circ]$, edge length bounds $[r, 2r]$, and the triangle area bounds $[\frac{\sqrt{3}}{4}r^2, \frac{3\sqrt{3}}{4}r^2]$. These bounds are very important for applications like finite elements analysis [32].

However, thanks to our local clustering-based approach for mesh extraction, our framework can not only handle general input surfaces as in [1], but also the input surfaces that with thin-sheet structures or nearby regions, e.g., the mouth and legs of the Homer model. The Euclidean distance is used in [1] for conflict checking

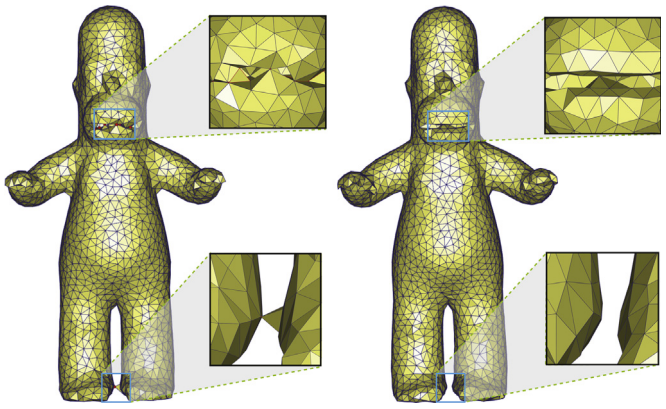


Fig. 9. Comparison of remeshing quality. Left: remeshing result of [1]. The non-manifold vertices and edges are shown in red and green colors, respectively; right: remeshing result of our algorithm. There are about 2000 vertices in each model. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

Table 1
Comparison of remeshing qualities with Yan and Wonka [1], using the models shown in Figs. 9 and 10. $|\mathbf{X}|$ is the number of sampled points. The definition of these standard quality measurements can be found in [1]. We omit them here.

Model	Method	$\|\mathbf{X}\|$	Q_{min}	Q_{avg}	$\theta_{\min}$	$\theta_{\max}$	V_{567}	d_H
Homer	Yan [1]	2.5K	0.611	0.913	33.63	105.51	100	0.28
	Our	2.5K	0.632	0.921	38.05	103.07	100	0.31
Gray's Klein bottle	Yan [1]	10.7K	0.627	0.918	39.97	102.81	100	0.25
	Our	10.7K	0.653	0.941	39.26	98.14	100	0.23
Klein bottle	Yan [1]	3.5K	0.615	0.896	34.30	104.35	99.6	0.35
	Our	3.5K	0.628	0.916	37.61	102.92	100	0.32

in a global uniform grid structure, and this strategy will cause some topological problems in thin-sheet regions or nearby regions that have small Euclidean distance but large geodesic distance. As shown in the left of Fig. 9, when the sampling radius is not adapted to the local feature size of the input mesh, the method of [1] will generate non-manifold topology in nearby regions (the lips and leg region). In contrast, since our algorithm uses a localized scheme, we can always obtain the valid topology in such difficult regions (see Fig. 9(right)). Fig. 10 shows more results that the input surfaces have thin-sheet structure (top) or even self-intersections (bottom).

Performance: We compare the algorithm efficiency with the previous work [1]. We use the same input mesh (12k triangles) and the same sampling radius for both methods. Fig. 11(a) shows the timing curves of our algorithm compared to [1]. As illustrated in this figure, our algorithm achieves 4–5 times speedup over [1] for both uniform and adaptive sampling. The speedup comes from two factors. First, the approach of [1] has to build the 3D regular triangulation and power diagram for gap computation and filling that limits the performance of the algorithm. While in our approach, we only perform a series of fragment sampling and subdivision. The algorithm always terminates after several levels of recursion (5–8 times in our tests). Second, a uniform gird is used in [1] for conflict checking. For a generated sample point with radius r_{cur}, a subset of $\lceil 2r_{cur}/r_{min}/\sqrt{3}\rceil^3$ grid cells will be checked. In our implementation, only a local patch of connected triangles will be traversed, which improves the speed drastically.

In addition, since our method generalizes the fast simple MPS algorithm of [9] to curved surfaces, here we provide a comparison with it. We use a triangulated planar square domain as our input and a unit square as the input of [9]. Fig. 11(c) shows that our running time is as fast as [9] for the 2D uniform sampling.

