Computer Aided Geometric Design 71 (2019) 43–62

Contents lists available at ScienceDirect

Computer Aided Geometric Design

www.elsevier.com/locate/cagd

Surface reconstruction by parallel and unified particle-based resampling from point clouds

Sikai Zhong, Zichun Zhong *, Jing Hua

Department of Computer Science, Wayne State University, Detroit, MI 48202, USA

ARTICLE INFO

Article history:
Available online 5 April 2019

Keywords:
Particle
Resampling
Meshing
Parallel
Point clouds

ABSTRACT

This paper introduces a new unified particle-based formulation for resamplings with specific patterns from original point clouds. Given the input point clouds, the proposed L_p-Gaussian kernel function is defined to simulate the inter-particle energy and force to form the isotropic/adaptive/anisotropic hexagonal and quadrilateral sampling patterns. Then, the particle-based optimization can be easily formulated and computed in parallel scheme with the high-efficiency and the fast convergence, without any control of particle population. Finally, based on the optimized particle distribution, the high-quality surface meshes are reconstructed by computing the restricted Voronoi diagram and its dual mesh with the parallel implementation. The experimental results are demonstrated by using extensive examples and evaluation criteria as well as compared with the state-of-the-art in the point cloud resampling and reconstruction.

1. Introduction

With the rapid development of various types of 3D scanners and imaging devices, point clouds and meshes are most common representations of 3D shapes. They are widely used in a variety of applications, such as designing 3D models for manufacturing, creating 3D shapes for animations, representing 3D objects for autonomous driving, etc. How to efficiently and effectively build such models is a challenging research problem. Traditionally, large-size and high-resolution raw point clouds or meshes are required to represent complicated 3D objects with a lot of features and details. It is difficult to build a real-time system to process the high-resolution 3D models in real 3D scenarios, such as registration, segmentation, classification, simulation, compression, transmission, etc. Hence it requires an unsufferable computational time and unaffordable storage, e.g., a 3D point cloud of 30M samples has 1GB data size. Therefore, it is important to resample the original dense point clouds into the relative sparse point sets with high-quality and high-fidelity, and then compute the meshes to reconstruct the 3D surfaces.

Point cloud resampling and surface remeshing are possible solutions to address the aforementioned problem. There are extensive methods for uniform and adaptive resampling from point clouds Lipman et al. (2007), Huang et al. (2009), Liao et al. (2013), Huang et al. (2013), Preiner et al. (2014), Luo et al. (2018), Chen et al. (2018), and isotropic and anisotropic remeshing from surface meshes Shimada and Gossard (1995), Shimada et al. (1997), Du et al. (1999, 2003), Yan et al. (2009), Du and Wang (2005), Valette et al. (2008), Lévy and Liu (2010), Zhong et al. (2013).

* Corresponding author.
E-mail addresses: sikai.zhong@wayne.edu (S. Zhong), zichunzhong@wayne.edu (Z. Zhong), jinghua@wayne.edu (J. Hua).

https://doi.org/10.1016/j.cagd.2019.04.011
0167-8396/

Among them, centroidal Voronoi tessellation (CVT) is a popular technique that has been successfully applied to isotropic/anisotropic resampling and remeshing. However, the CVT-based methods need to compute Voronoi diagram iteratively during the energy optimization Du et al. (1999, 2003), Yan et al. (2009), especially in adaptive and anisotropic cases Du and Wang (2005), Valette et al. (2008), Lévy and Liu (2010), Chen et al. (2018), which are complicated and time-consuming. On the other hand, in order to improve the efficiency and simplicity, particle-based methods are developed to formulate the inter-particle energy or force, and then generate the different sampling distributions Witkin and Heckbert (1994), Bossen and Heckbert (1996), Shimada and Gossard (1995), Shimada et al. (1997), Zhong et al. (2013), Ni et al. (2018). The particle insertion and deletion strategies are needed in most previous methods, which may not be efficient. Besides that, to our knowledge, all the previous particle-based methods are not designed and implemented in parallel schemes; and also not developed for point cloud applications.

In this paper, we propose a new unified particle-based method for high-quality resamplings from the input point clouds. The proposed particle-based optimization framework can be easily computed in a parallel scheme with the high efficiency. Finally, based on the optimized particle distribution, the surface meshes are reconstructed by computing the restricted Voronoi diagram (RVD) with a parallel implementation. The experimental results are demonstrated by using extensive examples and evaluation criteria as well as compared with the state-of-the-art in the point cloud resampling and reconstruction. The key *contributions* of our work are as follows:

- It proposes a unified L_p-Gaussian kernel function ($p = 2$ or $p = \infty$), by leveraging the classical L_2-Gaussian kernel, to simulate the inter-particle energy and force to optimize a set of particles to form the high-quality uniform/adaptive/anisotropic hexagonal and quadrilateral patterns from the input point clouds (with adaptive/anisotropic metrics).
- The L_∞-Gaussian kernel with squared L_p norm (such as $p \geq 4$) is a new and effective kernel formulation in the particle-based method for quadrilateral sampling and meshing.
- To our knowledge, this is the first time that a parallel particle-based computational algorithm is proposed for resampling of point clouds. Our empirical evaluations in several scenarios demonstrate that the proposed method is a practical utility with the fast speed.

2. Related work

In this section, we only review some most related work on point sampling methods, CVT-based methods, and particle-based methods for 3D shape surfaces/point clouds.

