Gap Processing for Adaptive Maximal Poisson-Disk Sampling

DONG-MING YAN
King Abdullah University of Science and Technology
and
PETER WONKA
King Abdullah University of Science and Technology

In this paper, we study the generation of maximal Poisson-disk sets with varying radii. First, we present a geometric analysis of gaps in such disk sets. This analysis is the basis for maximal and adaptive sampling in Euclidean space and on manifolds. Second, we propose efficient algorithms and data structures to detect gaps and update gaps when disks are inserted, deleted, moved, or when their radii are changed. We build on the concepts of regular triangulations and the power diagram. Third, we show how our analysis contributes to the state of the art in surface remeshing.

Categories and Subject Descriptors: I.3.6 [**Computer Graphics**]: Methodology and Techniques

Additional Key Words and Phrases: Adaptive Poisson-disk sampling, maximal sampling, gaps, regular triangulation, power diagram, blue noise, remeshing

ACM Reference Format:

Yan, D.-M., and Wonka, P.. 2013. Gap Processing for Adaptive Maximal Poisson-Disk Sampling. ACM Trans. Graph. 28, 4, Article xxx (xxx 2013), 15 pages.
DOI = xxx
http://doi.acm.org/xxx

1. INTRODUCTION

Maximal Poisson-disk sampling (MPS) can generate point sets with interesting properties. One issue in MPS is how to locate and approximate the gaps in an already sampled disk set.

This research was partially funded by National Natural Science Foundation of China (No. 61271431, 61172104 and 61271430), and the National Science Foundation.

Authors' addresses: D.-M. Yan (corresponding author), KAUST, email: yandongming@gmail.com; P. Wonka (corresponding author), KAUST, email: pwonka@gmail.com.

DOI 10.1145/XXXXXXX.YYYYYYY
http://doi.acm.org/10.1145/XXXXXXX.YYYYYYY

In this paper, we study the geometry of gaps in disk sets. Given a set of disks with varying radii in a compact domain Ω in Euclidean space or on a manifold, we would like to know if they fully cover the domain or if there is a gap, i.e., if there is uncovered space to insert other disks into the disk set. We are interested in knowing if gaps exist, where the gaps are, and how to implement efficient gap processing operations.

There are three important papers that conduct such an analysis in the context of uniform 2D MPS: Dunbar and Humphreys [2006], Jones [2006], and Ebeida et al. [2011]. In the first part of this paper, we present a simpler and more natural analysis that additionally extends to 1) disk sets with varying radii, 2) arbitrary dimensions, and 3) 2-manifolds. The main idea of our approach is to study gaps in the context of the regular triangulation and the power diagram [Aurenhammer 1991] of the disk sets.

Besides the general curiosity about an interesting geometric problem, gaps in disk sets play an important role in sampling applications, e.g., sample generation for ray tracing, image stippling, video stippling, environment map sampling, surface remeshing, plant ecosystem simulation, texture synthesis, video synthesis, and particle-based simulation. By analyzing this set of interesting applications to find commonalities, we identified several operations that need to be performed efficiently: gap detection, gap clustering, gap primitive extraction, updating gap primitives when points are deleted or inserted, updating gap primitives when points move, and updating gap primitives when the sampling radius changes. The second part of the paper will introduce algorithms for these operations, based on the analysis in the first part.

While our algorithms do not improve all aforementioned applications, in the third part of this paper, we investigate an application to surface remeshing (as well as 2D meshing), where we can successfully improve the state of the art in aspects such as minimal angle, vertex valence and triangle quality. We discuss why remeshing benefits from blue noise properties, maximal sampling, bounds on vertex valence, and geometric bounds (e.g., angle bounds). These properties are important to simulation applications. For example, Schechter and Bridson [2012] demonstrated that the simulation result based on Poisson-disk sampling performs much better than that based on the regular grid sampling. We propose a remeshing framework that jointly optimizes these criteria and evaluate it in a comparison to several recent remeshing algorithms. The main contributions of these three parts are:

—A simple and elegant theoretical analysis of the gap geometry that improves the work of [Ebeida et al. 2011] for 2D, and is the first analysis for d-D and manifolds.

—The design of efficient algorithms and data structures for all the gap processing operations identified above.

—A surface remeshing (as well as 2D meshing) algorithm that compares favorably to the state of the art in aspects such as minimal angle, vertex valence and triangle quality.

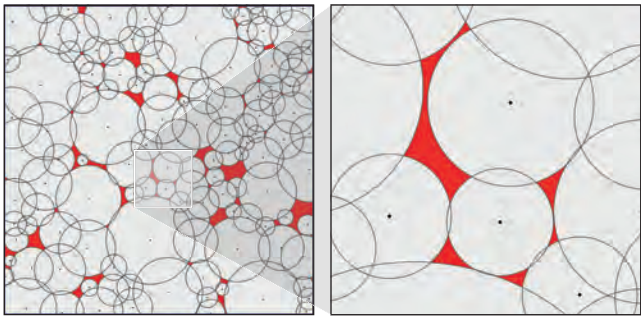


Fig. 1: *Left: Illustration of gaps of a disk set; the uncovered regions are shown in red. Right: zoom in view of the uncovered regions.*

1.1 Related work

We briefly review techniques for generating Poisson-disk point sets, but refer the reader to [Lagae and Dutré 2008] for a more comprehensive survey. Our review focuses on the treatment of gaps in these algorithms.

There are a large number of sampling algorithms that produce point sets with different attributes [Lloyd 1982; Dippé and Wold 1985; Cook 1986; Mitchell 1987; 1991; McCool and Fiume 1992; Turk 1993] that are either not concerned with the geometry of gaps or do not use an acceleration data structure. Our work can improve upon some of these methods, e.g., [McCool and Fiume 1992].

Poisson-disk sampling. Most recent algorithms for efficient Poisson-disk sampling maintain a data structure of gap primitives. The simplest primitive is a square, because it is easy to sample [White et al. 2007; Bridson 2007; Gamito and Maddock 2009; Ebeida et al. 2012] and it can be efficiently subdivided. These algorithms are extremely efficient when trying to fill a large mainly uncovered region, but even when filling small gaps, they are efficient enough so that the overall running time is much faster than the algorithm built on an exact representation of gaps [Dunbar and Humphreys 2006]. Ebeida et al. [2011] propose a hybrid approach that first uses squares and later convex polygons bounding the intersections of a square and multiple circles as gap primitives. A two-step Poisson-disk sampling framework is proposed that first performs dart-throwing in the grid and then against the extracted gaps. The maximal property is achieved and the sampling framework is shown to be more efficient then previous approaches. Jones [2006] uses a Voronoi diagram to extract gap primitives. The algorithm repeatedly generates disks and inserts them in a global Voronoi diagram one by one. Each vertex in the Voronoi diagram maintains a value that indicates the area of empty region of the corresponding Voronoi cell. A new sampling is generated by first selecting a Voronoi cell based on the empty area. The Voronoi diagram and the value of each vertex is updated each time a new sample is generated. Jones' algorithm uses the fact that a maximal sampling can be obtained if and only if the Voronoi cell of each vertex is fully covered by the disk centered at the vertex. However, Jones [2006] as well as the other approaches discussed above handle only 2D uniform sampling. In contrast, we use a power diagram and the dual regular triangulation for our analysis, which results in a more general and much simpler solution.

Adaptive sampling. Fattal presents an adaptive sampling algorithm based on kernel density estimation [Fattal 2011]. Kalantari et al. [2011] propose to use joint distributions by breaking down the 2D probability density function (pdf) into a 1D conditional pdf (cpdf) and a 1D pdf for maximal adaptive sampling. However, the proposed algorithm is dependent on a threshold for discretizing the 1D pdf, which cannot achieve maximal sampling. In concurrent work, Mitchell et al. [2012] study 2D Poisson-disk sampling with various radii, and de Goes et al. formulate the blue noise sampling problem using optimal transport [de Goes et al. 2012]. However, the maximality is not discussed in these approaches.

Sampling on surfaces. Fu and Zhou [2008] generalize scalloped sector based sampling [Dunbar and Humphreys 2006] to 3D mesh surfaces, and present an isotropic remeshing algorithm by extracting a mesh from the samples. Lloyd iterations are used to further smooth the resulting mesh. Cline et al. [2009] propose dart throwing algorithms on surfaces based on a hierarchical triangulation. Bowers et al. [2010] extend the parallel sampling method [Wei 2008] to mesh surfaces and introduce a spectral analysis for uniform surface sampling algorithms as well. Wei and Wang [2011] present a framework for spectral analysis of nonuniform blue noise sampling for both 2D domains and surfaces. Again, none of these methods satisfy the maximal sampling property. Xu et al. [2012] extend the concept of *Capacity-Constrained Voronoi Tessellation* (CCVT) [Balzer et al. 2009] by introducing capacity-constrained Delaunay triangulations for blue noise sampling on surfaces. Chen et al. [2012] combine the CCVT [Balzer et al. 2009] and the *Centroidal Voronoi Tessellation* (CVT) framework [Yan et al. 2009] for blue noise sampling on surfaces. However, the approaches of Chen et al. [2012] and Xu et al. [2012] are also based on the Lloyd iterations. Corsini et al. [2012] present an algorithm for surface blue noise sampling based on a space subdivision combined with a pre-generation of the samples. Although many surface blue noise sampling algorithms have been proposed recently, few of them use blue noise sampling for geometric applications, such as remeshing.

Other approaches. There are several extensions to Poisson-disk sampling that we did not consider in our current framework, but that might be interesting avenues for future work, e.g., parallel sampling [Wei 2008; Gamito and Maddock 2009] and multi-class sampling [Wei 2010]. Schlömer et al. [2011] proposed the use of the farthest Voronoi diagram for blue noise sampling. Some approaches pre-compute tile sets and then quickly arrange them in realtime [Ostromoukhov et al. 2004; Kopf et al. 2006]. Our work might be beneficial for the tile pre-computation phase.

2. THEORETICAL GAP ANALYSIS

We study a set of disks $\mathcal{D} = \{(\mathbf{p}_i, r_i)\}_{i=1}^{n}$ in a compact d-D domain Ω, where $\mathbf{p}_i$ and r_i are the center and the radius of the i^{th} disk, respectively. Here, we use the word *disk* to represent a general disk, i.e., a sphere in 3D and a hyper-sphere in d-D ($d > 3$). We assume that the center $\mathbf{p}_i$ of any disk cannot be covered by the other disks, i.e., $\forall i, j\,(i \neq j), \|\mathbf{p}_i - \mathbf{p}_j\| \geq \max(r_i, r_j)$. If we draw a disk at each center $\mathbf{p}_i$ with radius r_i, the domain Ω will be fully or partially covered. We are interested in the properties of the uncovered region, which is defined as $\Omega - \mathcal{D}$. As shown in Figure 1, if the domain Ω is partially covered by a set of disks, the uncovered region is split into a set of isolated regions. We use the term *gap* to refer to a connected component of the uncovered region.

At first glance, a gap can be arbitrarily complex or can be arbitrarily small so that it is very difficult to sample gaps directly. Therefore, like most previous work, we shall decompose these complex gaps to a set of gap primitives GP_i such that 1) the union of these gap primitives covers the gaps $\Omega - \mathcal{D} \subseteq \cup GP_i$, 2) the gap primitives are non-intersecting $\forall i, j\,(i \neq j), GP_i \cap GP_j = \emptyset$, and 3) it is easy to sample a gap primitive using rejection sampling.

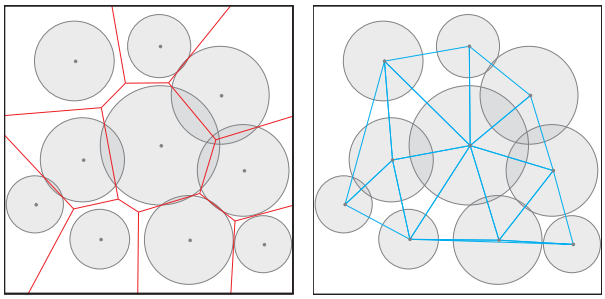


Fig. 2: *The power diagram (left) and the regular triangulation (right) of a set of disks.*

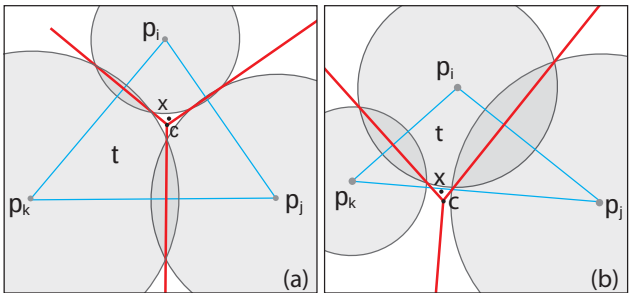


Fig. 3: *Existence of gaps in the 2D plane. The power center* $\mathbf{c}$ *of a triangle* $\triangle \mathbf{p}_i\,\mathbf{p}_j\,\mathbf{p}_k$ *can be used to test for the existence of gaps.*

2.1 Power diagram and regular triangulation

We analyze the geometric properties of gaps based on the concept of the power diagram [Aurenhammer 1987], and its dual, called regular triangulation. The power diagram is a tessellation of the Euclidean space into a set of convex polygons defined by a set of disks. Figure 2 shows an example of the power diagram of a set of disks. The power diagram and the regular triangulation are equivalent to the Voronoi diagram and the Delaunay triangulation when the radius of all the points are the same.

The disk set $\mathcal{D}$ is represented by a set of weighted points $\mathcal{P}_w = \{(\mathbf{p}_i, w_i)\}_{i=1}^n$, where $w_i = r_i^2$. The power of two weighted points is defined as

$$\Pi(\mathbf{p}_i, w_i, \mathbf{p}_j, w_j) = \|\mathbf{p}_i - \mathbf{p}_j\|^2 - w_i - w_j.$$

Then, the power diagram $\mathcal{PD}$ of $\mathcal{P}_w$ is a set of non-overlapping power cells $\{\Omega_i\}_{i=1}^n$, such that

$$\Omega_i = \{\mathbf{x} \in \Omega \mid \Pi(\mathbf{p}_i, w_i, \mathbf{x}, 0) \leq \Pi(\mathbf{p}_j, w_j, \mathbf{x}, 0), \forall j \neq i\}.$$

A vertex of the power diagram is called a *power vertex* and an edge is called a *power edge*. A d-D power diagram can be interpreted as the intersection of a $(d+1)$-D Voronoi diagram and a d-D hyperplane [Ash and Bolker 1986].

The regular triangulation of $\mathcal{P}_w$ is a d-D simplicial complex $\mathcal{T}$ and it is the dual of the power diagram. A d-D simplicial complex is the union of a set of k-simplices ($0 \leq k \leq d$). Each k-simplex is the convex hull of $k+1$ linear independent points that are dual to a $(d-k)$-D convex polyhedron of the power diagram. The simplicial complex $\mathcal{T}$ satisfies: i) any face of a simplex is also in $\mathcal{T}$, and ii) the intersection of any two simplices is either empty, or is a face of both simplices. We denote the set of the d-simplices (i.e., triangles in 2D and tetrahedra in 3D) as $\mathcal{RT} = \{t_j\}$. For each $t \in \mathcal{RT}$, there exists a point $\mathbf{c}_t \in \Omega$ that has equal powers w.r.t. the $d+1$ vertices of t, and this power is less than the power of $\mathbf{c}_t$ w.r.t. any other weighted points in the triangulation. Each $\mathbf{c}_t$ is called the *power center* of t, which is a power vertex of $\mathcal{PD}$. We denote the power of a d-simplex t by $\Pi(t) = \Pi(\mathbf{p}, w, \mathbf{c}_t, 0)$, where $\mathbf{p}$ refers to one of the vertices of t, and w is the corresponding weight of $\mathbf{p}$.

2.2 Existence of gaps

In the following, we give the theorems that state the condition of the existence of gaps in d-D spaces and on 3D surfaces, respectively.

THEOREM 1. *A gap exists iff* $\exists\, t \in \mathcal{RT}, \Pi(t) > 0$.

Here, we give the proof only in 2D (see Figure 3), as it is more intuitive and the proof for higher dimensions is similar.