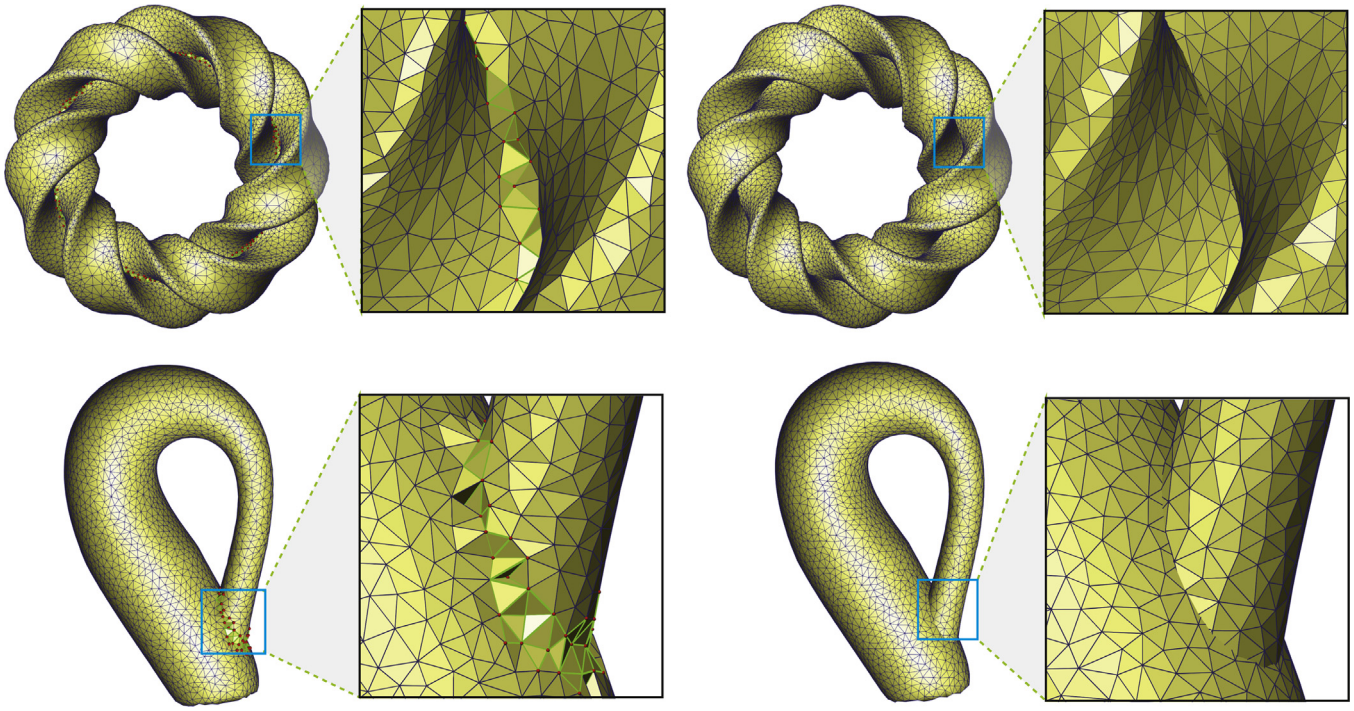


Fig. 10. More comparisons of the remeshing quality on thin-sheet regions (top) and self-intersecting surface (bottom). In each row, left: remeshing result of [1], where the non-manifold vertices and edges are shown in red and green colors, respectively; right: remeshing result of our algorithm. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