2.1. Point sampling methods

In computer graphics, sampling distribution on a shape has become an interesting research topic in the past few decades. There are some methods proposed to generate uniform distributions with blue noise properties on a surface mesh, such as Öztireli et al. (2010), Chen et al. (2012, 2013), Yan et al. (2015), which have been used in many applications such as stippling, rendering, as well as surface remeshing. Lipman et al. (2007) introduced the Locally Optimal Projection (LOP) operator for surface approximation from point-set data. Then, Huang et al. (2009) proposed an improved weighted-LOP (WLOP) operator, which is combined with a novel normal estimation and propagation algorithm, and this method can produce a set of clean and uniformly distributed points endowed with reliable normals. Liao et al. (2013) further considered both spatial and geometric feature information of the point clouds and proposed a feature-preserving LOP operator. Huang et al. (2013) proposed an edge-aware point set resampling method. Preiner et al. (2014) presented a continuous formulation of the WLOP operator and achieved a significant speed acceleration. Recently, Luo et al. (2018) presented a point cloud resampling method based on the Gaussian-weighted graph Laplacian to make the point distribution conformal to a target density distribution. All the aforementioned work does generally focus on uniform/adaptive point sampling without considering anisotropic and specific sampling patterns (e.g., hexagonal or quadrilateral). In this work, we propose an efficient method that constructs high-quality isotropic/anisotropic hexagonal/quadrilateral resampling results from point clouds.

2.2. CVT-based methods

A centroidal Voronoi tessellation (CVT) is a special Voronoi diagram, where each generating sample coincides with the centroid of its Voronoi cell Du et al. (1999). The Lloyd relaxation Lloyd (1982) and a quasi-Newton energy optimization solver Liu et al. (2009) are widely used to compute CVT and generate a regular sampling. In order to make the computation practical on 3D shape surfaces, the restricted Voronoi diagram (RVD) or restricted Delaunay triangulation (RDT) Edelsbrunner and Shah (1994) are used in meshing algorithms Dey and Ray (2010). Then, Du et al. (2003) defined a restricted CVT theoretically and then Yan et al. (2009) developed an efficient algorithm to compute the RVD for isotropic surface remeshing.

Du and Wang (2005) further extended the concept of CVT to the anisotropic case, that is anisotropic CVT (ACVT). An anisotropic Voronoi diagram (AVD) with the given Riemannian metric needs to be constructed in each Lloyd iteration, which is time-consuming. To improve the computational speed, Valette et al. (2008) proposed a discrete approximation of ACVT by clustering the vertices of a dense triangulation of the domain, by sacrificing the mesh quality. Lévy and Bonneel (2012) extended the computation of CVT to a 6D space (using vertex positions and normals) in order to achieve the

curvature-adaptation for anisotropic meshing, but did not provide user's flexibility to control the anisotropy. Zhong et al. (2014) provided a method to solve the anisotropic meshing by conformally mapping the metric surface to an appropriate 2D parametric domain and then compute CVT on it, but it cannot handle surfaces with complicated topologies. Lévy and Liu (2010) introduced an L_p-CVT, which is to minimize a higher-order norm of the coordinates on the Voronoi cells with the input surface meshes, and generate triangular and quad-dominant surface meshes. Recently, Chen et al. (2018) extended the L_p-CVT method for resampling isotropic or anisotropic distributions from a given point cloud.

In order to further improve the computations on CVT or Voronoi diagram, Vasconcelos et al. (2008) proposed to utilize GPU to compute CVT in a 2D plane, by using a pre-defined mask to estimate the Voronoi cell for each site. Rong et al. (2011) presented a GPU-assisted method to compute the constrained CVT on a surface, based on its 2D geometry image. The surface is discretized as pixels on a rectangular domain, and then the discrete Voronoi diagram is computed by using the jump flooding algorithm Rong and Tan (2006). Fei et al. (2014) solved the CVT problem by improving the non-linear optimization on GPU with a parallel L-BFGS-B algorithm. Recently, Boltcheva and Lévy (2017) presented a parallel method for reconstructing a 3D surface triangulation from an input point set by computing the RVD and implemented the algorithm by using a multi-core CPU. Ray et al. (2018) proposed a GPU-based algorithm to compute a 3D Voronoi diagram. Compared to all the above CVT-based approaches, our particle-based scheme avoids the construction of Voronoi diagram/AVD in the intermediate iterations of energy optimization. Thus, it can achieve faster performance in the computational complexity as demonstrated in Sec. 6.

2.3. Particle-based methods

Witkin and Heckbert (1994) constructed particles with pair-wise Gaussian energy to sample and control implicit surfaces. Meyer et al. (2005) formulated the energy kernel as a modified cotangent function with finite support. Bronson et al. (2012) worked on the particle-based isotropic and adaptive meshing of CAD models by using the parametric space of each surface patch. Zhong and Hua (2016) introduced a kernel-based adaptive sampling approach for 2D/3D image reconstruction and triangular meshing, but it does not handle with sampling for mesh or point cloud surfaces. As for anisotropic meshing, Bossen and Heckbert (1996) incorporated the metric into a distance function to model the repulsion and attraction forces between particles. Shimada and Gossard (1995), Shimada et al. (1997), Yamakawa and Shimada (2000) proposed physics-based relaxation of "bubbles" with a bounded cubic function of the distance to model the inter-bubble forces, and extended it to anisotropic meshing by converting spherical bubbles to ellipsoidal ones. Both Bossen et al. and Shimada et al.'s work requires to adaptively insert or delete particles/bubbles in certain regions. Thus, it takes a longer convergence time. Adaptive Smoothed Particle Hydrodynamics (ASPH) Shapiro et al. (1996) uses inter-particle Gaussian kernels with an anisotropic smoothing tensor, but it directly formulates the energy in the original space without using the embedding space concept. This procedure leads to an inaccurate anisotropy in the computed mesh as demonstrated in Zhong et al. (2013).

In order to address the above limitations, Zhong et al. (2013) showed that by formulating an accurate inter-particle energy optimization for anisotropic meshing in a high-d embedding space, such optimizations have the fast convergence without any need for the explicit control of particle population. Ni et al. (2018) proposed a particle-based method to optimize a set of particles to form the desired lattice pattern for tetrahedral meshing. In their method, the particle insertion and deletion strategies are required, which may not be as efficient as our proposed method. Besides that, all the previous particle-based methods are not designed and implemented in the parallel scheme.