PROOF. $\Rightarrow$ Suppose that there is a gap. For any point $\mathbf{x}$ in this gap, we define the function $F(\mathbf{x})$ as the minimal power between the point $\mathbf{x}$ and the weighted points $\mathcal{P}_w$. Suppose that $\mathbf{x}$ falls in the power cell Ω_i of the sample $(\mathbf{p}_i, w_i)$. Clearly, $F(\mathbf{x}) = \Pi(\mathbf{x}, \mathbf{p}_i)$ by definition of the power diagram. $F(\mathbf{x})$ attains its maxima when $\mathbf{x}$ coincides with a power vertex $\mathbf{c}$, which is the power center of a triangle $t \in \mathcal{RT}$. When $\mathbf{x}$ is in the gap, by definition $F(\mathbf{x}) > 0$ since $\Pi(\mathbf{x}, \mathbf{p}_i) > 0$. Note that $\Pi(t) = \Pi(\mathbf{c}, \mathbf{p}_i) = \max(F(\mathbf{x}))$. Hence, $\Pi(t) > F(\mathbf{x}) > 0$.

$\Leftarrow$ If $\exists t$, $\Pi(t) > 0$, then the power center of the triangle t is not covered. Therefore, there exists a gap. □

Gaps on 3D surfaces. If the input domain Ω is a 3D surface, Thm. 1 does not apply since the domain is not flat. In this case, we generalize the approach of Jones [2006] to 3D surfaces using the concept of the *Restricted Power Diagram* ($\mathcal{RPD}$).

The restricted power diagram is a generalization of the *Restricted Voronoi Diagram* (RVD) [Edelsbrunner and Shah 1997] on surfaces. The $\mathcal{RPD}$ is defined as the intersection of the 3D power diagram and the surface, i.e., $\mathcal{RPD}(\mathcal{P}_w) = \mathcal{PD}(\mathcal{P}_w) \bigcap \Omega = \{\Omega_i \bigcap \Omega\}_{i=1}^n$.

THEOREM 2. *A gap in the disk set* $\mathcal{D}$ *(sphere set in 3D) on a 3D surface exists iff there exists a sample* $\mathbf{p}_i$ *whose corresponding restricted power cell is not fully covered by the sphere* $(\mathbf{p}_i, r_i)$ *centered at* $\mathbf{p}_i$.

Thm. 2 describes the global condition of gap existence on surfaces. The proof is straightforward and we omit it here. In Section 5.3, we show how to locally detect gaps on mesh surfaces in the context of ϵ-sampling and *Restricted Regular Triangulations* ($\mathcal{RRT}$).

3. 2D GAP PRIMITIVE PROCESSING

In the following, we describe the most important operations related to gap processing in the 2D plane. In the following discussion, a triangle t is called a *gap triangle* if $\Pi(t) > 0$, or in other words, the power center of the triangle is not covered by the three disks centered on the vertices of the triangle. We provide a short overview of each of these operations here and in the subsections we describe each operation in more detail:

—Gap detection: gap detection refers to analyzing an existing point set to determine if gaps exist and where they are. If no gap exists, we can determine that a point set is maximal. In our solution, this operation outputs a list of gap triangles (see Section 3.1).

—Gap clustering: this operation groups all gap triangles belonging to the same independent gap set (Section 3.2) together. In our proposed solution, this clustering is performed on the level of

gap triangles, before the actual gaps are extracted. This step is optional but it enables a simple parallelization of gap sampling.

—Gap primitive extraction: given a set of input points, this operation computes a set of gap primitives covering all gaps (Section 3.3).

—Gap primitive updates: these operations update the gap primitives after points are inserted, deleted, moved, or when one or multiple disk radii are changed (Section 3.4).

The state of the framework is the weighted point set $\mathcal{P}_w$ and the corresponding regular triangulation $\mathcal{RT}$. At the beginning, $\mathcal{RT}$ has to be initialized once.

3.1 Gap detection

We can optionally collect all the gap triangles in an array for later use or simply return a Boolean to indicate the existence of gaps. Note that a gap triangle does not necessarily mean that the triangle itself contains some uncovered region, since its power center can be outside the triangle. See Figure 5(b) for an example where the triangle t_0 is fully covered but $\Pi(t_0) > 0$. The incident gap region is outside of the triangle t_0. Also, since a gap might contain several triangle power centers, one gap can have several corresponding gap triangles. We traverse all the triangles in $\mathcal{RT}$, compute the power center c_i and power $\Pi(t_i)$ of each triangle t_i. If $\Pi(t_i) > 0$, triangle t_i is marked as a *gap triangle* according to Thm. 1.

3.2 Gap clustering

We say that two gaps are dependent, if a disk inserted in one gap might intersect the other gap. We cluster all dependent gaps into *independent gap sets* (IGS), which can be used later for parallel sampling. With slight abuse of notation, we sometimes also use IGS to refer to the corresponding collection of gap triangles or gap primitives.

It is clear that connected gaps are dependent. However, even disconnected gaps can be dependent if the sampling radius inside the gaps is larger than the distance between the gaps. This is only a sufficient condition, because additionally a disk can only be placed if it does not cover another existing disk center. Since it is difficult to exactly compute the dependence of gaps, we propose a conservative approach to compute IGS clustering: all gap triangles in the one-ring neighborhood of each other are clustered in the same IGS (Figure 4). While we checked that the clustering is conservative in praxis, we do not offer a formal proof in this paper.

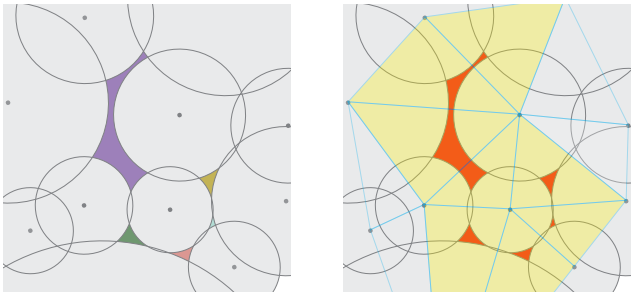


Fig. 4: *Gap clustering. There are five gaps (left) that clustered together forming an independent gap set (right) by the clustering algorithm.*

In our implementation, the algorithm is performed for gap triangles, but there is a simple one-to-one correspondence to gap primitives (see Section 3.3). All the gap triangles are marked as unvisited

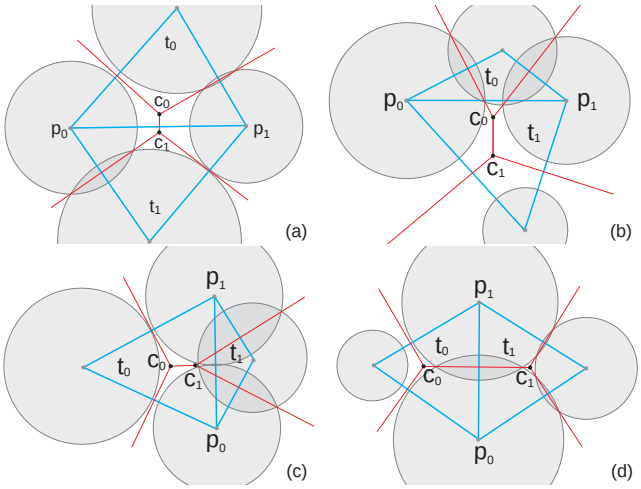


Fig. 5: *Illustration of the connectivity between two neighboring gap triangles t_0, t_1 (blue triangles) where c_0, c_1 are the corresponding power centers, respectively. $\mathbf{p}_0\,\mathbf{p}_1$ are on the common edge shared by t_0, t_1. (a) $|\mathbf{p}_0\,\mathbf{p}_1| > r_0 + r_1$, and c_0, c_1 are on the different sides of $\mathbf{p}_0\,\mathbf{p}_1$. (b) $|\mathbf{p}_0\,\mathbf{p}_1| > r_0 + r_1$, and c_0, c_1 are on the same side of $\mathbf{p}_0\,\mathbf{p}_1$. (c) $|\mathbf{p}_0\,\mathbf{p}_1| < r_0 + r_1$, and c_0, c_1 are on the same side of $\mathbf{p}_0\,\mathbf{p}_1$. (d) $|\mathbf{p}_0\,\mathbf{p}_1| < r_0 + r_1$, but c_0, c_1 are on the different sides of $\mathbf{p}_0\,\mathbf{p}_1$.*

at the beginning. We then traverse all the gap triangles. Each time when we encounter an unvisited triangle, we extract the IGS using a simple region growing algorithm, i.e., if two neighboring triangles are gap triangles, then they belong to the same group of an IGS. The grouped triangles are marked as visited and the traversal continues. This step ends when all the gap triangles are marked as visited. There are three cases of the relationship between two neighboring gap triangles, which are called *connectivity rules*. To cluster IGSs, all three rules apply, but if only connected components are desired, the last rule does not apply.

Connectivity rules. Given a pair of neighboring gap triangles t_0, t_1, and their power centers $\mathbf{c}_0, \mathbf{c}_1$, respectively (Figure 5), we use the following connectivity rules:

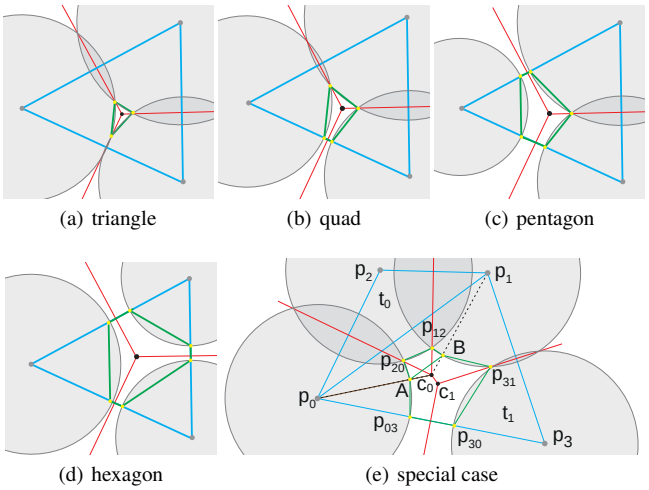
—The length of the common edge shared by t_0, t_1 is larger than $r_0 + r_1$, where r_0, r_1 are the corresponding sampling radii at the vertices of the common edge $\mathbf{p}_0, \mathbf{p}_1$ (Figure 5(a) and 5(b)).

—The length of the common edge shared by t_0, t_1 is smaller than $r_0 + r_1$, but $\mathbf{c}_0, \mathbf{c}_1$ are on the same side of the common edge (Figure 5(c)).

—The length of the common edge shared by t_0, t_1 is smaller than $r_0 + r_1$, and $\mathbf{c}_0, \mathbf{c}_1$ are on different sides of the common edge (Figure 5(d)). In this case, although two gaps are disconnected, putting a new disk in one empty region may affect the geometry on the other side. The two triangles are therefore classified as belonging to the same IGS.

3.3 Gap primitive extraction

The algorithm extracts a *gap primitive* for each triangle of the gap separately. Each gap primitive is a simple convex polygon with up to six edges. The *connectivity rules* are used to assist the extraction. Given a triangle t_0 belonging to a gap, we simply traverse the three neighbors of t_0 and extract the vertices of the gap primitive based



(a) triangle (b) quad (c) pentagon

(d) hexagon (e) special case

Fig. 6: *(a)-(d) Simple gap primitives contained inside the triangle, and (e) a special case when the power center is outside the triangle.*

on the classification of the types of neighboring triangles. We assume that t_1 is one of t_0's neighbors, and $\mathbf{c}_0, \mathbf{c}_1$ are power centers of t_0 and t_1, respectively. $\mathbf{p}_0\,\mathbf{p}_1$ is the common edge shared by t_0 and t_1, and the direction of $\mathbf{p}_0\,\mathbf{p}_1$ is *ccw* (counter-clockwise) in triangle t_0. Then, there can be the following two cases:

First, if the triangle t_0 contains only the power center $\mathbf{c}_0$ of itself (see Figure 6(a)-(d)), then we can extract the gap primitive directly by traversing its three (ccw) edges using the following rules:

—If $|\mathbf{p}_0\,\mathbf{p}_1| \leq r_0 + r_1$, we compute the intersection points of the disks centered at $\mathbf{p}_0$ and $\mathbf{p}_1$ and the intersection point on the left side of $\mathbf{p}_0\,\mathbf{p}_1$ is added to the gap primitive of t_0.

—If $|\mathbf{p}_0\,\mathbf{p}_1| > r_0 + r_1$, we compute the intersection points of the disks and edge $\mathbf{p}_0\,\mathbf{p}_1$ and the resulting points $\mathbf{p}_{01}$ and $\mathbf{p}_{10}$ are added to the gap primitive of t_0.

Second, if the triangle t_0's power center is outside t_0, e.g., if it lies in its neighboring triangle t_1, then all these triangles belong to the same connected gap component according to the connectivity rules. Note that in complex cases, t_0's power center can also lie in its neighbor's neighboring triangle. This case is equivalent to that of a triangle containing more than one power center. To extract gap primitives is equivalent to decomposing the gap w.r.t. each gap triangle. We handle this configuration as follows:

—For the edges we currently traversed, if the power centers of two incident triangles lie on two different sides, we compute the intersection points as before.

—For the edges with two power centers of two incident triangles lying on the same side, e.g., edge $\mathbf{p}_0\,\mathbf{p}_1$ shown in Figure 6(e), we simply connect t_0's power center $\mathbf{c}_0$ and $\mathbf{p}_0, \mathbf{p}_1$, which results in two intersection points A and B. The segment AB is then added to the primitive polygon of t_0. Similarly, if the neighboring triangle's power center lies on the same side of of the common edge of the current triangle, e.g., t_1 in Figure 6(e), we also add the segment BA to t_1's gap primitive.

The above extraction process decomposes a connected gap component into non-overlapping simple polygons. Since the time complexity of extracting each gap is constant (each gap primitive has up to six edges), the total time complexity of the gap extraction algorithm is $O(n_t)$, where n_t is the number of triangles. However, given an initial sampling generated by dart-throwing, the number of gap triangles is much less than the total number of triangles. We provide more details in Appendix A to show the validness of this process.

Boundary handling. We also consider bounded domains. The boundary can be a simple polygon with or without holes. In these cases, we first compute the clipped power diagram using the technique presented in [Yan et al. 2013]. The gaps that touch the domain boundary are simply computed by clipping the power cells that contain parts of gaps with the surrounding disks. Figure 11 shows a result of sampling/meshing a 2D polygon.

3.4 Gap primitive updates

Appropriate algorithms exist to construct regular triangulations [Edelsbrunner and Shah 1996]. The points can be dynamically inserted and deleted from a regular triangulation. Inserting points requires $O(n^2)$ in the worst case and $O(\log(n))$ in the average case [Edelsbrunner and Shah 1996], where n is the total number of points. A point can be deleted in constant time [Devillers and Teillaud 2006]. Deleting k points requires $O(k)$ time.

If all points move in a coherent fashion, the regular triangulation might be updated efficiently by existing algorithms [Vigo et al. 2002], but in our implementation, we simply rebuild the data structures if all points move at once. Movement of individual points is handled by deletion and insertion.

Changing the sampling radius of all disks causes a change in the combinatorial structure and we also have to rebuild the regular triangulation. However, if the sampling radius is constant, the regular triangulation is equivalent to the Delauney triangulation and remains unchanged in this case.