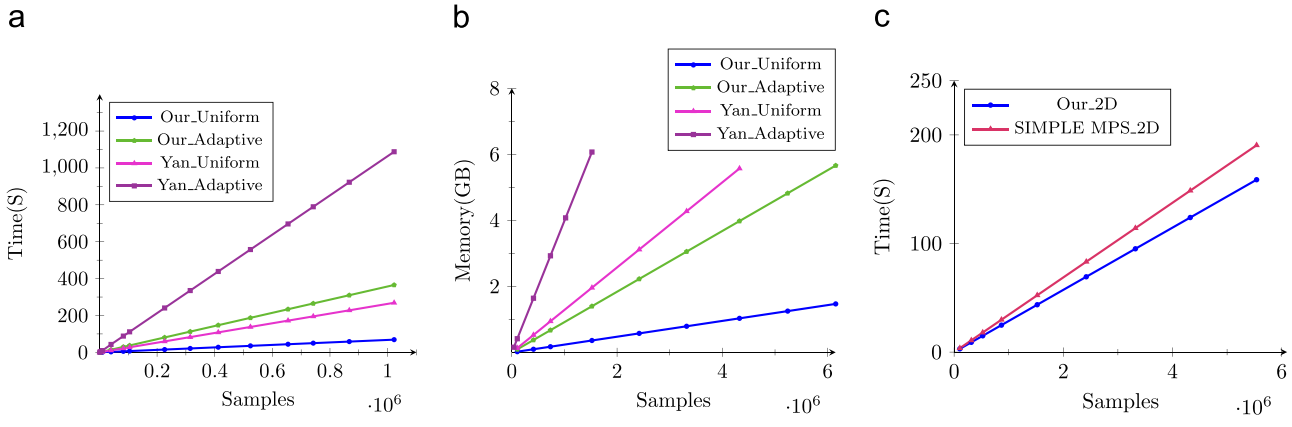


Fig. 11. The timing (a) and memory (b) comparison of our algorithm with Yan's algorithm [1]. (c) The timing comparison of our algorithm with the fast simple MPS [9]. *Note that the comparison is based on our own implementation of [9], the code might not be fully optimized.*

Memory usage: We compare the memory consumption of ;our algorithm with the previous method [1] in Fig. 11(b). Our memory usage mainly includes the triangle buffer of the subdivided mesh $\mathcal{M}'$, the results buffer of points and the extra memory used for local conflict checking and storage of the empty regions. Although the uniform 3D grid used in [1] is much simpler, it requires to store other additional information for conflict checking. In addition, the extra memory is also used in [1] to store the regular triangulation and the power diagram. Fig. 11(b) shows that our algorithm uses a factor of 7 less memory than [1] for uniform sampling and a factor of 5 less memory for the adaptive sampling.

6. Conclusion

This paper presents a simple method for maximal Poisson-disk sampling on mesh surfaces. Our approach improves both the sampling quality and the algorithmic efficiency upon previous work by using a simpler data structure for conflict checking and empty region tracking. Unlike the previous work [1], we are able to handle mesh surfaces with thin-sheets structure and even with self-intersections. One limitation of our work is that the connectivity information of the input mesh is required, so that we are not able to handle triangle soups or noisy data.

In the future, we would like to look for a GPU implementation to further improve the efficiency. We are also interested in generating MPS inside 3D shapes for volumetric tetrahedral meshing.

Acknowledgments

We would like to thank the anonymous reviewers for their suggestive comments. This work was partially funded by the National Natural Science Foundation of China (61372168, 11201463, 61271431, and 61331018), and the KAUST Visual Computing Center the KAUST Visual Computing Center, and the open funding project of the State Key Laboratory of Virtual Reality Technology and Systems, Beihang University (Grant No. BUAA-VR-14KF-10).