3. A novel L_p particle formulation

In the particle-based framework, considering each sample as a particle, the potential energy between particles determines inter-particle forces. When forces applied on each particle become equilibrium, particles reach the optimal balanced state with the specified distribution under the given metric (e.g., isotropic or anisotropic). In the following subsection, we introduce and extend L_2 optimal approximation idea into a unified L_p in the designed inter-particle energy, which can guide particles to form the hexagonal and quadrilateral patterns in a 3D surficial (2-manifold) space when they reach the equilibrium.

Merits: (1) The progress of the physically-based system leads to force equilibrium of particles in the nature, such as honeycomb/grid, glass/steel panel structures, etc. Particle system is a natural design in the mesh generation Shimada and Gossard (1995), Shimada et al. (1997), Zhong et al. (2013), Ni et al. (2018) as well as the shape sampling Witkin and Heckbert (1994), Meyer et al. (2005), since it is essentially structurally-aware and physically-aware. **(2)** The computational complexity is relatively low (i.e., considering particle interactions with a small neighborhood) and the implementation is very simple, which is superbly efficient among sampling algorithms. The details will be given in Sec. 3.4 and Sec. 6. For example, compared with 2D CVT, it is complicated to compute restricted Voronoi cells on the shape surface, and optimize the sampling positions by using CVT method. **(3)** The particle-based system is easily formulated in the (isotropic or anisotropic) metric space as discussed in the following of this section, and the inter-particle energy can be independently calculated for parallelism. The details will be given in Sec. 4.3 and Algorithm 1.

Definition: The aim of *L_p-Gaussian kernel* is to propose a new energy function well suited to variational hexagonal and quadrilateral resampling. Given n particles with their positions $\mathbf{X} = \{\mathbf{x}_i | i = 1, \ldots, n\}$ on the point clouds of the shape

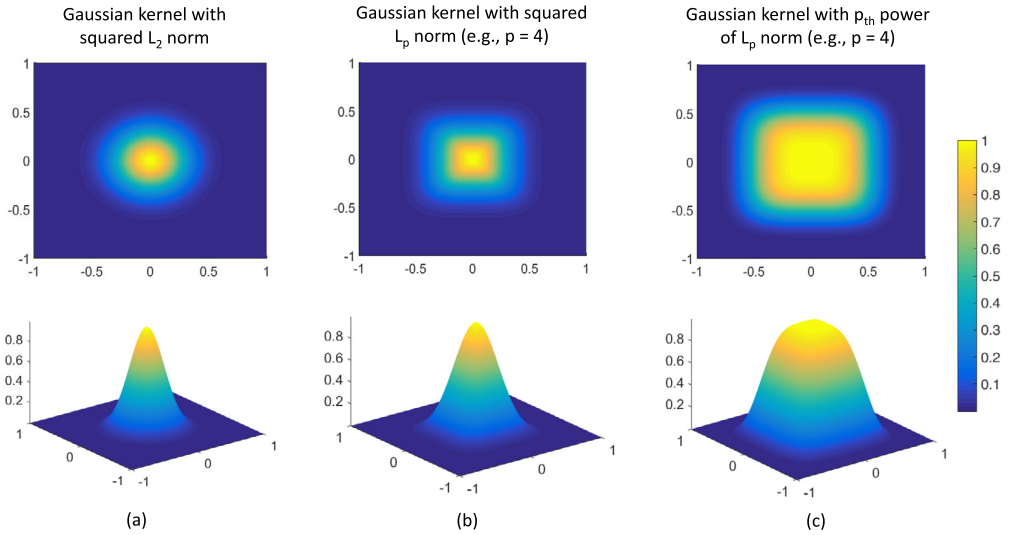

Fig. 1. The visualization of the proposed Gaussian kernels with different norms. The top row is visualized from the 2D x-y view and the bottom row is visualized from the 3D view. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

surface Ω embedded in $\mathbb{R}^m$ (m is the dimension of 2-manifold in this work), the *inter-particle energy* can be defined by an L_p-Gaussian kernel:

$$E_{L_p}^{ij} = e^{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|_p^2}{2\sigma^2}}, \tag{1}$$

where $\sigma = c_\sigma \sqrt{|\Omega|/n}$ is a kernel width. $|\Omega|$ denotes the area of underlying shape surface (represented by point clouds in this work) and c_σ is a constant. p is used to define different norms in finite-dimensional vector spaces. More detailed discussion about the kernel width and c_σ will be given in Sec. 4.2. In this work, we investigate several potential variants of L_p-Gaussian kernel for generating different sampling patterns in the following subsections.

3.1. Hexagonal sampling: L_2-Gaussian kernel

L_2-Gaussian kernel, i.e., $p = 2$ in Eq. (1), is a nonnegative radially-symmetric exponential function (Fig. 1 a). Essentially, in 2D kernel, domain is a circle, which leads to simulate the interactions between particles and form the ideal 2D hexagonal patterns of the sampling. L_2-Gaussian kernel is C^∞ continuous, ensuring to obtain a good optimum. For instance, the previous CVT-based energies for meshing are at most C^2 continuity Liu et al. (2009). The previous particle-based work, such as Zhong et al. (2013), Ni et al. (2018), Zhong et al. (2018), only designs and uses L_2-Gaussian kernel for mesh generation. In our work, we are working on the L_2-Gaussian kernel for resampling on point clouds.