4. GAP COMPUTATION ON SURFACES

In this section, we generalize the gap analysis and computation framework to mesh surfaces. Suppose that the input domain $\Omega = \{f_j\}_{j=1}^{m}$ is a triangle mesh surface (where f_j refers to a triangle) and $\mathcal{P}_w$ is a set of weighted samples lying on the surface. Recall that in Section 2.2, we defined the restricted power diagram on surfaces. In this case, each restricted power cell is the intersection of the 3D power cell and the mesh surface, i.e., $\Omega_i \bigcap \Omega = \{\bigcup\{\Omega_i \bigcap f_j\}, \forall f_j \in \Omega\}$. Figure 7 shows an example of the $\mathcal{RPD}$ of a torus. There are three types of vertices in the $\mathcal{RPD}$ of a mesh surface, called restricted power vertices, i.e., A) the original vertices of the mesh, B) the intersection of a mesh edge and a bi-sector power plane, and C) the intersection of a power edge and a mesh triangle (Figure 7(right)). Thm. 2 can be adapted as follows:

THEOREM 3. *A gap of a disk set $\mathcal{P}_w$ exists on a mesh surface iff there exists a restricted power cell of a sample $\mathbf{p}_i$, whose restricted power vertices are not fully covered by the respective sphere $(\mathbf{p}_i, r_i)$ centered at $\mathbf{p}_i$ with radius r_i.*

In the above definition, a type C restricted power vertex is also called a *restricted power center*. Note that each restricted power center is dual to a triangle of the so-called restricted regular triangulation ($\mathcal{RRT}$). The triangles of the $\mathcal{RRT}$ do not lie on the surface, only the vertices do. Since the restricted triangles no longer lie on the surface, we cannot use the 2D per-triangle gap computation (Section 3) for meshes. Alternatively, based on Thm. 3, we present an approach similar to [Jones 2006] to compute the gap primitives on surfaces. The gap primitives are computed by clipping the restricted power cell by the sphere centered at each weighted point. Each restricted power cell can be split into a set of triangles. Then,

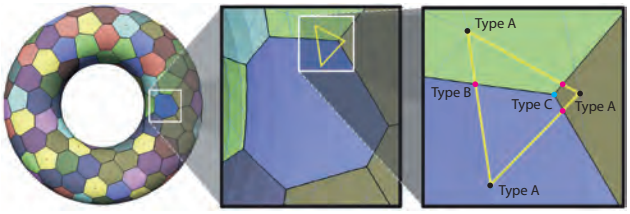


Fig. 7: *Illustration of the restricted power diagram on a surface (left); middle: a restricted power cell; right: a mesh triangle is split into polygons shared by the incident cells.*

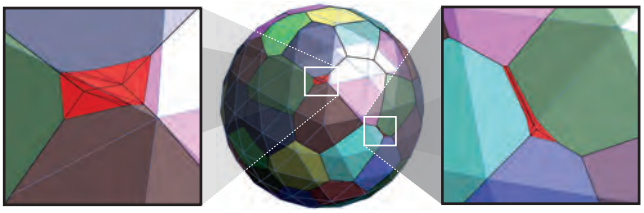


Fig. 8: *Gap computation on surfaces by triangle-sphere clipping. Gaps are shown in red.*

the clipping problem is reduced to a triangle-sphere intersection problem. The clipped regions, i.e., the gap primitives, are approximated by a set of triangles and associated with their incident gaps, as shown in Figure 8. The details of the $\mathcal{RPD}$ computation are analogous to the restricted Voronoi diagram computation in [Yan et al. 2009].

Now we are able to identify and compute gaps on mesh surfaces. However, in the maximal sampling framework (see Section 5.1), we are willing to cluster the independent gap sets as in the 2D counterpart, which can be used for parallel gap filling. A simple region-growing-based approach can be used by detecting the connectivity between clipped gap primitives. Furthermore, if the initial sampling $\mathcal{P}_w$ adapts to the local properties of the mesh surface (e.g., the ϵ-*sampling property* [Amenta and Bern 1999]) and the *topological ball property* [Edelsbrunner and Shah 1997] is met, then the dual restricted regular triangulation is homeomorphic to the input domain Ω. In this case, we are able to define the 3D gap triangles similarly to the 2D plane. A gap triangle is a triangle of the $\mathcal{RRT}$ with at least one vertex whose restricted power cell is not fully covered by the sphere centered at the vertex. Once the gap triangles are detected, we group the neighboring gap triangles/primitives into IGSs for further processing.

5. SURFACE SAMPLING AND REMESHING

In this section, we first describe a framework for adaptive Poisson-disk sampling on surfaces, which also works well in 2D. Then, we present a brief discussion of the link between Poisson-disk sets and surface remeshing. Moveover, we present a high-quality surface remeshing algorithm, as well as a randomized mesh optimization algorithm built on top of the sampling framework, which greatly improves the sampling/meshing quality.

5.1 Adaptive sampling on surfaces

As input, we use a mesh $\Omega = \{f_i\}_{i=1}^m$, a minimal sampling radius r_{min}, a maximal sampling radius r_{max} (default value $16\,r_{min}$), as well as a density function $\rho(\mathbf{x})$ defined on the mesh surface. A voxel grid is built for accelerating the sampling process. We first voxelize the mesh with voxels whose sizes are equal to $\frac{r_{min}}{\sqrt{3}}$. Each voxel records the indices of samples that fully cover it. A voxel is *valid* if it is not fully covered by any sphere. We follow the two-step sampling strategy presented in [Ebeida et al. 2011]: 1) dart throwing with a grid and 2) gap filling.

Initial sampling. In the first step, we perform classic dart throwing on surfaces. Each triangle f is associated with a weight $w_f = \rho(\mathbf{c}_f)|f|$, where $\mathbf{c}_f$ is the barycenter and $|f|$ is the area of the triangle. The cumulative probability density function (cpdf) of the weights is stored in a flat array. Each time, a dart is generated by first selecting a triangle from the cpdf and then a point $\mathbf{p}$ is randomly generated in the triangle, as well as a radius $r = \min(r_{max}, \max(r_{min}, \frac{1}{\sqrt{\rho(\mathbf{p})}}))$ associated with the point. The new dart is tested against the grid. The dart is accepted if it is not contained by previous samples and does not contain any other existing samples. The index of the new sample is recorded by the voxels that are fully contained by this sample. The first step is terminated when k consecutive rejections are observed ($k = 300$ in our implementation).

Gap filling. The second step is an iterative algorithm. At each iteration, all gap triangles are first grouped into IGSs. All gap primitives of each IGS are triangulated and each triangle is associated with a weight as before. For each IGS, we create a cpdf for the triangles belonging to it. Since the sampling of each IGS does not affect another IGS, we perform gap filling in parallel. We use the same method as in the initial sampling stage to generate new disks for each IGS. Note that in the gap filling step, the newly generated disks whose centers are not covered by existing disks may cover the centers of the existing disks. In this case, we recompute the radius of the new disk so that it will not contain any previous samples, i.e., we set the radius of the new disk as the distance to the nearest sample. Here, we choose the maximal conflict metric [Kalantari and Sen 2011] since we assume that the centers of the disks cannot be covered by any other disks. This solution is reasonable for a smooth density function. Actually, our gap computation framework can also handle the cases when the disk centers are covered by other disks. This will be further explored in future work. While there are multiple applications for sampling points on surfaces, the most important application is remeshing, which we will describe next.

5.2 Poisson-disk sampling and surface remeshing

Surface remeshing is a broad topic and the best choice of a surface remeshing algorithm depends on the application [Alliez et al. 2008]. Poisson-disk sampling is mainly useful for remeshing for simulations. In other applications, e.g., architectural panel layouts, the blue noise pattern will often be considered unattractive. Before presenting our solution to adaptive remeshing, we would first like to discuss what criteria are required for good mesh and also establish the link between our disk sampling method and remeshing. This discussion suggests why our work can be more successful than previous surface sampling algorithms at remeshing for simulation.

First, the blue noise property is important because it reduces directional bias in the simulation [Ebeida et al. 2012]. While this fact has been used for uniform remeshing using disk sampling with a global uniform radius, it is also important for adaptive remeshing. Second, the most important geometric characteristic of a triangle mesh is the minimal angle in a triangle or the percentage of triangles with small angles [Shewchuk 2002]. Statistics relating to the minimal angle θ_{min}, as well as $Q(t) = \frac{6}{\sqrt{3}} \frac{|t|}{p(t)\,h(t)}$, (where $|t|$ is

the area of t, $p(t)$ is the half-perimeter of t and $h(t)$ the longest edge length of t [Frey and Borouchaki 1997]) are mentioned in almost all recent remeshing papers. The reason for this is that the minimal angle influences the condition number of the matrices in FEM. Interestingly, there is a theoretical guarantee that the minimal angle is 30 degrees in a uniform 2D maximal sampling [Chew 1989; Ebeida et al. 2011]. It can also be shown that the same conclusion still holds on surfaces. However, in the case of adaptive sampling, we do not have any angle guarantees. We propose a randomized optimization technique that can improve the angle bounds dramatically for adaptive remeshing (see Section 5.3). This is an important link that is often overlooked in previous work: maximal sampling is essential for theoretical guarantees about triangle angles. Maximal sampling is also important in praxis. We verified this by generating 100 point sets with $10k$ samples and then removing the last 20 sampled points. The minimal angle is significantly worse. Only in about 10% of the cases it is still $\geq 30^o$. Third, irregular vertices are undesirable, because they require special handling and often separate code. Therefore, we aim at reducing the different types of irregular vertices to only two types: valence 5 and valence 7 vertices. It would also be helpful to reduce the total number of irregular vertices, but this conflicts with the blue noise property and the minimal angle properties. Fourth, the surface approximation is important. Most commonly, this is measured in the Hausdorff distance or the root mean squared error. Fifth, it is important to use Euclidean distances and not geodesic distances. While sampling geodesic disks seems more complex and sophisticated, this can be counterproductive for remeshing. The edges in a mesh are straight and not geodesic paths on surfaces. A simple implication is that the use of geodesic distances voids all angle guarantees.

5.3 Surface remeshing and optimization

To obtain a mesh, we extract the dual triangulation from the restricted power diagram of the samples, which is a good remeshing of the input surface Ω. There are several potential challenges to surface remeshing that we address here.

Topological validity. In this work, we adapt the sampling radius to the local feature size to ensure the topological validity. Theoretically, the ϵ-sampling theorem [Amenta et al. 1998] requires that for each point $\mathbf{x} \in \Omega$, there exists a sample $\mathbf{p}$, such that $|\mathbf{x}-\mathbf{p}| < 0.3\, lfs(\mathbf{p})$. The $\mathcal{RRT}$ of the samples is then homeomorphic to Ω. However, in practice, even if these theoretical guarantees are not met, our algorithm still generates topologically correct results as shown in Figure 18. We can detect topological inconsistencies using the algorithm presented in [Yan et al. 2009]. Once such a configuration is detected, we discard the samples that cause the problem and resample the gaps. We refer to [Yan et al. 2009] for more details of the mesh extraction algorithm since it is not the main contribution of this paper.

Valence and angle optimization. Uniform blue noise remeshing has many nice properties. Given a constant sampling radius r, the meshes generated from blue noise sampling exhibit the following bounds: an angle bound $[30^o, 120^o]$, an edge length bound $[r, 2\,r]$ and an area bound $[\frac{\sqrt{3}}{4}\, r^2, \frac{3\sqrt{3}}{4}\, r^2]$ as its 2D counterpart. All these properties are desired in many applications, particularly the angle bound, which is crucial to FEM applications [Shewchuk 2002]. However, in the case of adaptive sampling, the above theoretical bounds do not hold. To improve the meshing quality, we introduce a novel and simple randomized optimization algorithm, i.e., angle bound and valence optimization.

This algorithm iteratively removes the sample points with unsatisfactory properties and their neighborhoods and then refills the gaps. In angle bound optimization, the vertices with one triangle angle less than a minimal angle threshold or larger than a maximal threshold are removed. In valence optimization, the vertices whose degrees are less than 5 or larger than 7 are removed. These two optimization criteria can either be performed separately or jointly. The valence/angle optimization terminates when the required criteria are met or the maximal iteration number (25 in our implementation) is reached. In a joint valence and angle optimization, a global optimization is performed interleaving between valence and angle optimization. Typically, it takes 5-10 global iterations to meet both quality requirements. During the optimization, the $\mathcal{RRT}$ and $\mathcal{RPD}$ are locally updated.

Edge length optimization. Besides the valence/angle optimization, we are also able to optimize other geometric properties, such as edge length, using the same framework. Given a user input threshold λ_e, we iteratively remove all the edges $(\mathbf{p}_i, \mathbf{p}_j)$ with $|\mathbf{p}_j - \mathbf{p}_i| > \lambda_e\,(r_i + r_j)$, followed by a gap filling step, until the desired edge length quality is met.

Feature preservation. We also implemented a simple feature-preserving sampling step in our framework. The features are provided by the user, in the form of a 1D curved skeleton. The corners (vertices of the skeleton with more than 2 neighboring edges, or with sharp turning angles) are inserted directly as sample points, and the feature curves are first sampled before the surface sampling. The edge lengths of the feature samples are optimized by edge length optimization ($\lambda_e = \sqrt{3}$) so that the neighboring spheres are deeply intersected [Cheng et al. 2007]. The samples of the feature skeleton remain fixed during the surface sampling/optimization stages.

6. RESULTS AND DISCUSSION

In this section, we present several sampling and meshing results using our framework based on the proposed gap processing techniques. We also compare our results with current state-of-the-art approaches. We use CGAL [cga] for the sequential 2D/3D regular triangulation, and the OpenMP library for parallel gap filling. The experimental results are conducted on an Intel X5680 Dual Core 3.33GHz CPU with 4GB memory and a 64-bit Windows 7 operating system.

6.1 2D sampling and meshing

We first present the results of 2D sampling. Figure 9 shows two examples of uniform sampling and optimization. The first example is a 2D square with a periodic boundary condition, and the second is defined by two concentric circles. The statistics of these results are given in Table I.

We show a comparison to uniform 2D sampling and the corresponding spectral analysis in Figure 10, which demonstrates that our optimization framework preserves the blue noise properties well. We use the PSA software [Schlömer and Deussen 2011] to perform the spectral analysis of the Poisson-disk sets. Table. II lists the statistics of vertex valences of different approaches.

An example of adaptive meshing is given in Figure 11, where we set the desired angle bounds to $[35^o, 105^o]$. For the examples with boundaries, we first perform 1D maximal sampling on the boundary; then, we apply edge length optimization to ensure that the boundary disks are deeply intersected [Cheng et al. 2007] (we

Result	#v	θ_{min}	θ_{max}	$\|e\|'_{min}$	$\|e\|'_{max}$	$\|t\|'_{min}$	$\|t\|'_{max}$	$\theta < 30^o$	v_{567}
Square1	26.8k	21.6	130.3	1.0	1.458	0.947	2.043	3.33	94.2%
Square2	34.5k	30.1	118.6	1.0	0.999	1.001	0.998	0.00	96.4%
Square3	35.1k	35.0	104.9	1.0	0.999	1.001	0.997	0.00	100%
Circles1	19.6k	18.9	131.4	1.0	1.950	1.088	2.950	8.83	93.1%
Circles2	24.9k	30.0	118.5	1.0	0.999	1.003	0.996	0.00	96.2%
Circles3	25.3k	35.0	105.0	1.0	0.999	1.003	0.995	0.00	100%

Table I. : *Statistics of 2D uniform sampling/meshing ($r_{min} = 4.5 \times 10^{-3}$). $|e|'_{min} = \frac{|e|_{min}}{r_{min}}$ is the ratio of the minimal edge length over the theoretical minimal edge length bound, as it is for $|e|'_{max} = \frac{|e|_{max}}{2r_{min}}$, $|t|'_{min} = \frac{|t|_{min}}{\frac{\sqrt{3}}{4} r^2_{min}}$, and $|t|'_{max} = \frac{|t|_{max}}{\frac{3\sqrt{3}}{4} r^2_{min}}$. $\theta < 30^o$ is the percentage of the triangles with θ_{min} smaller than 30 degrees. v_{567} is the percentage of vertices with valence 5, 6, or 7. For each model, we show the meshing quality of 1) classic dart throwing, 2) maximal sampling and 3) optimized sampling. The data shown in the table are averages of 10 runs for each model.*

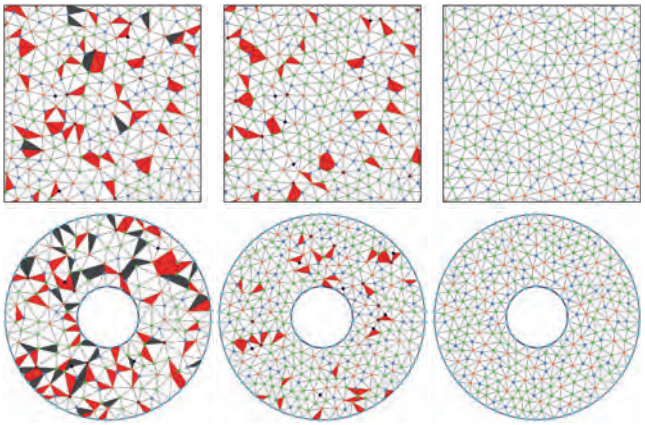


Fig. 9: *Illustrations of uniform 2D sampling/optimization. Top: a unit square with a periodic boundary condition; bottom: two concentric circles. Left column: dart throwing; middle: maximal sampling; right: optimized sampling (angle and valence). Vertices with valence 5: blue, 6: green, 7: orange. Darker points correspond to higher (> 7) or lower (< 5) valences. The triangles with $\theta_{min} < 30^o$ or $\theta_{max} > 120^o$ are shown in dark gray, triangles with $\theta_{min} \in [30^o, 35^o]$ or $\theta_{max} \in [105^o, 120^o]$ are shown in red.*

	$v4$	$v5$	$v6$	$7v$	$v8$	$v9$	$v10$
Lloyd	0	7.42	85.15	7.43	0	0	0
CCVT	0.49	21.63	56.30	20.55	1.02	$4.9\,e^{-5}$	0
MPS	1.23	24.04	50.48	22.06	2.18	$4.7e^{-4}$	$7.5\,e^{-7}$
Ours1	0	23.02	53.96	23.02	0	0	0
Ours2	0	22.01	55.98	22.01	0	0	0

Table II. : *Statistics of the vertex valences of different methods. Each column is the percentage of the vertices of the corresponding valence. MPS stands for any maximal Poisson sampling generated by dart throwing. Ours1 is the result of valence optimization, and Ours2 is the valence and angle optimization.*

set $\lambda_e = \sqrt{3}$ in all our experiments). The boundary samples remain fixed during the later sampling/optimization inside the boundary.