References

[1] Yan D-M, Wonka P. Gap processing for adaptive maximal Poisson-disk sampling. ACM Trans Graph 2013;32(5):148:1–15.
[2] Ebeida MS, Mitchell SA, Davidson AA, Patney A, Knupp PM, Owens JD. Efficient and good Delaunay meshes from random points. Comput Aided Des 2011;43 (11):1506–15.
[3] Cook RL. Stochastic sampling in computer graphics. ACM Trans Graph 1986;5 (1):69–78.
[4] Dunbar D, Humphreys G. A spatial data structure for fast Poisson-disk sample generation. ACM Trans Graph (Proc SIGGRAPH) 2006;25(3):503–8.
[5] Jones TR. Efficient generation of Poisson-disk sampling patterns. J Graph Tools 2006;11(2):27–36.
[6] White KB, Cline D, Egbert PK. Poisson disk point sets by hierarchical dart throwing. In: Proceedings of the IEEE symposium on interactive ray tracing; 2007. p. 129–32.
[7] Gamito MN, Maddock SC. Accurate multidimensional Poisson-disk sampling. ACM Trans Graph 2009;29(1):8:1–19.
[8] Ebeida MS, Patney A, Mitchell SA, Andrew Davidson PMK, Owens JD. Efficient maximal Poisson-disk sampling. ACM Trans Graph (Proc SIGGRAPH) 2011;30 (4):49:1–12.
[9] Ebeida MS, Mitchell SA, Patney A, Davidson AA, Owens JD. A simple algorithm for maximal Poisson-disk sampling in high dimensions. Comput Graph Forum (Proc EUROGRAPHICS) 2012;31(2):785–94.
[10] Wei L-Y. Parallel Poisson disk sampling. ACM Trans Graph (Proc SIGGRAPH) 2008;27(3):20:1–9.
[11] Mitchell SA, Rand A, Ebeida MS, Bajaj CL. Variable radii Poisson disk sampling. In: 24th Canadian conference on computational geometry (CCCG); 2012. p. 185–90.
[12] Cline D, Jeschke S, Razdan A, White K, Wonka P. Dart throwing on surfaces. Comput Graph Forum (Proc EGSR) 2009;28(4):1217–26.
[13] Bowers J, Wang R, Wei L-Y, Maletz D. Parallel Poisson disk sampling with spectrum analysis on surfaces. ACM Trans Graph(Proc SIGGRAPH Asia) 2010;29(6):166:1–10.
[14] Corsini M, Cignoni P, Scopigno R. Efficient and flexible sampling with blue noise properties of triangular meshes. IEEE Trans Vis Comput Graph 2012;18 (6):914–24.
[15] Ying X, Xin S-Q, Sun Q, He Y. An intrinsic algorithm for parallel Poisson disk sampling on arbitrary surfaces. IEEE Trans Vis Comput Graph 2013;19 (9):1425–37.
[16] Ying X, Li Z, He Y. A parallel algorithm for improving the maximal property of poisson disk sampling. Comput Aided Des 2014;46(9):37–44.
[17] Dong-Ming Yan, Johannes Wallner, Peter Wonka. Unbiased Sampling and Meshing of Isosurfaces. IEEE Trans. Vis. Comput. Graph. 2014;20(11):1579–89.
[18] Fu Y, Zhou B. Direct sampling on surfaces for high quality remeshing. In: ACM symposium on solid and physical modeling; 2008. p. 115–24.
[19] Lloyd SA. Least squares quantization in PCM. IEEE Trans Inf Theory 1982;28 (2):129–37.
[20] Yan D-M, Lévy B, Liu Y, Sun F, Wang W. Isotropic remeshing with fast and exact computation of restricted Voronoi diagram. Comput Graph Forum 2009;28(5):1445–54.
[21] Liu Y, Wang W, Lévy B, Sun F, Yan D-M, Lu L, et al. On centroidal Voronoi tessellation—energy smoothness and fast computation. ACM Trans Graph 2009;28(4):101:1–17.
[22] Chen Z, Yuan Z, Choi Y-K, Liu L, Wang W. Variational blue noise sampling. IEEE Trans Vis Comput Graph 2012;18(10):1784–96.
[23] Xu Y, Hu R, Gotsman C, Liu L. Blue noise sampling of surfaces. Comput Graph 2012;36(4):232–40.
[24] Chen J, Ge X, Wei L-Y, Wang B, Wang Y, Wang H, et al. Bilateral blue noise sampling. ACM Trans Graph 2013;32(6):216:1–216:11.
[25] Amenta N, Bern M, Kamvysselis M. A new Voronoi-based surface reconstruction algorithm. In: Proceedings of ACM SIGGRAPH; 1998. p. 415–21.
[26] Turk G. Generating random points in triangles. In: Graphics gems; 1993. p. 24–8.
[27] Cohen-Steiner D, Alliez P, Desbrun M. Variational shape approximation. ACM Trans Graph (Proc SIGGRAPH) 2004(23):905–14.
[28] Valette S, Chassery J-M, Prost R. Generic remeshing of 3D triangular meshes with metric-dependent discrete Voronoi diagrams. IEEE Trans Vis Comput Graph 2008;14(2):369–81.
[29] Guo J, Yan D-M, Bao G, Dong W, Zhang X, Wonka P. Efficient triangulation of Poisson-disk sampled point sets. Vis Comput 2014;30(6–8):773–85.
[30] Dyer R, Zhang H, Möller T. Delaunay mesh construction. In: Proceedings of symposium of geometry processing; 2007. p. 273–82.
[31] Pan H, Choi Y-K, Liu Y, Hu W, Du Q, Polthier K, et al. Robust modeling of constant mean curvature surfaces. ACM Trans Graph (Proc SIGGRAPH) 2012;31 (4):85:1–85:11.
[32] Du Q, Wang D, Zhu L. On mesh geometry and stiffness matrix conditioning for general finite element spaces. SIAM J Numer Anal 2009;47(2):1421–44.
[33] Aghdaii N, Younesy H, Zhang H. 5–6–7 meshes: remeshing and analysis. Comput Graph 2012;36(8):1072–83.
[34] Wei L-Y, Wang R. Differential domain analysis for non-uniform sampling. ACM Trans Graph (Proc SIGGRAPH) 2011;30(4):50:1–8.
[35] Dong-Ming Yan, Jianwei Guo, Xiaohong Jia, Xiaopeng Zhang, Peter Wonka. Blue-Noise Remeshing with Farthest Point Optimization. Computer Graphics Forum (Prof. SGP) 2014;33(5):167–76.
[36] T. Schlömer, D. Heck and O. Deussen, 2011. Farthest-Point Optimized Point Sets with Maximized Minimum Distance. In: High Performance Graphics Proceedings, 135-1420.