Then, the gradient of $E_{L_2}^{ij}$ w.r.t. $\mathbf{x}_j$ is the force $\mathbf{F}_{L_2}^{ij}$ applied on particle j by the other particle i:

$$\mathbf{F}_{L_2}^{ij} = \frac{\partial E_{L_2}^{ij}}{\partial \mathbf{x}_j} = \frac{(\mathbf{x}_i - \mathbf{x}_j)}{\sigma^2} e^{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{2\sigma^2}}. \tag{2}$$

3.2. Quadrilateral sampling: L_∞-Gaussian kernel

L_∞-Gaussian kernel, i.e., $p = \infty$ in Eq. (1), is a nonnegative grid-symmetric exponential function. Essentially, in 2D kernel, domain is a square, which leads to simulate the interactions between particles and form the ideal 2D quad patterns of the sampling. However, the L_∞ norm is not differentiable. The L_p norm is a good approximation, and easier to manipulate algebraically (Fig. 1 b). $||\cdot||$ represents the L_p norm. In other words, $||\mathbf{x}||_p = \sqrt[p]{|x|^p + |y|^p + |z|^p}$ and $||\mathbf{x}||_p^2 = \sqrt[p]{|x|^p + |y|^p + |z|^p}$. In this work, we use L_4 in a Gaussian kernel for quadrilateral sampling.

Then, the gradient of $E_{L_p}^{ij}$ w.r.t. $\mathbf{x}_j$ is the force $\mathbf{F}_{L_p}^{ij}$ applied on particle j by the other particle i:

$$\mathbf{F}_{L_p}^{ij} = \frac{\partial E_{L_p}^{ij}}{\partial \mathbf{x}_j} = \frac{(\mathbf{x}_i - \mathbf{x}_j)^{p-1} \|\mathbf{x}_i - \mathbf{x}_j\|_p^2}{\sigma^2 \|\mathbf{x}_i - \mathbf{x}_j\|_p^p} e^{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|_p^2}{2\sigma^2}}. \tag{3}$$

Importance of squared L_p norm in Gaussian kernel: The visualization of the proposed Gaussian kernels with different norms is illustrated in Fig. 1. Fig. 1 (a) shows the classical L_2-Gaussian kernel with circular iso-contours. Fig. 1 (b) and (c)

show two variants of L_p-Gaussian kernel with square/quad iso-contours (e.g., L_4 norm as an example). Their key difference is: (b) uses the squared L_p norm and (c) uses the pth power of L_p norm (this is intuitive from L_p-CVT Lévy and Liu (2010)). Through Fig. 1, it is easy to see that the domain area and variance ratio of iso-contours in (b) are quite similar to those in (a), except the shape of the iso-contours, i.e., (a) is circular and (b) is square/quad. That is what we exactly desire. When we apply the pth power of L_p norm in a Gaussian kernel, the iso-contours are still square. However, the larger iso-contour values of the kernel are dominantly occupied in the domain and they are changing dramatically and steeply around domain boundary, which leads to the instability of the inter-particle energy and force/gradient.

3.3. Anisotropic and adaptive sampling

In order to simulate the adaptive or anisotropic sampling patterns, we can integrate the density or anisotropic metric in the proposed L_p-Gaussian kernel function. In anisotropic case, at a given point $\mathbf{x} \in \Omega$, the dot product between two vectors $\mathbf{a}$ and $\mathbf{b}$ is denoted by $\langle \mathbf{a}, \mathbf{b} \rangle_{\mathbf{M}(\mathbf{x})}$, which is defined over the tangent space of the surface (the point cloud). The metric can be represented by an $m \times m$ symmetric, positive, and definite (SPD) matrix $\mathbf{M}(\mathbf{x})$. Its square root $\mathbf{Q}(\mathbf{x}) = \sqrt{\mathbf{M}(\mathbf{x})}$ is also a unique SPD matrix Horn and Johnson (1990), which is defined on vectors $\mathbf{a}$ and $\mathbf{b}$. The *anisotropic* L_p-Gaussian kernel function is:

$$E^{ij}_{aniso} = e^{-\frac{\|\mathbf{Q}_{ij}(\mathbf{x}_i - \mathbf{x}_j)\|_p^2}{2\sigma^2}}, \tag{4}$$

where $\mathbf{Q}_{ij}$ is the anisotropic metric at the middle position of the particles i and j and it is defined on the vector $\mathbf{x}_i - \mathbf{x}_j$.

Then the gradient of E^{ij}_{aniso} is the force $\mathbf{F}^{ij}_{aniso}$ applied on particle j by the other particle i in the anisotropic embedding space (derived from Eq. (3) with integrating metric $\mathbf{Q}_{ij}$):

$$\mathbf{F}^{ij}_{aniso} = \frac{[\mathbf{Q}_{ij}(\mathbf{x}_i - \mathbf{x}_j)]^{p-1}\|\mathbf{Q}_{ij}(\mathbf{x}_i - \mathbf{x}_j)\|_p^2}{\sigma^2\|\mathbf{Q}_{ij}(\mathbf{x}_i - \mathbf{x}_j)\|_p^p} e^{-\frac{\|\mathbf{Q}_{ij}(\mathbf{x}_i - \mathbf{x}_j)\|_p^2}{2\sigma^2}}. \tag{5}$$

It is noted that when $\mathbf{M}(\mathbf{x}) = \rho(\mathbf{x})\mathbf{I}$, where $\mathbf{I}$ is an identity matrix, it defines an adaptive metric with the density function $\rho(\mathbf{x})$. Then, Eq. (4) becomes the *adaptive* L_p-Gaussian kernel function.

3.4. Particle objective function

Once the inter-particle L_p-Gaussian energy is defined, the particle optimization is formulated as an energy minimization problem, i.e., to sum up all inter-particle energies. The total objective function is:

$$E(\mathbf{X}) = \sum_i \sum_{j \neq i} E^{ij} \approx \sum_i \sum_{j \in N(i)} E^{ij}, \tag{6}$$

where $\mathbf{X} = \{\mathbf{x}_i | i = 1, \ldots, n\}$, which are constrained in the domain of the point cloud surface Ω, $N(i)$ is the set of neighbors of particle i within five standard deviations of the Gaussian energy kernel (5σ). The key reason is that we only need to consider the inter-particle energies within a certain neighborhood, instead of every pair of particles, since the Gaussian energy is close to 0 when the neighborhood distance is larger than 5σ. With the help of k-d tree data structure for computing K-Nearest Neighbors (K-NN), the computational complexity of particle optimization is $O(n \log n)$, instead of $O(n^2)$. This computational complexity also includes the k-d tree construction. In our implementation, Approximate Nearest Neighbor (ANN) library Mount and Arya (1998) is applied to efficiently search local neighborhoods. It is noted that, in practice, the number of output particles (samples) in this work is less than 1M, so that $\log n \leq 20$ (e.g., even if $n \approx$ 1B, $\log n = 30$). In conclusion, the practical computational performance of the proposed particle system is quite efficient.