6.2 Comparison of 2D uniform MPS

If the sampling radius is a constant r, then the regular triangulation/power diagram is equivalent to the Delaunay triangulation/Voronoi diagram. In this case, we can use the more efficient implementation of the Delaunay triangulation instead of the regular triangulation. In the following, we compare results from our framework with previous approaches for 2D uniform MPS. We call these approaches Ebeida [Ebeida et al. 2011], White [White et al. 2007], Gamito [Gamito and Maddock 2009], and Jones* [2006]. Here, Jones* stands for an improved version of Jones' method [2006] where we changed the code to follow the framework of Ebeida by using a lazy update instead of continuously updating the data structures after each new sample.

First, we show that our gap primitives capture the geometry of the problem very well. Figure 12 (left) compares the number of gaps and gap primitives computed for a varying number of initial samples in the first stage. For 1M points, we obtain $129.3K$ gaps and $695.3K$ gap primitives compared with $841.7K$ gap primitives in Ebeida and $3.57M$ in Jones*. In Figure 12 (right), we illustrate the advantage of extracting connected components. We compare the number of iterations in the second stage to generate a point set with sampling radius $r = 5 \times 10^{-4}$ in a unit square domain with a periodic boundary (this gives approximately $2.8M$ points for each method). We perform multiple tests and select a representative experiment for each method. Typically our approach only needs 4 or 5 iterations to obtain a maximal point set while others need 10 or more iterations.

Second, we compare the running times of different algorithms. Our running time is slightly faster than the performance reported by White, Gamito, and Ebeida. However, we could only verify White's performance on our machine. One advantage of our method compared with Ebeida is that it relies on Delaunay triangulation, a well-tested method where the code is publicly available, e.g., CGAL [cga] and Qhull [Barber et al. 1996][1]. It takes only $1.91s$ to triangulate one million points obtained in stage one of the algorithm. Another advantage of Delaunay triangulation is that it can nicely track changes to the point set. For example, after an initial Delaunay triangulation is available, newly sampled points can be inserted into Delaunay triangulation using only O(1) instead of O(log(n)), because we know the gap triangle that contains the new sampling point. This tracking ability is also especially useful for the applications we will describe next.

A good extension to dart throwing is to make the algorithm progressive by starting with a large radius and then continually shrinking the radius [McCool and Fiume 1992]. In this way, a progressive randomized triangulation (or point set) is obtained where the final triangulation as well as all intermediate ones has nice spectral properties. We reimplemented [McCool and Fiume 1992] and ex-

[1] `http://www.qhull.org`

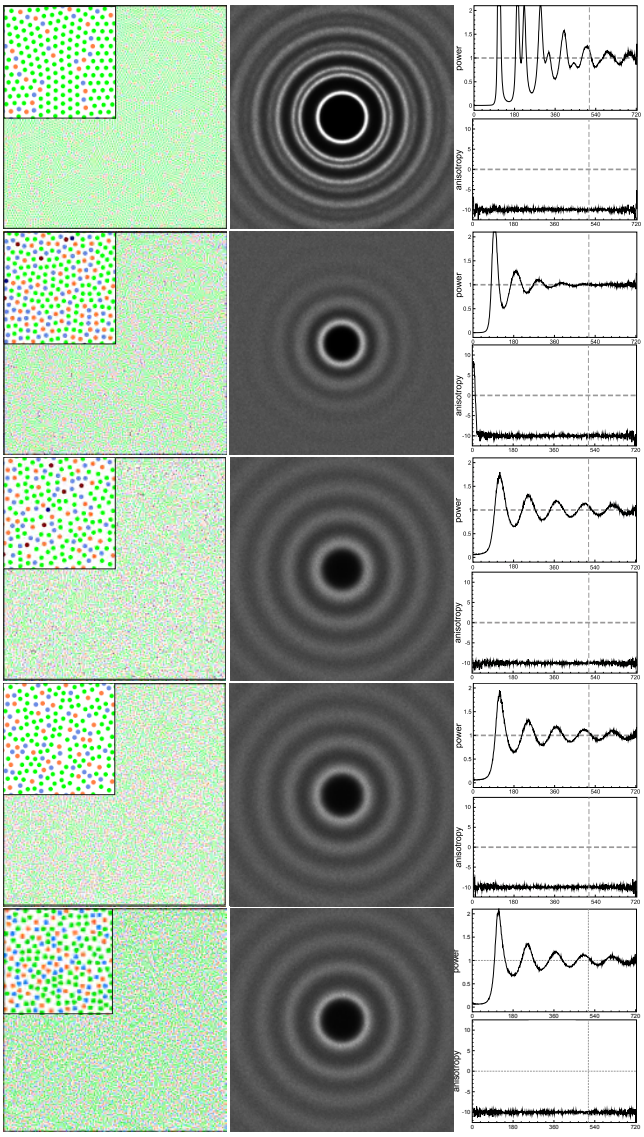


Fig. 10: *Spectral analysis of different approaches (10k samples). From top to bottom: Lloyd iteration [Lloyd 1982]; CCVT [Balzer et al. 2009]; maximal Poisson sampling; valence optimization; valence and angle optimization. The left column is the point set generated by each method, where green dots show vertices with valence 6, red for valence 7, and blue for valence 5. Darker colors signify higher valences.*

perimented with various heuristics to determine when to change the sampling radius to obtain a progressive triangle mesh (Figure 13). In this application, we need the updating operation that changes the sampling radius.

6.3 Surface sampling and remeshing

Uniform sampling/remeshing. We show several experimental results of surface remeshing and optimization. Figure 14 shows the results of uniform sampling on a sphere and a torus. The minimal radius is set to $r_{min} = 4.5 \times 10^{-3}$. In this example, we generate three meshes for each model using non-maximal sam-

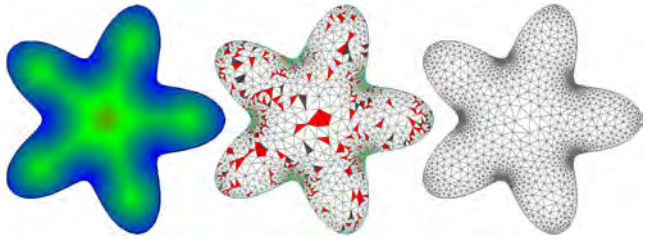


Fig. 11: *Adaptive 2D sampling/meshing. Left: density field; middle: initial sampling. Triangles with $\theta_{min} < 30^o$ and $\theta_{max} > 120^o$ are shown in dark gray and triangles with $\theta_{min} \in [30^o, 35^o]$ and $\theta_{max} \in [105^o, 120^o]$ are shown in red; right: optimized sampling.*

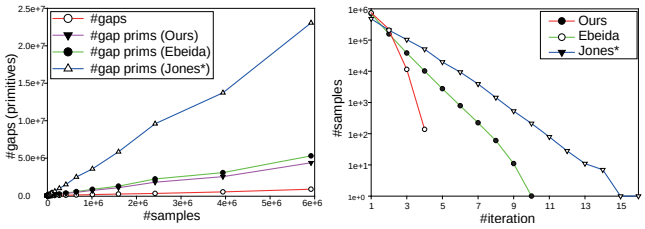


Fig. 12: *The convergence of sampling independent gaps is much faster than sampling all gap primitives together. Left: comparison of the number of gap primitives. Right: our independent gap sampling uses fewer iterations than do Ebeida and Jones*.*

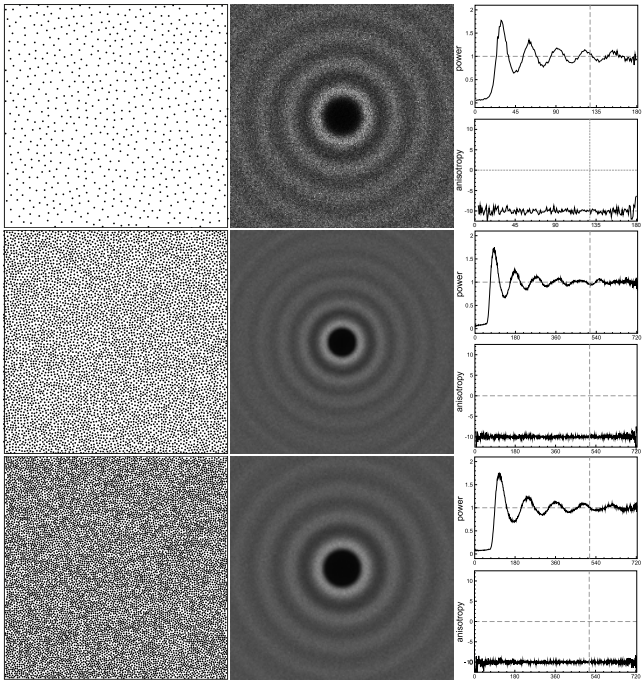


Fig. 13: *Spectral analysis of different levels of hierarchical sampling. Top 680 points, middle 5.9K points and bottom 10K points.*

pling (dart throwing without gap filling), maximal sampling, and optimized sampling. The statistics of the meshes are shown in Table III. We can see that the theoretical bounds do not hold for non-maximal sampling (first experiment), while these bounds are guar-

Result	#v	θ_{min}	θ_{max}	$\|e\|'_{min}$	$\|e\|'_{max}$	$\|t\|'_{min}$	$\|t\|'_{max}$	$\theta < 30^o$	v_{567}
Sphere1	23.5k	22.1	129.3	1.0	1.618	0.993	2.182	1.12	94.5
Sphere2	29.7k	30.2	119.0	1.0	0.998	1.002	0.991	0.00	96.6
Sphere3	30.4k	35.0	105.0	1.0	0.998	1.002	0.996	0.00	100
Torus1	22.4k	21.3	126.8	1.0	1.382	0.979	1.816	0.88	94.6
Torus2	25.1k	30.1	119.1	1.0	0.998	1.004	0.994	0.00	96.3
Torus3	27.4k	35.1	104.9	1.0	0.999	1.003	0.995	0.00	100

Table III. : *Statistics of uniform sampling/remeshing on surfaces ($r_{min} = 4.5 \times 10^{-3}$).*

anteed by maximal sampling (second experiment). For optimized (uniform) sampling, the desired angle bound is set to $[35^o, 105^o]$. This third experiment shows that the geometric properties are greatly improved.

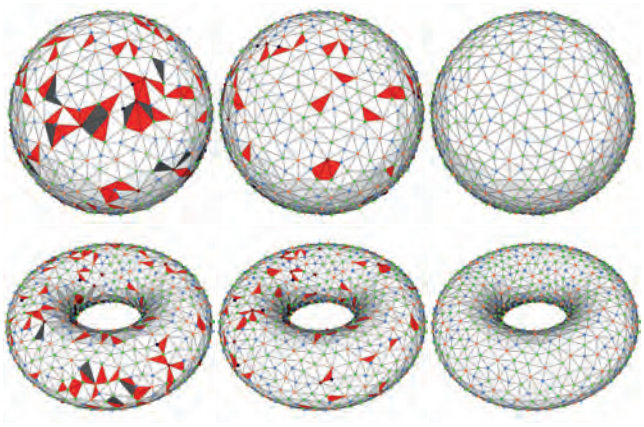


Fig. 14: *Illustration of uniform sampling/optimization on surfaces (we set $r_{min} = 0.03$ for better visualization). Left column: dart throwing; middle: maximal sampling; right: optimized sampling (angle and valence). (Please refer to Figure 11 for the color coding of vertices and triangles.)*

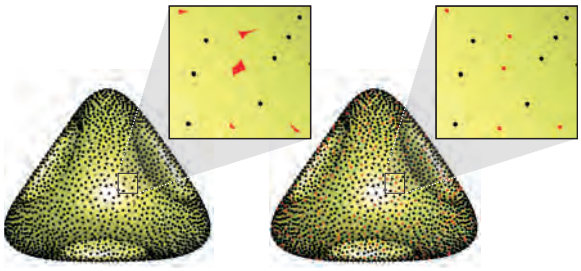


Fig. 15: *Comparison with [Corsini et al. 2012]. Left: uniform sampling result of [Corsini et al. 2012] with 3909 samples ($r \approx 0.012$). Gaps are detected by our technique (red regions); right: maximal sampling (4560 samples) by filling the gaps using our approach, and the new sampled points are shown in red (the model is from the Aim@Shape repository).*

We compare the uniform surface sampling with the most recent approach [Corsini et al. 2012]. To make a fair comparison, we set the over-sampling factor to 200 so that their result has better maximality and the algorithm takes a similar time as our approach for the same sampling radius (both methods take about 0.3s for $r \approx 0.012$). However, as shown in Figure 15(a), we are able to detect the gaps from their output and show that this competing result is not maximal. In another comparison of uniform sampling, the Elk model is remeshed with $31k$ vertices. We set the desired angle bound to $[37^o, 98^o]$ and still obtain a higher quality output, compared with the state of the art [Yan et al. 2009]. The results are summarized in Table IV.

We compare our feature-preserving (uniform) remeshing result with previous work in Figure 16. In this example, we request a minimal angle of 35^o, but we can only obtain 33.9^o. We also cannot get a pure valence $5, 6, 7$ solution because of the feature constraints.

Adaptive remeshing. We applied our adaptive remeshing/optimization to various models. We use the *local feature size* (lfs) [Amenta et al. 1998] as the density function, i.e., $\rho(\mathbf{x}) = \frac{1}{lfs(\mathbf{x})^2}$. Several results are shown in Figure 18 and 17. Another example of adaptive sampling on a mesh surface with boundaries is shown in Figure 19. The boundary is treated the same as a 1D feature curve.

We compare our remeshing algorithm with the state-of-the-art remeshing approaches [Cheng et al. 2007; Valette et al. 2008; Fu and Zhou 2008; Yan et al. 2009], in terms of Q_{min}, θ_{min}, θ_{max}, Hausdorff and RMS distances, the ratio of angles smaller than 30^o and the ratio of the valence 5,6,7 vertices (see Table IV). In Table IV, SAG refers to [Surazhsky et al. 2003], DEL refers to [Cheng et al. 2007], DSR refers to [Fu and Zhou 2008], CVD refers to [Valette et al. 2008], CVT refers to [Yan et al. 2009], CAP refers to [Chen et al. 2012], and MPS refers to our approach without optimization. In adaptive remeshing, the desired angle bound is set to $[32^o, 115^o]$. Similar to the uniform meshing case, we can observe that our approach exhibits better Q_{min} and θ_{min}, as well as high approximation quality to the input surface. For CVT and CAP, we use 100 iterations to generate all the results shown in this paper. Our method has a larger Hausdorff distance compared with CVT-based remeshing [Yan et al. 2009] due to the fact that we did not explicitly optimize the Hausdorff distance, while CVT tends to approximate the input mesh by minimizing the Hausdorff distance [Nivoliers et al. 2012]. We would like to address this issue in future work using our randomized optimization framework.

Efficiency. Figure 20(a) shows the timing curve of our algorithm with an increasing number of sample points, and the convergence of the valence/angle optimization is shown in Figure 20(b) on the Bunny model shown in Figure 17. For this example, our algorithm takes $12.4s$ for the initial sampling, $6.4s$ for gap filling and $4.9s$ for optimization ($23.7s$ in total), while CVT takes $182s$ and CAP takes $391s$ for the same number of samples.