4. Algorithm design on point clouds

In the following subsections, the details of the key components in the proposed algorithm on point clouds are discussed, i.e., tangent plane disk, kernel width, particle optimization, and surface reconstruction.

4.1. Tangent plane disk

Point clouds do not have explicit surface information, so we need to define the local tangent planes to constrain the particles during the optimization. The idea we use is to compute the local tangent plane disks derived from the original point clouds to approximate the underlying surfaces. The first step is to build a k-d tree for all the input point clouds. Then, the normal of each point is computed by using the least-square plane fitting estimation Rusu (2009) based on K-NN. The number of neighbors is 30 in our experiments. The normals do not need to be oriented, since we only use normals to define the tangent planes. After that, we need to compute suitable radiuses to define local tangent plane disks. The radius for every disk has to be adaptive because the original point clouds are not uniformly distributed, though they are relatively dense. There is a trade-off for overlaps and gaps between disks. When larger radius is used, more overlaps will exist. On

the other hand, when smaller radius is used, more gaps will appear. In our experiments, we use the radius defined based on the average distance between the point and its neighbors (such as six nearest neighbors) to determine the size of the local tangent plane disk.

4.2. Kernel width

The Gaussian kernel energies as defined in Eq. (1) and Eq. (4) depend on the choice of the fixed kernel width σ, which determines the final sampling. Once users specify the total number of particles, the proposed particle-based computation can be automatically optimized without any control of particle population for different models (e.g., inserting or deleting particles during the optimization).

The slope of the proposed energy peaks at distance of σ and it is near zero at much smaller or much greater distances. When σ is chosen too small, kernels will nearly stop spreading because there is almost no overlapping/interaction between Gaussian kernels, which may lead to aliasing and artifacts in the computed sampling. When σ is chosen too large, nearby kernels cannot repel each other and the resulting sampling pattern will be poor. In this work, σ is set to be proportional to the average "radius" of each kernel when they are uniformly distributed on the underlying surface $\overline{\Omega}$: $\sigma = c_\sigma \sqrt{|\overline{\Omega}|/n}$, when $|\overline{\Omega}|$ represents the area of the surface Ω in the embedding space (i.e., isotropic, adaptive, and anisotropic space), n is the number of particles, and c_σ is a constant coefficient. From our extensive experiments, we find out that the best value of c_σ is around 0.3.

It is noted that when the isotropic sampling is computed, $\overline{\Omega} = \Omega$ and it can be computed based on the area summation of all the local tangent plane disks as discussed in Sec. 4.1 (with multiplying a constant $\alpha = 1.3$). As for anisotropic and adaptive samplings, given any input anisotropic/adaptive metric field $\mathbf{M}(\mathbf{x})$ ($\mathbf{Q}(\mathbf{x}) = \sqrt{\mathbf{M}(\mathbf{x})}$), the area of the embedded space is: $|\overline{\Omega}| = \int_\Omega \det \mathbf{Q}(\mathbf{x}) ds$. In this work, the input surfaces are represented by the dense point clouds, with metric defined on each point. We need to approximate the surface area through point clouds. For each point $\mathbf{p}_i$, we use its nearest neighbors, e.g., six neighbors $\mathbf{p}_{i_1}, \ldots, \mathbf{p}_{i_6}$, to approximate a local disk area under the metric. The average distance between $\mathbf{p}_i$ and its neighbors in the embedded space is:

$$\overline{d}_{p_i} = \frac{1}{6} \sum_{j=1}^{6} || \frac{\mathbf{Q}(\mathbf{p}_i) + \mathbf{Q}(\mathbf{p}_{i_j})}{2} \cdot (\mathbf{p}_i - \mathbf{p}_{i_j}) ||. \tag{7}$$

Then, we use the above distance as the local disk radius to approximate the underlying local surface area covering by $\mathbf{p}_i$ as: $\overline{S}_{p_i} \approx \alpha\pi (\frac{\overline{d}_{p_i}}{2})^2$, where α is a constant ($\alpha = 1.3$) in our experiments. After summing all the disk areas of the point clouds, we can approximate the total area of the surface $|\overline{\Omega}|$ represented by the point clouds in the embedded space.

4.3. Particle optimization

During the particle optimization, with the summation of inter-particle energy defined in Eq. (6) and force defined in Eq. (2), or Eq. (3), or Eq. (5), we use the L-BFGS algorithm Liu and Nocedal (1989) to optimize the particle positions. It is a quasi-Newton method, which can quickly minimize the energy function of our particle-based sampling with less storage requirement. The total energy and the gradient are updated at each iteration during the optimization. As we discussed before, the gradient of particle $\mathbf{x}_i$ can be considered as the force $\mathbf{F}_i$ applied to itself. Furthermore, particles are supposed to move on the tangent space T_Ω of the surface. So the gradient used by the L-BFGS optimizer needs to be projected onto T_Ω:

$$\mathbf{F}_i|_{T_\Omega} = \mathbf{F}_i - [\mathbf{F}_i \cdot \mathbf{n}(\mathbf{x}_i)]\mathbf{n}(\mathbf{x}_i), \tag{8}$$

where $\mathbf{n}(\mathbf{x}_i)$ is the unit normal of particle $\mathbf{x}_i$ on the point cloud surface.