Limitations. There are several limitations to our current approach that can be addressed in future work. 1) Although the sampling framework works well for mesh surfaces, we might fail to compute a valid remeshing if the minimal radius is larger than the local feature size of the surface. For example, we can remesh a thin

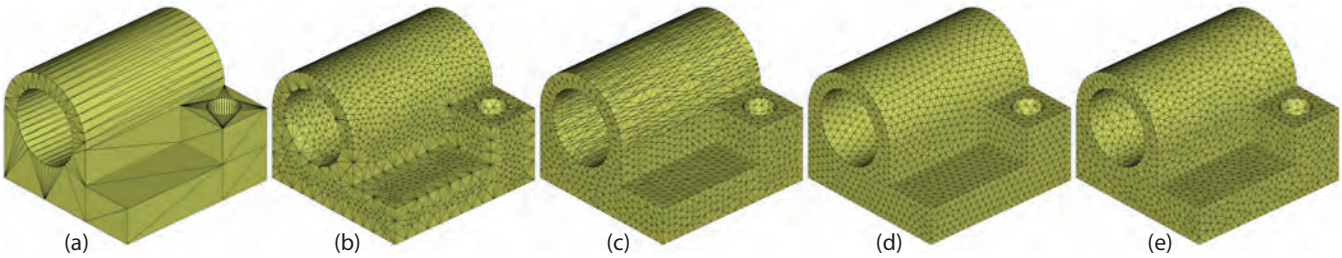


Fig. 16: *Comparison of the remeshed Joint model, 3.2k vertices. (a) Input mesh, (b) DelPSC [Cheng et al. 2007]; (c) CVD [Valette et al. 2008]; (d) CVT [Yan et al. 2009]; (e) ours (the model is from the Aim@Shape repository).*

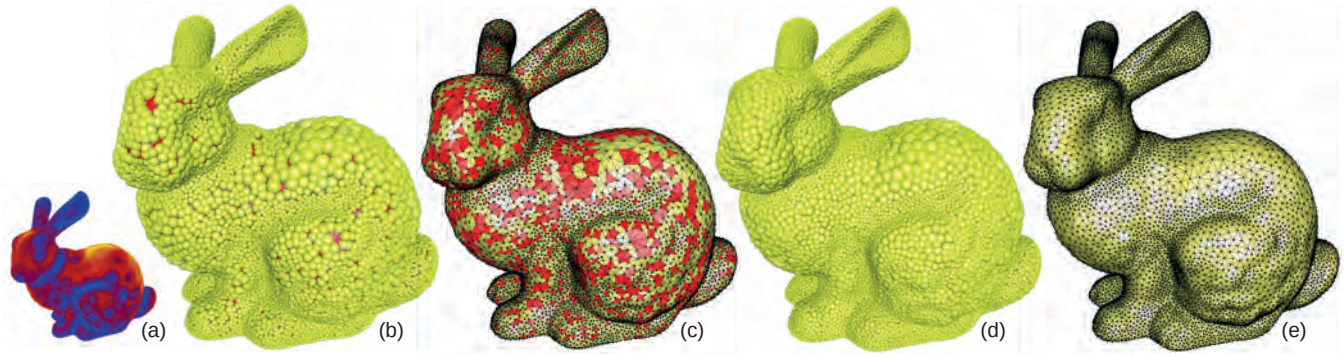


Fig. 17: *Adaptive maximal Poisson-disk sampling on the Bunny model. (a) the density map, where cooler colors correspond to a smaller radius while warmer colors correspond to a larger radius; (b) non-maximal sampling results in gaps (red regions); (c) triangles of the remeshing that are effected by gaps (red triangles); (d) maximal sampling; and (e) remeshing using maximal sampling. Only maximal sampling leads to good triangle shapes that are essential for simulation (the model is from the Stanford 3D scanning repository).*

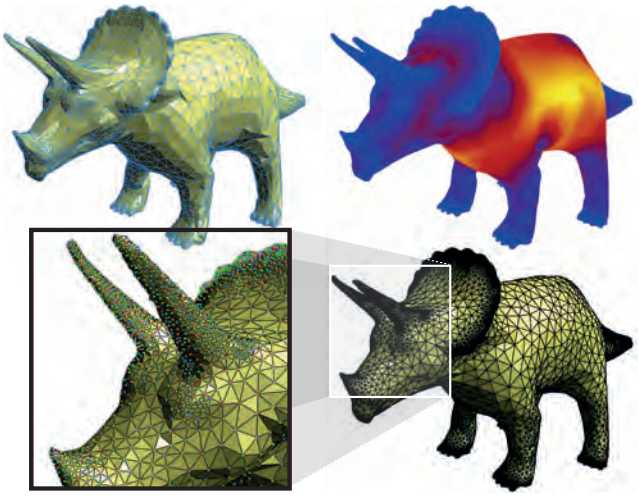


Fig. 18: *Adaptive sampling/remeshing. Top: input mesh and the density function; Bottom: remeshing result (the model is from the Aim@Shape repository).*

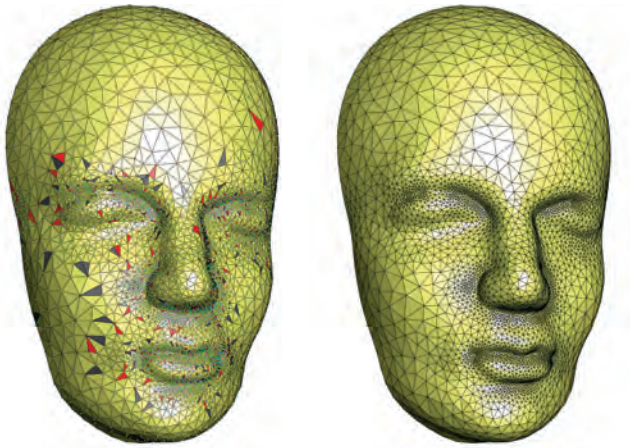


Fig. 19: *Boundary handling, 9k vertices. (a) maximal sampling/remeshing. The triangles with $\theta_{min} < 30^o$ are shown in dark gray and the triangles with $\theta_{min} \in [30^o, 32^o]$ are shown in red. (b) Optimized sampling and remeshing. The optimized angle range is $[32^o, 115^o]$ (the model is from the Aim@Shape repository).*

sheet mesh with parallel planes using small triangles, but we cannot remesh it using large triangles. Another limitation is that we do not have any theoretical guarantee for convergence of the randomized mesh optimization. This is partially due to the fact that in some extreme cases the valence and angle optimization conflict with each other. One possible solution is to enlarge the region further to be resampled. The third limitation is that our framework depends on the regular triangulation/power diagram, which limits the performance of our algorithm compared with GPU-based approaches [Bowers et al. 2010].

Model	#v	Q_{min}	θ_{min}	θ_{max}	Hdist	RMS	$\theta_{<30^o}$	v_{567}	Time(s)
adaptive remeshing									
Rockerarm[CVD]	5.8k	0.353	16.5	133.2	0.44	0.048	0.30	98.1	3.0
Rockerarm[DSR]	5.8k	0.004	0.14	152.9	0.55	0.057	1.34	93.1	35.0
Rockerarm[CVT]	5.8k	0.428	21.3	124.4	**0.34**	**0.032**	0.11	99.7	47.5
Rockerarm[CAP]	5.8k	0.413	23.5	126.4	0.58	0.050	0.12	98.8	59.8
Rockerarm[MPS]	5.3k	0.320	14.0	130.0	0.48	0.034	1.49	90.6	3.2
Rockerarm[OUR]	5.8k	**0.516**	**32.0**	**113.6**	0.48	0.033	**0**	**100**	4.8
Homer[CVD]	7.2k	0.049	2.82	173.2	0.35	0.057	2.03	94.3	3.1
Homer[DSR]	7.5k	0.150	21.7	129.9	0.43	0.068	0.10	95.2	67.0
Homer[CVT]	7.2k	0.334	16.2	120.8	**0.20**	**0.029**	2.04	99.5	74.2
Homer[CAP]	7.2k	0.405	21.0	126.6	0.23	0.046	0.42	96.4	81.9
Homer[MPS]	7.3k	0.371	21.8	131.4	0.32	0.028	0.29	95.4	3.5
Homer[OUR]	7.2k	**0.513**	**32.0**	**115.0**	0.31	**0.023**	**0**	**100**	5.1
Triceratops[CVD]	9k	0.007	0.46	179.1	0.59	0.050	4.37	94.1	4.8
Triceratops[CVT]	9k	0.385	15.5	127.3	**0.24**	**0.038**	0.29	99.2	82.9
Triceratops[CAP]	9k	0.411	21.2	126.4	0.32	0.035	0.43	97.2	116.4
Triceratops[MPS]	9k	0.29	13.5	138.6	0.48	0.062	1.23	92.0	6.3
Triceratops[OUR]	9k	**0.506**	**32.0**	**114.8**	0.46	0.062	**0**	**100**	28.4
Elephant[CVD]	11k	0.040	1.33	173.2	0.38	0.039	4.15	95.4	7.2
Elephant[CVT]	11k	0.408	21.2	126.6	**0.23**	**0.024**	0.72	94.0	102.5
Elephant[CAP]	11k	0.316	18.5	138.1	0.24	**0.024**	0.04	93.7	115.6
Elephant[MPS]	11k	0.301	13.0	130.0	0.46	0.029	1.09	92.6	8.4
Elephant[OUR]	11k	**0.505**	**32.0**	**114.9**	0.38	0.061	**0**	**100**	30.6
Bunny[CVD]	12k	0.15	9.6	160.1	0.34	**0.028**	0.71	96.0	5.2
Bunny[CVT]	12k	0.36	17.8	133.2	**0.20**	0.029	0.38	96.4	176.3
Bunny[CAP]	12k	0.20	7.48	137.8	0.48	0.038	4.99	97.5	185.0
Bunny[MPS]	9.8k	0.36	19.2	133.2	0.46	0.042	0.90	94.7	18.8
Bunny[OUR]	12k	**0.51**	**32.0**	**114.6**	0.37	0.035	**0**	**100**	23.7
uniform remeshing									
Joint[CVD]	3.2k	0.040	2.4	174.6	**0.12**	**0.011**	15.6	96.0	2.5
Joint[DEL]	3.2k	0.057	2.83	171.5	0.38	0.031	2.6	91.1	8.9
Joint[CVT]	3.2k	0.555	29.6	108.4	0.26	0.056	0.005	99.2	23.7
Joint[OUR]	3.2k	**0.688**	**33.9**	**104.9**	0.37	0.056	**0**	**99.4**	4.5
Bunny[CVD]	12k	0.111	3.81	146.2	0.43	0.037	17.2	97.6	4.3
Bunny[CVT]	12k	0.618	37.4	101.2	**0.20**	**0.029**	**0**	**100**	171.9
Bunny[CAP]	12k	0.215	20.5	126.0	0.35	**0.029**	0.41	99.5	181.2
Bunny[MPS]	12k	0.480	30.3	118.0	0.44	0.037	0	96.5	10.6
Bunny[OUR]	12k	**0.629**	**38.0**	**100.0**	0.47	0.034	**0**	**100**	56.8
Elk[SAG]	31k	0.092	4.72	166.8	0.30	0.047	0.07	99.7	60.0
Elk[CVD]	31k	0.037	2.28	175.0	**0.20**	**0.012**	1.60	96.5	6.1
Elk[CVT]	31k	0.634	36.5	99.2	0.21	0.014	**0**	99.9	215.3
Elk[CAP]	31k	0.305	11.4	133.4	0.30	0.027	1.93	98.7	242.8
Elk[MPS]	31k	0.475	30.1	118.6	0.22	0.020	**0**	96.4	6.3
Elk[OUR]	31k	**0.645**	**37.0**	**97.9**	0.26	0.020	**0**	**100**	87.5

Table IV. : *Statistics of the remeshing quality compared with previous work. $\#v$ is the number of sampled points. Q_{min} is the minimal triangle quality [Frey and Borouchaki 1997]; θ_{min} and θ_{max} are the minimal and maximal angle; Hdist and RMS are the Hausdorff distance and the root mean square distance between remeshing and the input mesh (% of the diagonal length of the bounding box); v_{567} is the percent of vertices with valences 5,6 and 7. The best result is highlighted with bold font.*

7. CONCLUSION AND FUTURE WORK

We presented a theoretical analysis of gaps in disk sets with varying radii in arbitrary dimensions. Based on this analysis, we proposed algorithms and data structures for gap detection and gap updates when the disk set changes in the 2D plane and on 3D surfaces. The contribution of our work is illustrated with an adaptive remeshing algorithm that can improve the remeshing quality in aspects of minimal angle, vertex valence and triangle quality. In future work, we would like to build a solution for volumetric remeshing that can be applied to static and deformable objects and to extend our gap processing framework to higher dimensions.

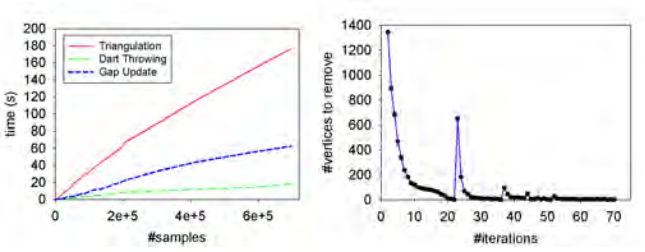


Fig. 20: *Left: Timing curves of our sampling algorithm on the Bunny model. Right: convergence curve of the valence/angle optimization of Figure 17. Each peak of the curve corresponds to a switch between valence/angle optimization.*

ACKNOWLEDGMENTS

We would like to thank the anonymous reviewers for their detailed comments. We thank Helmut Pottmann, Xiaohong Jia and Rachid Ait-Haddou for their valuable discussions, Virginia Unkefer and Sawsan Al-Halawani for proof-reading and all the other GMSV members for their enthusiastic help during this work. We are grateful to Thouis R. Jones, Manuel N. Gamito and David Cline for sharing their code, Tamal K. Dey, Sébastien Vallete and Zhonggui Chen for their software, and Pierre Alliez and Bingfeng Zhou for their results.

REFERENCES

CGAL, Computational Geometry Algorithms Library. http://www.cgal.org.

ALLIEZ, P., UCELLI, G., GOTSMAN, C., AND ATTENE, M. 2008. Recent advances in remeshing of surfaces. In *Shape Analysis and Structuring*. 53–82.

AMENTA, N. AND BERN, M. 1999. Surface reconstruction by Voronoi filtering. *Discrete and Computational Geometry 22*, 481–504.

AMENTA, N., BERN, M., AND KAMVYSSELIS, M. 1998. A new Voronoi-based surface reconstruction algorithm. In *Proc. ACM SIGGRAPH*. 415–421.

ASH, P. F. AND BOLKER, E. D. 1986. Generalized Dirichlet tessellations. *Geometriae Dedicata 20,* 2, 209–243.

AURENHAMMER, F. 1987. Power diagrams: Properties, algorithms and applications. *SIAM J. Comput. 16*, 78–96.

AURENHAMMER, F. 1991. Voronoi diagrams—A survey of a fundamental geometric data structure. *ACM Comput. Surveys 23,* 3, 345–405.

BALZER, M., SCHLÖMER, T., AND DEUSSEN, O. 2009. Capacity-constrained point distributions: A variant of Lloyds method. *ACM Trans. on Graphics (Proc. SIGGRAPH) 28,* 6, 86:1–86:8.

BARBER, C. B., DOBKIN, D., AND HUHDANPAA, H. 1996. The quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software 22,* 4, 469–483.

BOWERS, J., WANG, R., WEI, L.-Y., AND MALETZ, D. 2010. Parallel Poisson disk sampling with spectrum analysis on surfaces. *ACM Trans. on Graphics (Proc. SIGGRAPH Asia) 29,* 6, 166:1–166:10.

BRIDSON, R. 2007. Fast Poisson disk sampling in arbitrary dimensions. In *ACM SIGGRAPH 2007 Sketches*.

CHEN, Z., YUAN, Z., CHOI, Y.-K., LIU, L., AND WANG, W. 2012. Variational blue noise sampling. *IEEE Trans. on Vis. and Comp. Graphics 18,* 10, 1784–1796.

CHENG, S.-W., DEY, T. K., AND LEVINE, J. A. 2007. A practical Delaunay meshing algorithm for a large class of domains. In *16th Intl. Meshing Roundtable*. 477–494.

CHEW, L. P. 1989. Guaranteed-quality triangular meshes. Department of computer science tech report 89-983, Cornell University.

CLINE, D., JESCHKE, S., RAZDAN, A., WHITE, K., AND WONKA, P. 2009. Dart throwing on surfaces. *Computer Graphics Forum (Proc. EGSR) 28,* 4, 1217–1226.

COOK, R. L. 1986. Stochastic sampling in computer graphics. *ACM Trans. on Graphics 5,* 1, 69–78.

CORSINI, M., CIGNONI, P., AND SCOPIGNO, R. 2012. Efficient and flexible sampling with blue noise properties of triangular meshes. *IEEE Trans. on Vis. and Comp. Graphics 18,* 6, 914–924.

DE GOES, F., BREEDEN, K., OSTROMOUKHOV, V., AND DESBRUN, M. 2012. Blue noise through optimal transport. *ACM Trans. on Graphics (Proc. SIGGRAPH Asia) 31*, 171:1–171:12.