In the L-BFGS optimization, the particles are constrained on the input point cloud surface Ω. In each iteration, the updated particle positions need to be projected to their nearest locations on the tangent plane disks, if they are out of the boundary or out of the surface. It is noted that the particle optimization formulation is constructed by using a pushing energy and force based on an L_p-Gaussian kernel, which can maximally and optimally push particles away within the boundary constraint so as to cover the entire domain (point cloud surface), leading to automatically capture and reconstruct the boundaries without requiring the user's tagging as an extra input (such as CVT-based approaches need user's intervention). This optimization process is iterated until convergence by satisfying a specified stopping condition, e.g., the magnitude of the gradient is smaller than a threshold or the maximal number of iterations. In our experiments of this work, we use the number of iterations as the stopping condition and the detailed settings are given in Sec. 6.

Since no inter-thread communication/synchronization is required for each particle i (w.r.t. its inter-particle energy and force), parallelization of the algorithm is easy and it can directly gain a factor nearly linear in the number of CPU cores. The L-BFGS algorithm can be implemented in parallel of CPU cores by using HLBFGS package Liu (2010).

Data: Point cloud surface Ω with metric $\mathbf{M}$, the norm L_p, and the desired number of resampling points n
Result: The isotropic/anisotropic resampling $\mathbf{X}$ and surface mesh of Ω
Initialize particle locations $\mathbf{X}$;
while *stopping condition not satisfied* **do**
  Build k-d tree for the current sampling $\mathbf{X}$;
  for *each particle* i ***in parallel*** **do**
    Get particle i's neighbor $N(i)$ from k-d tree;
    for *each particle* $j \in N(i)$ ***in parallel*** **do**
      Compute E^{ij} using Eq. (1) or Eq. (4);
      Compute $\mathbf{F}^{ij}$ using Eq. (2) or Eq. (3) or Eq. (5);
    end
    Sum the total force $\mathbf{F}^i$;
    Project $\mathbf{F}^i$ to the tangent space using Eq. (8);
  end
  Sum the total energy E in Eq. (6);
  Run **parallel L-BFGS** using HLBFGS with E and $\mathbf{F}^i$, to get updated locations $\mathbf{X}$;
  Project $\mathbf{X}$ onto the tangent plane disks or the boundary of the point clouds **in parallel**;
end
Compute the surface mesh **in parallel** using Geogram from the computed $\mathbf{X}$;

Algorithm 1: Parallel Particle Optimization and Surface Reconstruction.

4.4. Surface reconstruction

4.4.1. Triangular meshing

After the L_2 particle optimization, the final output triangular mesh is generated as the dual of restricted Voronoi diagram (RVD) Yan et al. (2009). The parallel computation of RVD on point clouds is developed by using the method Boltcheva and Lévy (2017). The idea is to compute the intersection between the 3D Voronoi diagram of the computed particles and a set of disks centered at such particles. Once the RVD is obtained, we can easily compute its dual graph, i.e., restricted Delaunay triangulation (RDT). The algorithm robustly computes the RVD using filtered geometric predicates and symbolic perturbation to resolve degeneracies Lévy (2016). Generally, there is no guarantee that the dual mesh will not have inverted elements. Whenever such an inverted element is detected, our implementation inserts additional points using the provably terminating algorithm in Rouxel-Labbé et al. (2016), which is also used in Lévy and Bonneel (2012). In our implementation, Geogram library ALICE/INRIA Nancy (2018) is used to generate isotropic and adaptive meshes (both in sequential and parallel). When reconstructing anisotropic meshes, the dual of anisotropic Voronoi diagram (AVD) Du and Wang (2005) restricted on point clouds is generated. We implement it by modifying the Geogram library in anisotropic case (both in sequential and parallel).

4.4.2. Quad-dominant meshing

After the L_p particle optimization, the quad-dominant mesh is extracted from the RDT (computed in the previous subsection) by merging pairs of triangles. Firstly, for every triangle $\triangle_i$, all its neighbors $\triangle_{i_j}$ are found. The neighbors and itself can construct the potential quads. All of those pairs are sorted by the angles at the corners of the so-obtained quads. Then, the triangles are merged in the priority order, as suggested by Lévy and Liu (2010). Finally, we can apply the all-quad mesh conversion technique Itoh and Shimada (2002) to clean all non-quad elements if necessary (which is beyond the scope of this paper).

In conclusion, our parallel particle-based resampling and reconstruction can be summarized in Algorithm 1.

5. Evaluations

5.1. Triangular mesh quality

To measure the isotropic triangular mesh quality, we use the criteria as follows Frey and Borouchaki (1999). The quality of a triangle is measured by $G = 2\sqrt{3}\frac{S}{ph}$, where S is the triangle area, p is its half-perimeter, and h is the length of its longest edge. G_{min}, G_{avg} are the minimal and average qualities of all triangles. θ_{min}, θ_{avg} are the smallest and average angles of the minimal angles of all triangles. $\%_{<30}$ is the percentage of triangles with their minimal angles smaller than 30 degrees. The angle histogram is also provided. It is noted that θ_{avg} should be 60 degrees if it is a regular triangle. G is between 0 and 1, where 0 denotes a skinny triangle and 1 denotes a regular triangle.

In the anisotropic triangular meshing, for each triangle $\triangle_{abc}$ in the final mesh, we use its approximated metric $\mathbf{Q}(\triangle_{abc}) = (\mathbf{Q}(\mathbf{x}_a) + \mathbf{Q}(\mathbf{x}_b) + \mathbf{Q}(\mathbf{x}_c))/3$ to affine-transform it from the original anisotropic space into the Euclidean space. After that, we employ the previous isotropic triangular criteria, used in Zhong et al. (2013) and Fu et al. (2014)'s recent work, to evaluate the quality of generated anisotropic triangular mesh.

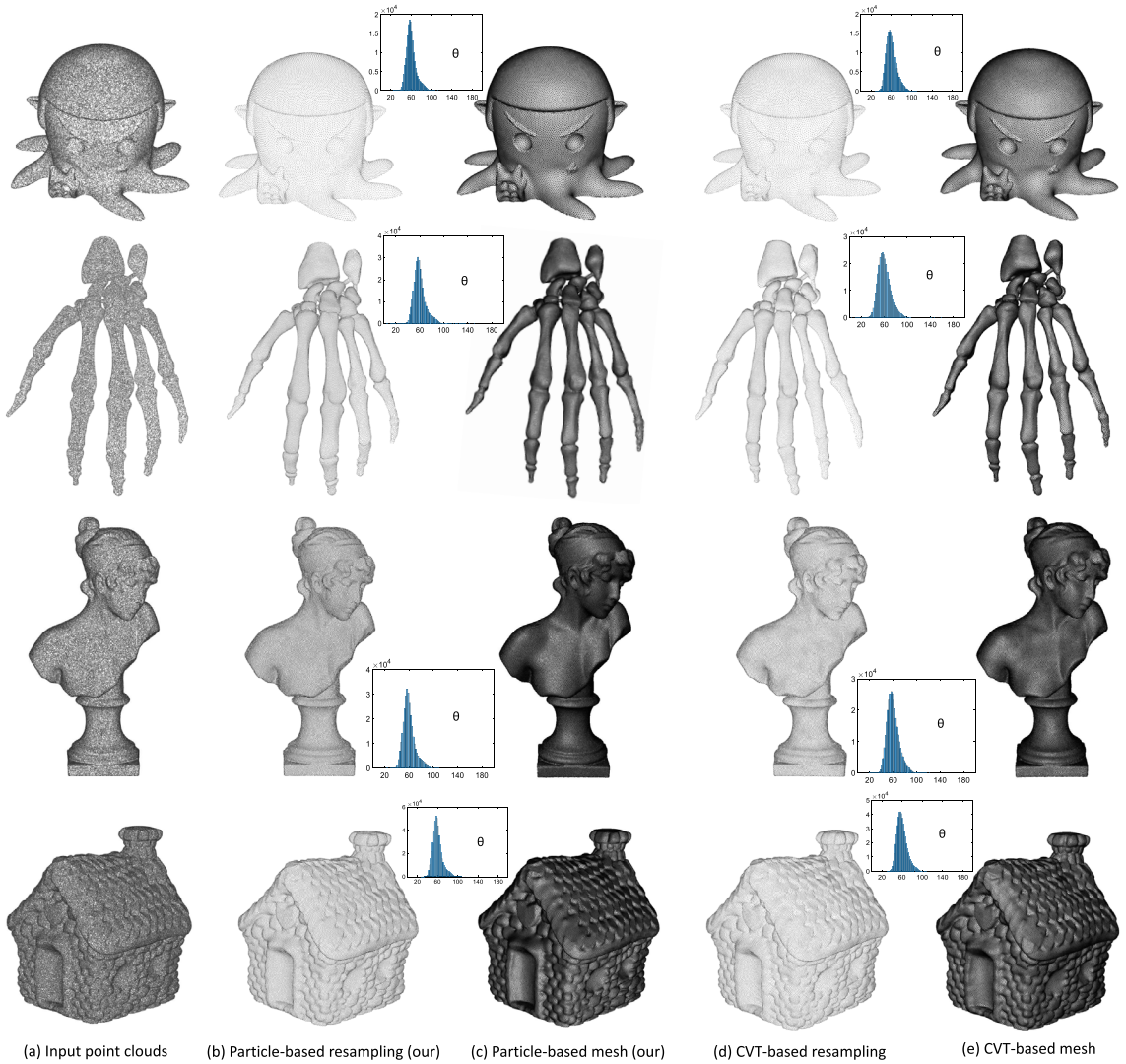

Fig. 2. Comparisons on uniform hexagonal resamplings and isotropic triangular meshes with CVT-based method Chen et al. (2018) and our particle-based method on Cute Spocktopus, Hand Skeleton, Sapphos Head, and Stonehaus1 point cloud models.

5.2. Quad-dominant mesh quality

To measure the quad-dominant mesh quality, we use the criteria as follows: $\%_{\square}$ is the percentage of the quads in all mesh elements. θ_{min}, θ_{avg} are the smallest and average angles of the minimal angles of all elements. $\%_{<30}$ is the percentage of elements with their minimal angles smaller than 30 degrees. The angle histogram is given as well. It is noted that θ_{avg} should be 90 degrees if it is a regular quad.

6. Results

We develop our algorithms by using Microsoft Visual C++ 2015 in sequential implementation, and OpenMP 2.0 in the parallel implementation. The mesh quality evaluations are implemented with Matlab R2015a. For the hardware platform, the experiments are run on a desktop computer with Intel(R) Core(TM) i7-6850K Processor with 12 threads (6 cores), 15MB Cache, 4.0 GHz, 32GB DDR4 RAM. Users provide the desired numbers of output samples for all the experiments.

6.1. Isotropic hexagonal resampling and mesh reconstruction

Comparison with CVT-based method: In this subsection, we first evaluate and compare our method with a recent state-of-the-art resampling approach on point clouds, i.e., a CVT-based method Chen et al. (2018), which is most close to our work. Fig. 2 shows the visualization comparison with the CVT-based method and our method (in L_2-Gaussian kernel), on

Table 1
Comparisons with CVT-based method Chen et al. (2018) and our particle-based method on statistics and timings for resampling and surface reconstruction.