DEVILLERS, O. AND TEILLAUD, M. 2006. Perturbations and vertex removal in Delaunay and regular 3D triangulations. Rapport de recherche, INRIA.

DIPPÉ, M. A. Z. AND WOLD, E. H. 1985. Antialiasing through stochastic sampling. *Proc. ACM SIGGRAPH*, 69–78.

DUNBAR, D. AND HUMPHREYS, G. 2006. A spatial data structure for fast Poisson-disk sample generation. *ACM Trans. on Graphics (Proc. SIGGRAPH) 25,* 3, 503–508.

EBEIDA, M. S., MITCHELL, S. A., DAVIDSON, A. A., PATNEY, A., KNUPP, P. M., AND OWENS, J. D. 2011. Efficient and good Delaunay meshes from random points. *Computer-Aided Design 43,* 11, 1506–1515.

EBEIDA, M. S., MITCHELL, S. A., PATNEY, A., DAVIDSON, A. A., AND OWENS, J. D. 2012. A simple algorithm for maximal Poisson-disk sampling in high dimensions. *Computer Graphics Forum (Proc. EUROGRAPHICS) 31,* 2, 785–794.

EBEIDA, M. S., PATNEY, A., MITCHELL, S. A., DAVIDSON, A., KNUPP, P. M., AND OWENS, J. D. 2011. Efficient maximal Poisson-disk sampling. *ACM Trans. on Graphics (Proc. SIGGRAPH) 30,* 4, 49:1–49:12.

EDELSBRUNNER, H. AND SHAH, N. R. 1996. Incremental topological flipping works for regular triangulations. *Algorithmica 15,* 3, 223–241.

EDELSBRUNNER, H. AND SHAH, N. R. 1997. Triangulating topological spaces. *Int. J. Comput. Geometry Appl. 7,* 4, 365–378.

FATTAL, R. 2011. Blue-noise point sampling using kernel density model. *ACM Trans. on Graphics (Proc. SIGGRAPH) 28,* 3, 48:1–48:10.

FREY, P. AND BOROUCHAKI, H. 1997. Surface mesh evaluation. In *6th Intl. Meshing Roundtable*. 363–374.

FU, Y. AND ZHOU, B. 2008. Direct sampling on surfaces for high quality remeshing. In *ACM Symposium on Solid and Physical Modeling (SPM)*. 115–124.

GAMITO, M. N. AND MADDOCK, S. C. 2009. Accurate multidimensional Poisson-disk sampling. *ACM Trans. on Graphics 29,* 1, 8:1–8:19.

JONES, T. R. 2006. Efficient generation of Poisson-disk sampling patterns. *Journal of Graphics Tools 11,* 2, 27–36.

KALANTARI, N. K. AND SEN, P. 2011. Efficient computation of blue noise point sets through importance sampling. *Computer Graphics Forum (Proc. EGSR) 30,* 4, 1215–1221.

KOPF, J., COHEN-OR, D., DEUSSEN, O., AND LISCHINSKI, D. 2006. Recursive Wang tiles for real-time blue noise. *ACM Trans. on Graphics (Proc. SIGGRAPH) 11,* 2, 509–518.

LAGAE, A. AND DUTRÉ, P. 2008. A comparison of methods for generating Poisson disk distributions. *Computer Graphics Forum 27,* 1, 114–129.

LLOYD, S. 1982. Least squares quantization in PCM. *IEEE Transactions on Information Theory 28,* 2, 129–137.

MCCOOL, M. AND FIUME, E. 1992. Hierarchical Poisson disk sampling distributions. In *Graphics Interface*. 94–105.

MITCHELL, D. P. 1987. Generating antialiased images at low sampling densities. In *Proc. ACM SIGGRAPH*. 65–72.

MITCHELL, D. P. 1991. Spectrally optimal sampling for distribution ray tracing. In *Proc. ACM SIGGRAPH*. 157–164.

MITCHELL, S. A., RAND, A., EBEIDA, M. S., AND BAJAJ, C. L. 2012. Variable radii Poisson disk sampling. In *24th Canadian Conference on Computational Geometry (CCCG)*. 185–190.

NIVOLIERS, V., YAN, D.-M., AND LÉVY, B. 2012. Fitting polynomial surfaces to triangular meshes with Voronoi squared distance minimization. In *20th International Meshing Roundtable–IMR2011*. 601–618.

OSTROMOUKHOV, V., DONOHUE, C., AND JODOIN, P.-M. 2004. Fast hierarchical importance sampling with blue noise properties. *ACM Trans. on Graphics (Proc. SIGGRAPH) 23,* 3, 488–495.

SCHECHTER, H. AND BRIDSON, R. 2012. Ghost SPH for animating water. *ACM Trans. on Graphics (Proc. SIGGRAPH) 31,* 4, 61:1–61:8.

SCHLÖMER, T. AND DEUSSEN, O. 2011. Accurate spectral analysis of two-dimensional point sets. *Journal of Graphics, GPU, and Game Tools 15,* 3, 152–160.

SCHLÖMER, T., HECK, D., AND DEUSSEN, O. 2011. Farthest-point optimized point sets with maximized minimum distance. In *High Performance Graphics Proceedings*. 135–142.

SHEWCHUK, J. R. 2002. What is a good linear element? interpolation, conditioning, and quality measures. In *11th Intl. Meshing Roundtable*. 115–126.

SURAZHSKY, V., ALLIEZ, P., AND GOTSMAN, C. 2003. Isotropic remeshing of surfaces: a local parameterization approach. In *12th Intl. Meshing Roundtable*. 204–231.

TURK, G. 1993. Generating random points in triangles. In *Graphics Gems*. 24–28.

VALETTE, S., CHASSERY, J.-M., AND PROST, R. 2008. Generic remeshing of 3D triangular meshes with metric-dependent discrete Voronoi diagrams. *IEEE Trans. on Vis. and Comp. Graphics 14,* 2, 369–381.

VIGO, M., PLA, N., AND COTRINA, J. 2002. Regular triangulations of dynamic sets of points. *Comp. Aided Geom. Design 19,* 2, 127 – 149.

WEI, L.-Y. 2008. Parallel Poisson disk sampling. *ACM Trans. on Graphics (Proc. SIGGRAPH) 27,* 3, 20:1–20:9.

WEI, L.-Y. 2010. Multi-class blue noise sampling. *ACM Trans. on Graphics (Proc. SIGGRAPH) 29,* 4, 79:1–79:8.

WEI, L.-Y. AND WANG, R. 2011. Differential domain analysis for non-uniform sampling. *ACM Trans. on Graphics (Proc. SIGGRAPH) 30,* 4, 50:1–50:8.

WHITE, K. B., CLINE, D., AND EGBERT, P. K. 2007. Poisson disk point sets by hierarchical dart throwing. In *Proceedings of the IEEE Symposium on Interactive Ray Tracing*. 129–132.

XU, Y., HU, R., GOTSMAN, C., AND LIU, L. 2012. Blue noise sampling of surfaces. *Computers & Graphics 36,* 4, 232–240.

YAN, D.-M., LÉVY, B., LIU, Y., SUN, F., AND WANG, W. 2009. Isotropic remeshing with fast and exact computation of restricted Voronoi diagram. *Computer Graphics Forum 28,* 5, 1445–1454.

YAN, D.-M., WANG, W., LÉVY, B., AND LIU, Y. 2013. Efficient computation of clipped Voronoi diagram for mesh generation. *Computer-Aided Design 45,* 4, 843–852.

APPENDIX

A. CORRECTNESS OF GAP DECOMPOSITION

In this appendix, we prove that the primitive extraction algorithm presented in Section 3.3 gives a valid decomposition of a gap.

A.1 Properties of gaps

Recall that a connected gap component is incident to a set of gap triangles. The gap triangles of a gap can be clustered together using the first three connectivity rules (Section 3.2). In the following discussion, when we refer to a triangle, we mean that the triangle is a gap triangle that is incident to the same gap.

The edges of gap triangles can be classified into two types, boundary edges (solid black) and inner edges (blue), as illustrated in Figure 21(a). A boundary edge has an edge length smaller than the sum of the radii of its two endpoints, i.e., $|\mathbf{p}_i\,\mathbf{p}_j| < r_i + r_j$, which means that the edge is covered by the two disks. The boundary edges isolate the gap from other gaps. Each inner edge is shared by two neighboring gap triangles. The inner edges can be further classified into two types: (a) inner edges with two power centers of neighboring gap triangles that lie on different sides of the edge, e.g., edge $\mathbf{p}_3\,\mathbf{p}_5$ in Figure 21(b); (b) inner edges with two power centers of neighboring gap triangles that lie on the same sides of the edge, e.g., edges $\mathbf{p}_1\,\mathbf{p}_7$, $\mathbf{p}_1\,\mathbf{p}_6$,$\mathbf{p}_1\,\mathbf{p}_3$ and $\mathbf{p}_3\,\mathbf{p}_6$.

A gap is bounded by a set of circular arcs. The vertices of the arcs are the intersection points of the disks (yellow points in Figure 21). Each vertex of the arcs is incident to a boundary edge of the regular triangulation. Note that no arc intersects with other arcs except with two neighboring arcs at the arc vertices (otherwise the gap is not connected). The arcs are simply connected. Without loss of generality, we assume the ccw orientation, such that the gap is on the left side when we traverse the arcs, see Figure 21(a) for an example.

If we replace each arc with a segment, then the resulting polygon covers the gap. The polygon is called a *gap polygon*, as shown in Figure 21(b). As discussed in Section 2, the gap is defined as the difference between the domain and the disks, i.e., $\Omega - \cup_i\{(\mathbf{p}_i, r_i)\} = \cup\{\Omega_i\} - \cup_i\{(\mathbf{p}_i, r_i)\}$, which is equivalent to the union of each power cell minus its incident disk, i.e., $\cup_i\{\Omega_i - (\mathbf{p}_i, r_i)\}$. Each power cell contains exactly one arc of the gap boundary. The gap polygon can be decomposed by the power cells (intersecting the gap polygon with the power cell of each disk), as shown in Figure 21(c). We call each such (colored) polygon a *gap cell* of this disk.

A.2 Proof of the validness

We prove the correctness of our algorithm by constructing a valid decomposition of the gap polygon and show that our algorithm is equivalent to this valid decomposition. We define $\deg(\mathbf{p}_i)$ as the number of incident triangles of each disk. Hence, each disk is incident to two boundary edges and $\deg(\mathbf{p}_i) - 1$ inner edges. We first split the gap cell of each disk into $\deg(\mathbf{p}_i)$ sub-cells by inserting $\deg(\mathbf{p}_i) - 1$ auxiliary vertices on each arc, such that each sub-cell is associated with an incident triangle of the disk. For each type (a) inner edge of a disk, we simply split the arc by the edge, and we also split the power edge by the edge, e.g., edge $\mathbf{p}_5\mathbf{p}_3$; For each type (b) inner edge of a disk, we split the arc by connecting the disk center and the power vertex incident to the edge.

After this splitting process, each gap cell is split into $\deg(\mathbf{p}_i)$ sub-cells, each sub-cell is a convex polygon and the union all the sub-cells covers the gap, as shown in Figure 21(d). Until now, all the sub-cells are associated with disks. Next, we reassign the sub-cells to triangles. Since each gap cell is split into $\deg(\mathbf{p}_i)$ sub-cells, we associate each sub-cell to an incident triangle of the disk. After the regrouping, each triangle has three sub-cells, and each pair of sub-cells shares a common power edge, as shown in Figure 21(e).

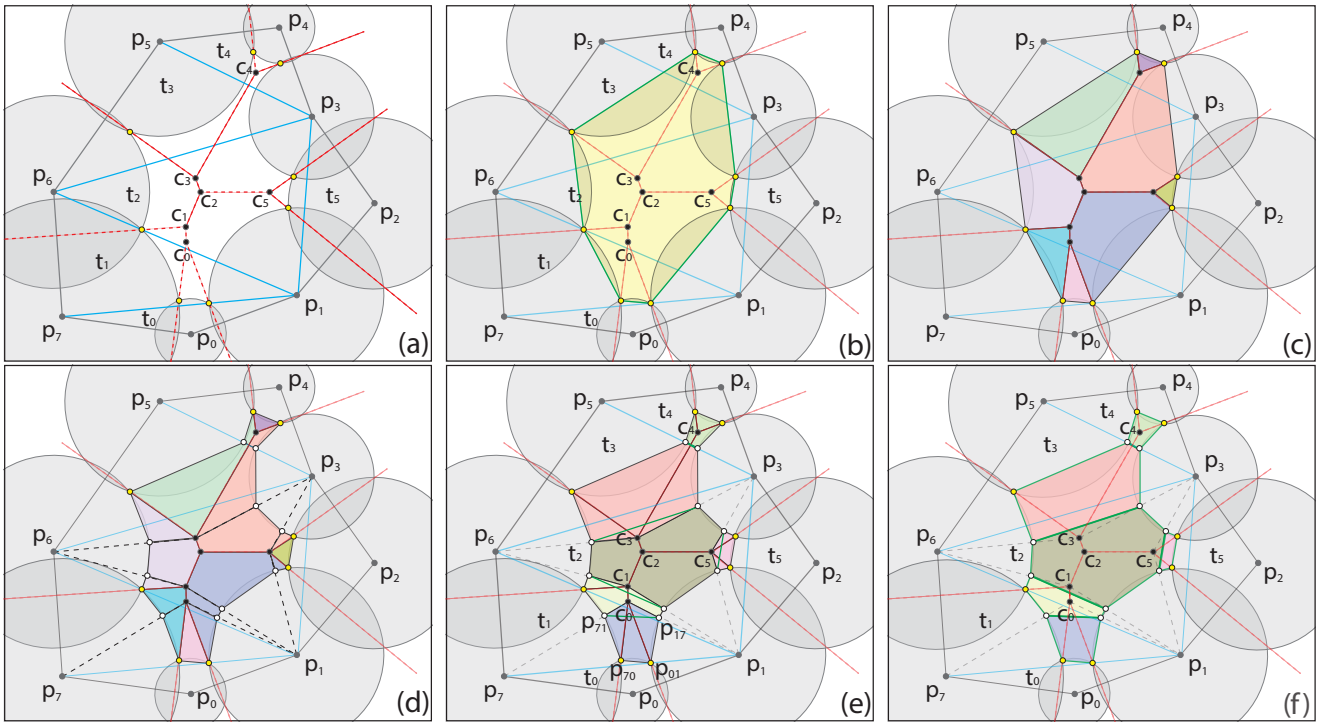


Fig. 21: *An illustration our gap decomposition algorithm. (a): power diagram (red) and regular triangulation (solid black and blue) of a connected gap component. Five power centers fall into the same triangle t_2; (b) initial gap polygon; (c) the gap polygon is subdivided into several gap cells; (d) each gap cell is split into $deg(\mathbf{p}_i)$ sub-cells; (e) the sub-cells are regrouped to from tri-cells for each triangle; (f) the tri-cells are split and merged by auxiliary splitting edges.*

We call the union of the three sub-cells associated with a triangle a *tri-cell*. The tri-cell of a triangle has the following properties:

—Each tri-cell is a simple polygon in ccw orientation.

—The vertices on the arcs of a tri-cell are convex in ccw orientation (white and yellow vertices).

—The tri-cell is convex if the triangle has one or no type (b) inner edge.

—A power center of a tri-cell contributes to a convex vertex to its own triangle and contributes to a concave vertex to its neighboring triangle (Figure 21(e)).

There are two types of vertices on the arc, the intersection point of two neighboring arcs (yellow) and the auxiliary vertices (white) by intersecting the disk center to its power vertices. An auxiliary vertex is convex since it is from a convex sub-cell by our splitting process. To see that the yellow vertices are convex, we use the tri-cell of t_0 as an illustration, as shown in Figure 21(e). There are two arc-arc vertices $\mathbf{p}_{70}$ and $\mathbf{p}_{01}$ and two auxiliary vertices $\mathbf{p}_{71}$ and $\mathbf{p}_{17}$. Let us look at the quad $\mathbf{c}_0\mathbf{p}_7\mathbf{p}_0\mathbf{p}_7$, since $\mathbf{c}_0$ is the power center of triangle $\mathbf{p}_7\mathbf{p}_0\mathbf{p}_1$, which lies outside of the triangle. Hence, $\mathbf{c}_0$ must lie on the left side of edge $\mathbf{p}_{70}\mathbf{p}_{01}$. Since $\mathbf{p}_{17}$ lies on the left side of the power edge $\mathbf{c}_0\mathbf{p}_{01}$, and similarly $\mathbf{p}_{71}$ lies on the right side of the power edge $\mathbf{c}_0\mathbf{p}_{70}$, and $\mathbf{c}_0$ lies on the right side of $\mathbf{p}_{17}\mathbf{p}_{71}$, we can see that the tri-cell $\mathbf{c}_0\mathbf{p}_{71}\mathbf{p}_{70}\mathbf{p}_{01}\mathbf{p}_{17}$ is a convex polygon. Hence the arc-arc vertices are convex. The convexity of the other arc-arc vertices can be explained in a similar way.

Now, we add our auxiliary edges by connecting the pair of auxiliary vertices incident to each inner edge. The auxiliary edge for type (a) inner edges is just the edge itself. We restrict our discussion to type (b) inner edges only. It is obvious that the auxiliary edges in a convex region do not intersect other edges in other convex regions, e.g., the union of sub-cells of t_0, t_3, t_5 cannot intersect with each other. These auxiliary edges split the convex union of sub-cells into two convex parts, i.e., a triangle and the remaining part, which is still a convex polygon. We reassign the triangle part to its neighboring cell, which fills the concave part. For triangle t_2, which has one concave and one convex power center, once the concave part is filled by its neighboring cell, it becomes convex. The auxiliary edge of t_2 now lies in a convex region. It cannot intersect with any other auxiliary edges. Similarly, we reassign the regions using the auxiliary edge of t_2. Figure 21(f) shows the final result of the gap decomposition, which is equivalent to the output of the algorithm described in Section 3.3.

To this end, we have shown that the splitting edges do not intersect with each other, and we can conclude that our decomposition algorithm fulfills the following properties:

—Each gap triangle is associated with a simple convex polygon with up to six edges (three edges from three arcs and the edges or vertices connecting the arcs).

—All these polygons are non-intersecting except along the common edges (auxiliary splitting edge).

—The union of these polygons covers the whole gap.

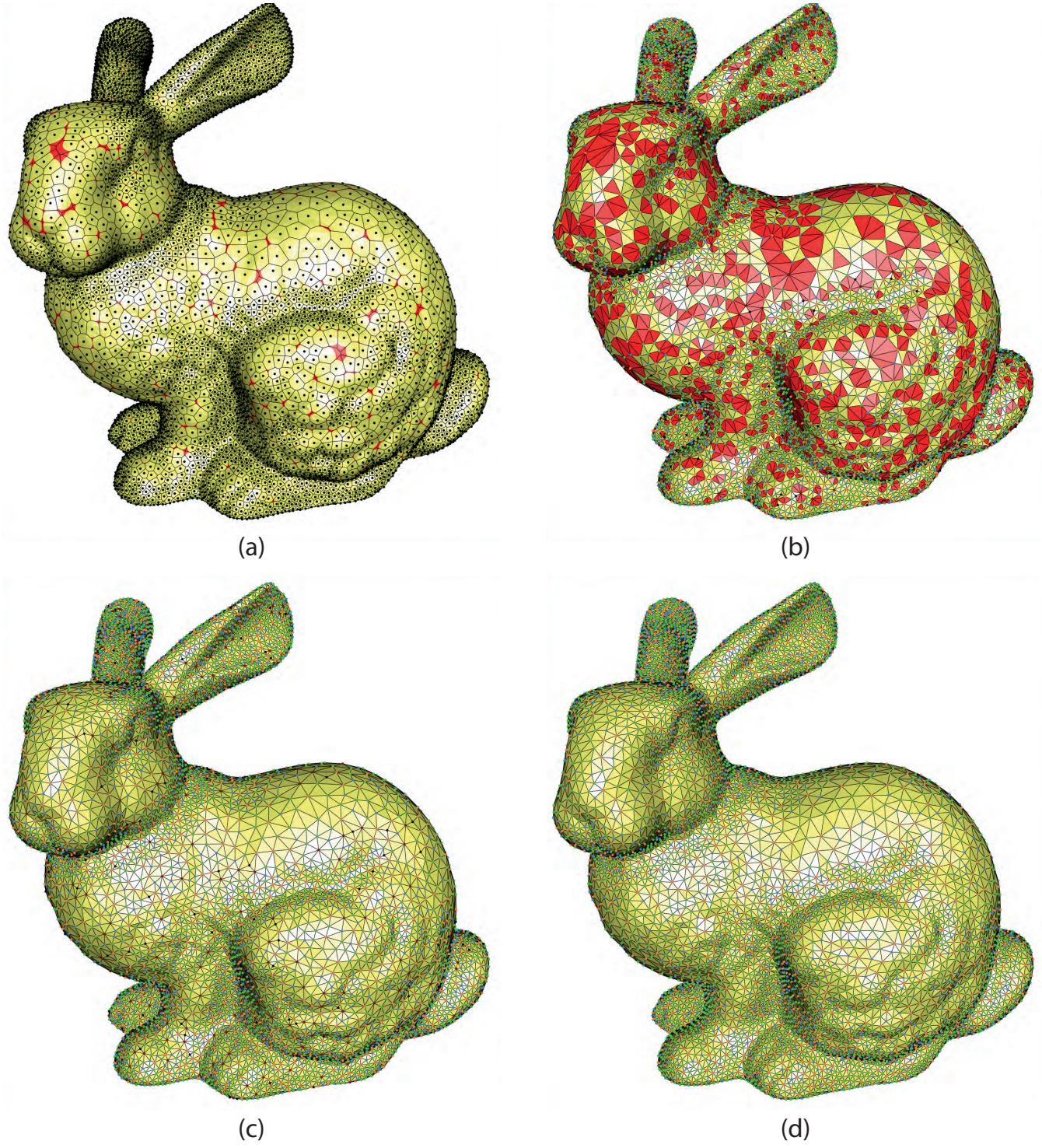


Fig. 1: *Illustration of gap computation and valence optimization of the Bunny model. (a) The RPD and gaps on the surface, (b) remeshed triangles correspond to the non-maximal samples (red), (c) maximal sampling without optimization, (d) optimized remeshing. (The vertices with valence 5 (v5) are shown in blue, v6 green and v7 orange. Vertices with other valences are shown in dark.*

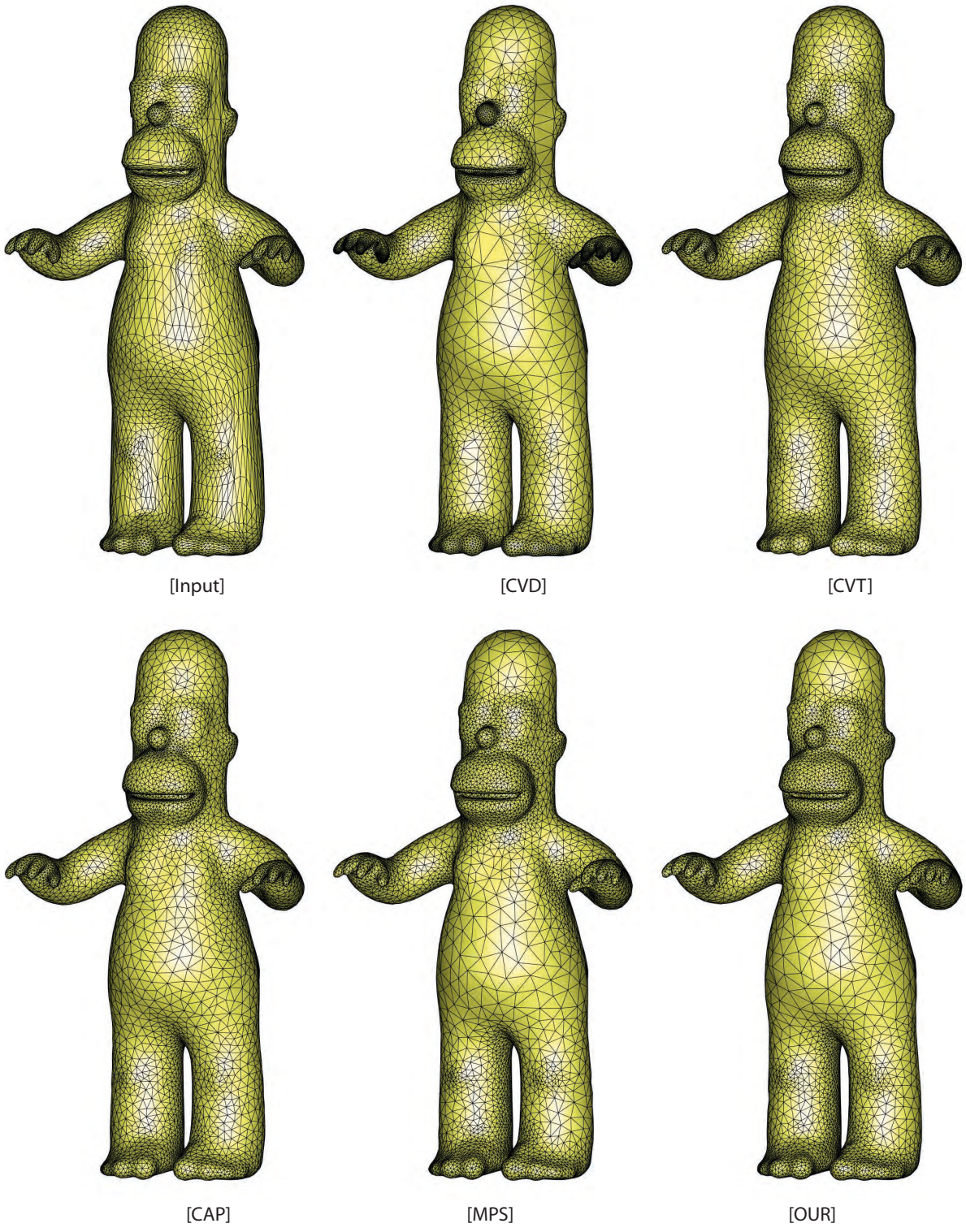


Fig. 2: *Remeshing results of the Homer model.*

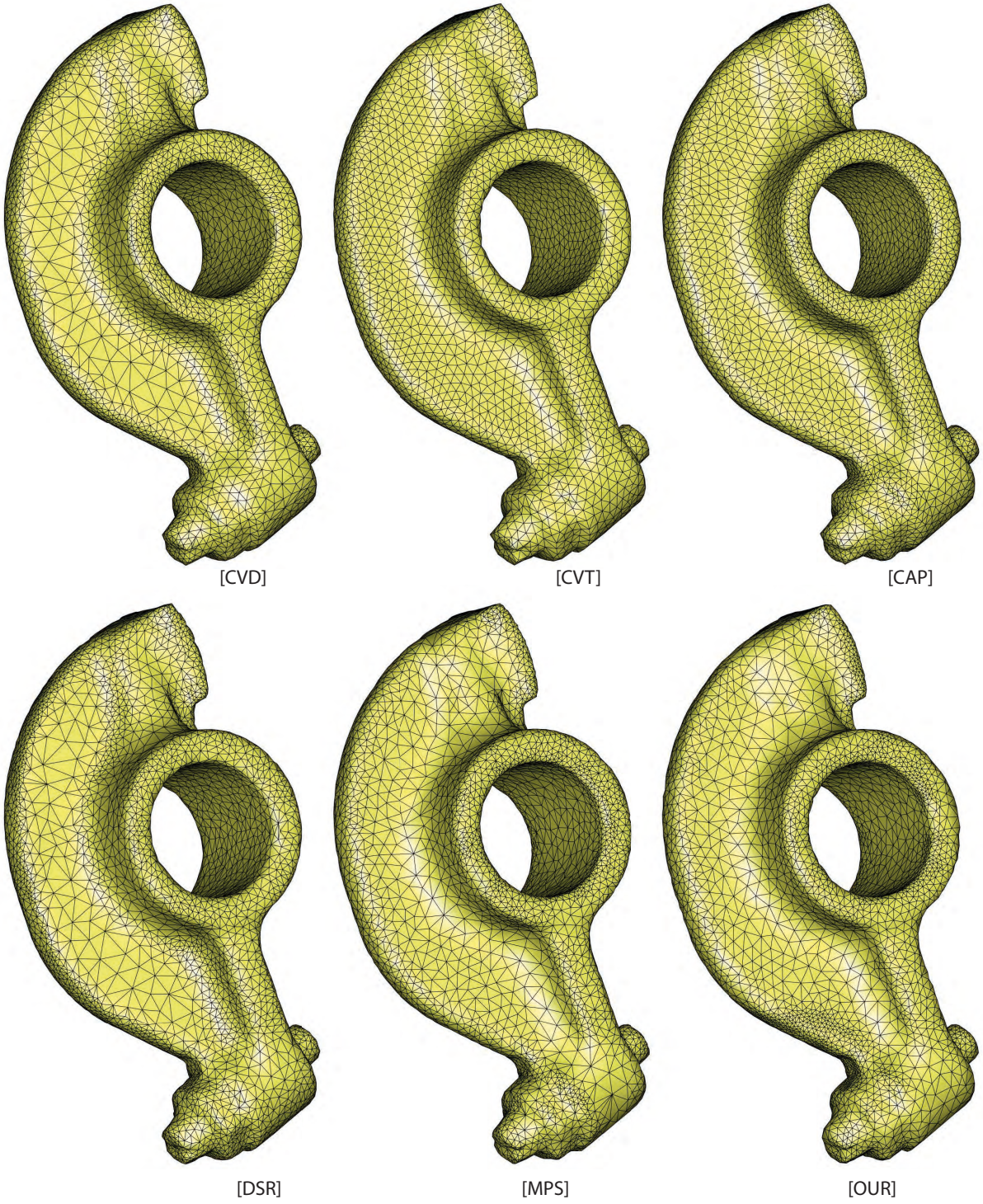


Fig. 3: *Remeshing results of the RockerArm model.*

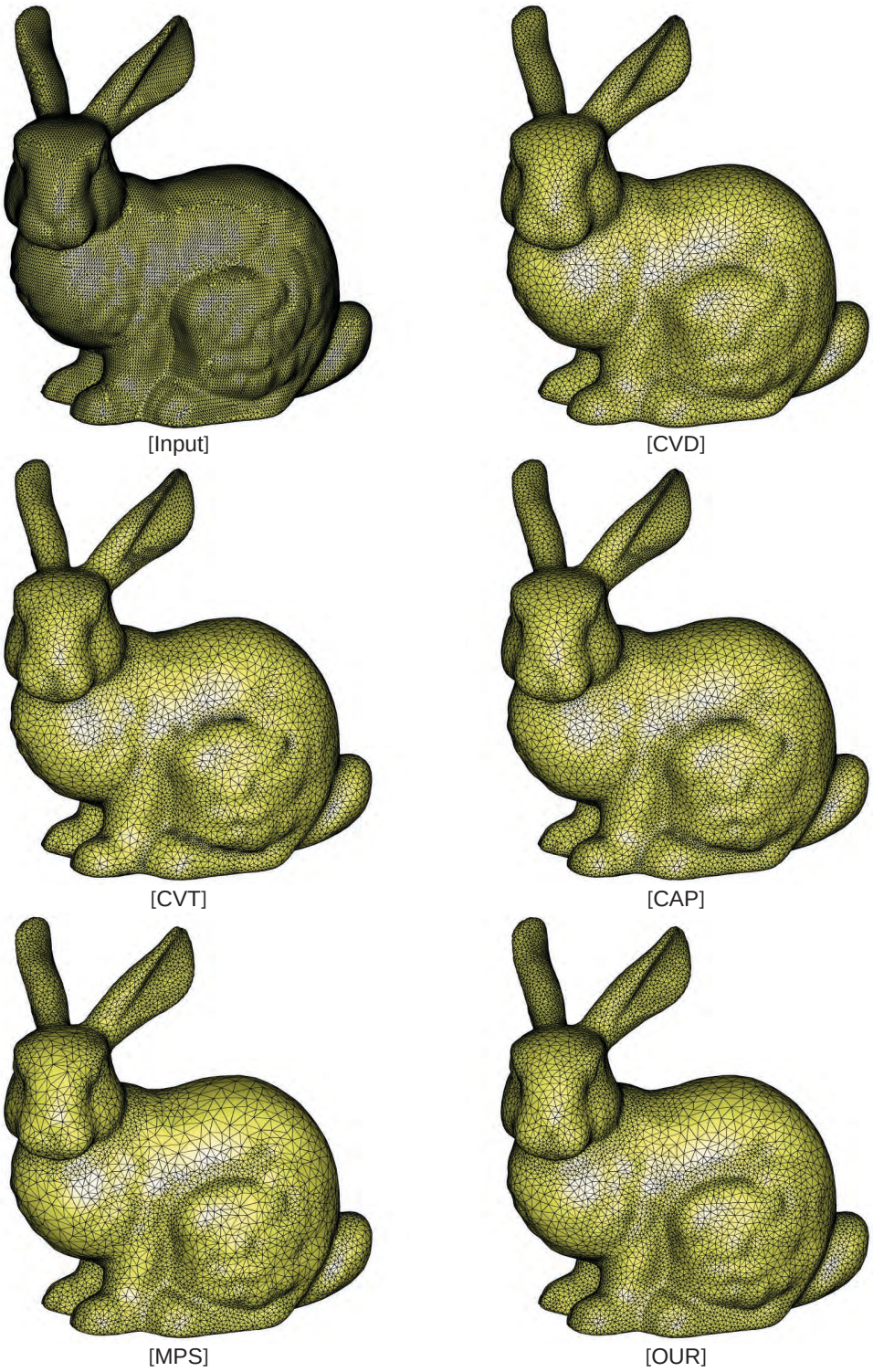


Fig. 4: *Remeshing results of the Bunny model.*

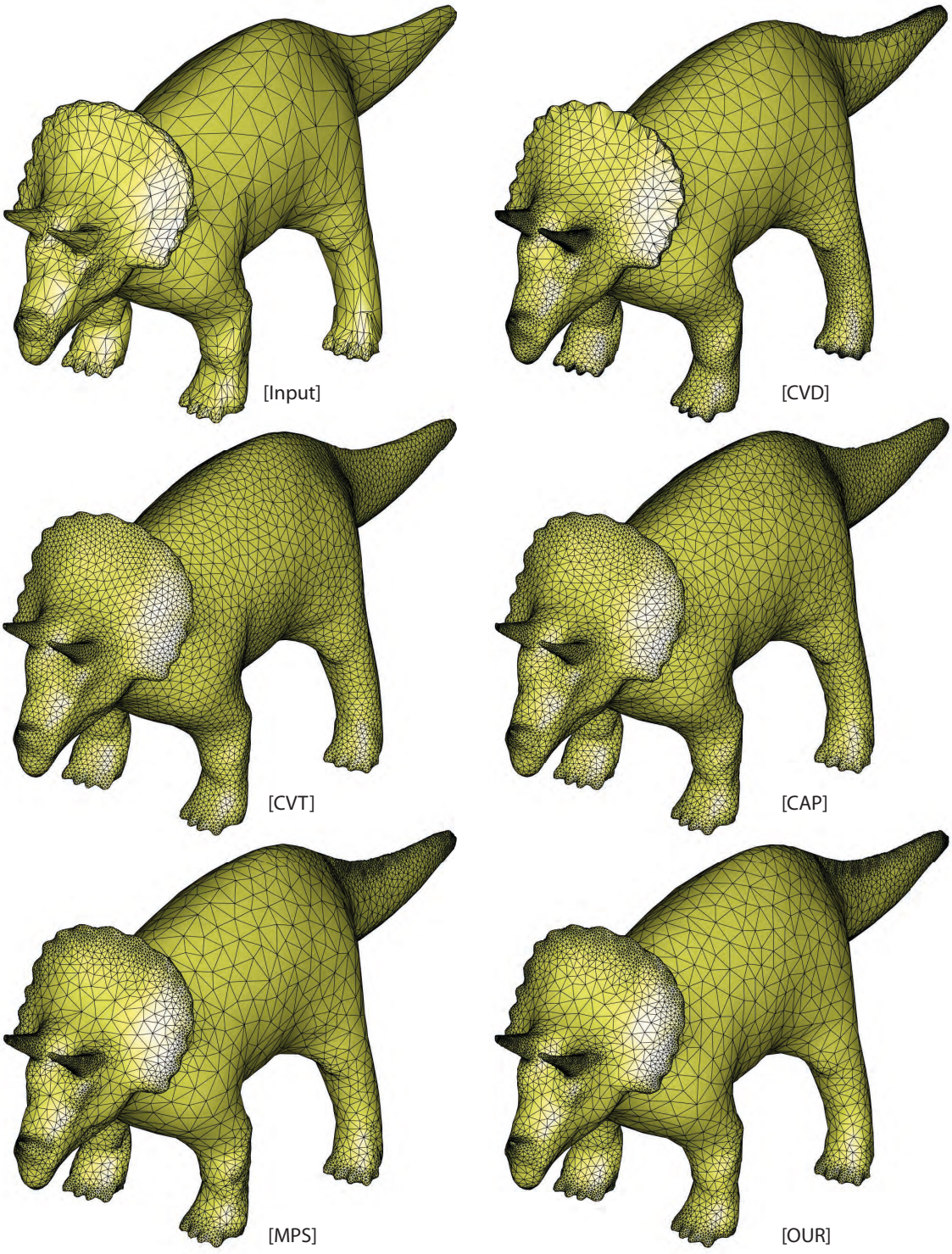


Fig. 5: *Remeshing results of the Triceratops model.*

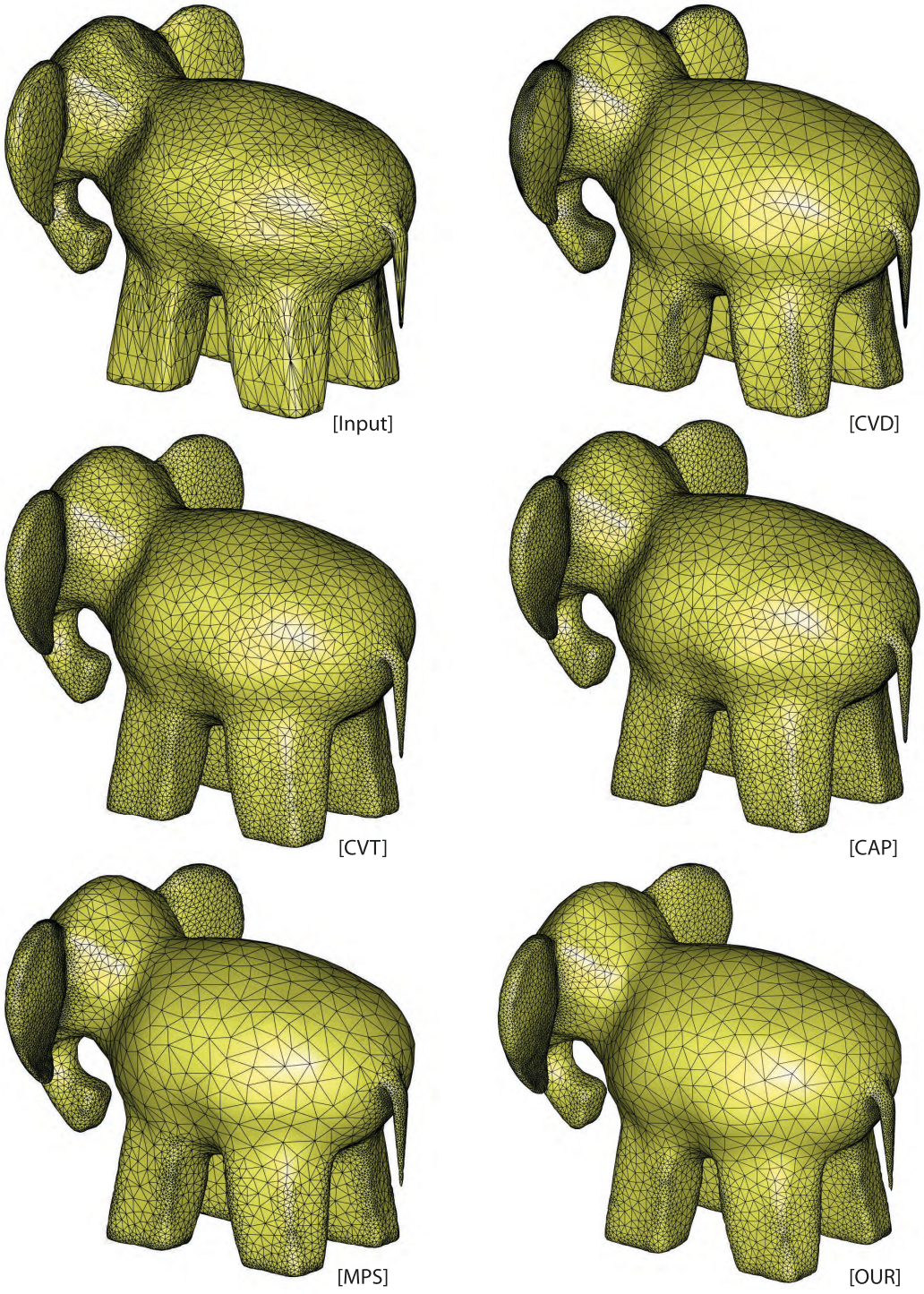


Fig. 6: *Remeshing results of the Elephant model.*

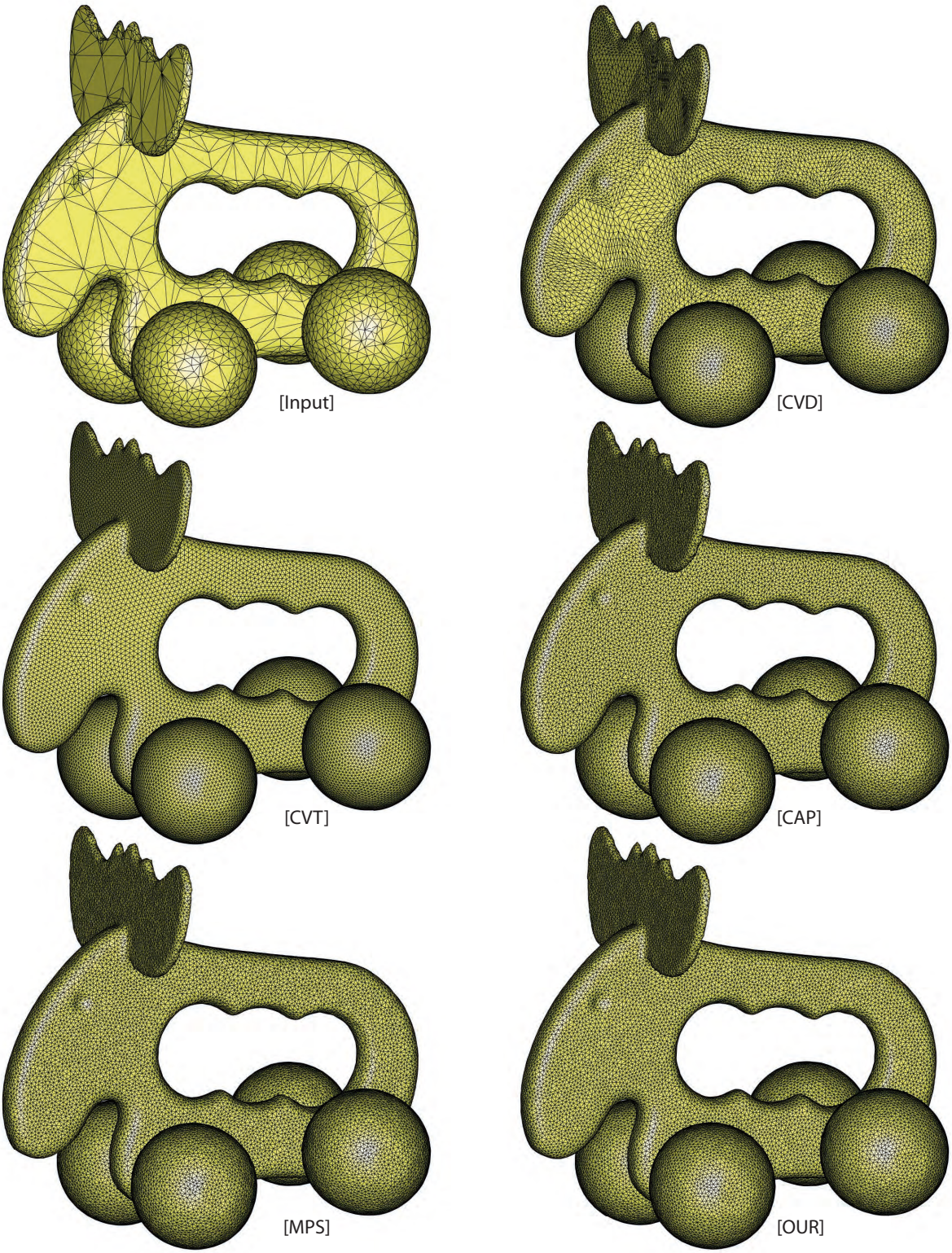


Fig. 7: *Uniform remeshing results of the Elk model. The angle bound of our result is* $[37^o, 98^o]$.

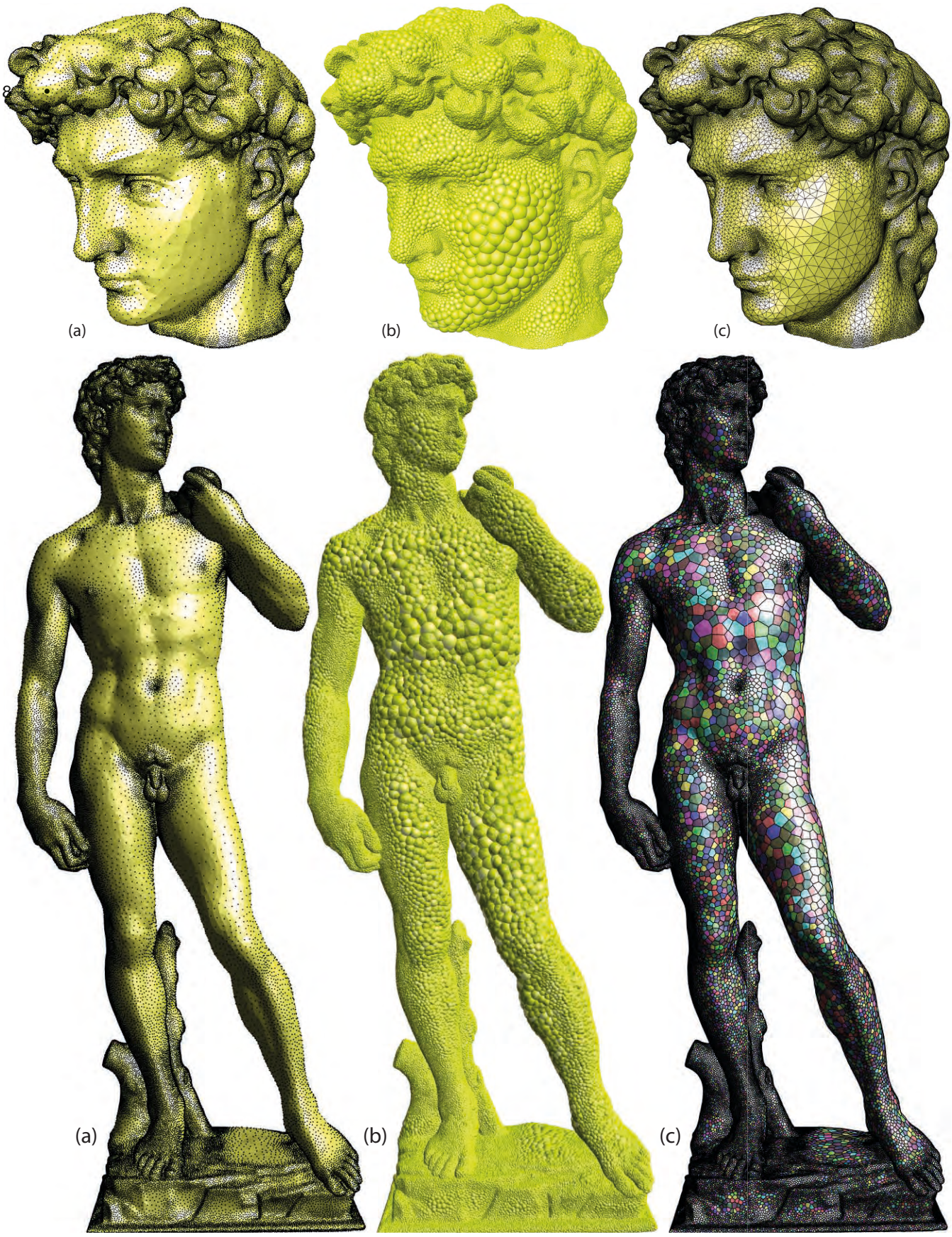


Fig. 8: *Top row: sampling of the David head. (a) Sample points, (b) spheres centered at points that cover the surface, (c) remeshing result. Bottom row: sampling results of the David model. (a) Sample points, (b) covering spheres, (c) restricted power diagram on the surface. The remeshed David has* $Q_{min} = 0.505$, $\theta_{min} = 32.0^o$ *and* $\theta_{max} = 115.0^o$.