Model	Method	#Points	#Samples	G_{min}	G_{avg}	θ_{min}	θ_{avg}	$\%_{<30^\circ}$	T_R (Seq.)	T_M (Seq.)	T_R (Para.)	T_M (Para.)
Cute Spocktopus	CVT	182, 351	30, 000	0.38	0.88	17.24°	51.11°	0.01%	37.72 s	0.48 s	–	–
Cute Spocktopus	Particle	182, 351	30, 000	**0.49**	**0.89**	**25.47°**	**52.16°**	**0.008%**	**14.97** s	0.48 s	**3.07** s	**0.31** s
Hand Skeleton	CVT	269, 889	50, 000	0.05	0.87	2.02°	50.16°	0.2%	54.79 s	0.70 s	–	–
Hand Skeleton	Particle	269, 889	50, 000	**0.14**	**0.89**	**4.92°**	**51.91°**	**0.1%**	**24.13** s	0.70 s	**4.24** s	**0.50** s
Sapphos Head	CVT	271, 937	50, 000	0.44	0.88	19.74°	50.93°	0.022%	65.87 s	0.74 s	–	–
Sapphos Head	Particle	271, 937	50, 000	**0.50**	**0.89**	**23.43°**	**52.34°**	**0.007%**	**26.73** s	0.74 s	**5.53** s	**0.48** s
Stonehaus1	CVT	639, 842	80, 000	**0.51**	0.88	21.42°	51.07°	0.003%	133.36 s	1.09 s	–	–
Stonehaus1	Particle	639, 842	80, 000	0.42	**0.90**	**27.22°**	**52.55°**	0.003%	**56.33** s	1.09 s	**10.43** s	**0.67** s

Note: #Points: the number of points in the input point clouds. #Samples: the number of output samples. T_R (Seq.) and T_R (Para.): timings for resampling computations in sequential and parallel designs with 35 iterations. T_M (Seq.) and T_M (Para.): timings for meshing computations in sequential and parallel designs. The best values are highlighted in bold for each group. It is noted that the timings T_M (Seq.) for meshing computations on CVT and Particle methods are the same, since both of them use Geogram to compute the final mesh.

several point clouds (e.g., Cute Spocktopus, Hand Skeleton, Sapphos Head, and Stonehaus1 models). The final mesh angle histograms show that our method has more regular triangles (i.e., more number of angles close to 60°/more regular hexagonal sampling patterns). In essence, the particle method is a physically-based system, which can lead to the force equilibrium of particles in the nature, so as to generate the nice sampling and meshing results. Table 1 shows the statistics and timings for resampling computation and surface reconstruction with CVT-based method Chen et al. (2018) and our particle-based method. In quantitative, we can clearly see that our particle-based method has better sampling and mesh quality (especially G_{avg}, θ_{avg}). Meanwhile, our method also has the faster convergence speed, since the CVT-based method needs to compute Voronoi diagram iteratively during the energy optimization and particle-based method only needs to compute the inter-particle energy and force in each iteration, which is much more efficient. Theoretically, the computational complexity of CVT-based method by using Lloyd's algorithm is $O(m \log n)$ for each iteration, where m is the number of points in the original point clouds Secord (2002), Chen et al. (2018); while our particle-based method is $O(n \log n)$ for each iteration, where n is the number of output samples. Note that, m is always much greater than n in our point cloud applications. We have developed and implemented our particle-based resampling and meshing computations both in sequential and parallel ways. Our method is several times faster than CVT-based method in the sequential design, not to mention that in the parallel design, our method is even faster. In the implementation of CVT-based method, each RVD is obtained by using the clipping method Lévy and Bonneel (2012), Chen et al. (2018) from Geogram library ALICE/INRIA Nancy (2018).

Due to the page limit, there are some more uniform hexagonal resampling and isotropic triangular meshing results and comparisons on different 3D point clouds given in Appendix.

Large-size point clouds: In order to demonstrate the scalability of the proposed method on large-size resampling and meshing of point clouds, Fig. 3 shows the final results with resamplings ranging from 150K to 200K of the original point clouds with more than 1M points. It is noted that our particle-based method is quite efficient on the large-size models, and it takes 47.96 s (100 iterations and meshing) and 47.84 s (60 iterations and meshing) for Arch and Lucy models (in parallel implementation), respectively. The final resampling and mesh quality are quite good as well.

6.2. Anisotropic and adaptive resampling and mesh reconstruction

Anisotropic metric on point clouds: For the anisotropic resampling and meshing on 3D point clouds, we use the following 3×3 metric tensor: $\mathbf{M} = [\mathbf{v}_{min}, \mathbf{v}_{max}, \mathbf{n}] diag(1, (\frac{s_2}{s_1})^2, 0) [\mathbf{v}_{min}, \mathbf{v}_{max}, \mathbf{n}]^T$, where $\mathbf{v}_{min}$ and $\mathbf{v}_{max}$ are the directions of the principal curvatures, $\mathbf{n}$ is the unit point normal. s_1 and s_2 are two user-specified stretching factors along principal curvature directions. Since the resampling and meshing are computed by curvature-based metric tensor fields, we use the above metric with $s_1 = \sqrt{K_{min}}$ and $s_2 = \sqrt{K_{max}}$, where K_{min} and K_{max} are the principal curvatures. We set small thresholds to preserve both K_{min} and K_{max} not vanishing. The principal curvatures and normals of original dense point clouds are estimated by PCL library Rusu and Cousins (2011). Then, Laplacian smoothing is applied to both the stretching factors and directions, to ensure smoothness of the input metric field on the point cloud surface. $\frac{s_2}{s_1}$ is defined as stretching ratio (≥ 1) in the metric field $\mathbf{M}$ of the point clouds. The adaptive metrics are computed based on the mean curvatures of the original dense point clouds as the density function.

Anisotropic results: Fig. 4 shows the visualization results of our method (in anisotropic L_2-Gaussian kernel) on anisotropic resampling and mesh reconstruction on several point cloud models with curvature-based anisotropic metrics, such as Duck, Kitten, Fertility, Gargo, Genus3, and Rocker Arm. By using our method, samples are denser along the maximal principal curvature directions and stretched along the minimal principal curvature directions. While in the low stretching regions (i.e., isotropic metric regions), the regular sampling distributions are obtained. The final mesh angle histograms show that our method has good mesh quality to match the input curvature-based anisotropy. Table 2 shows the statistics and timings for anisotropic resampling computation and surface reconstruction by using our particle-based method. We can

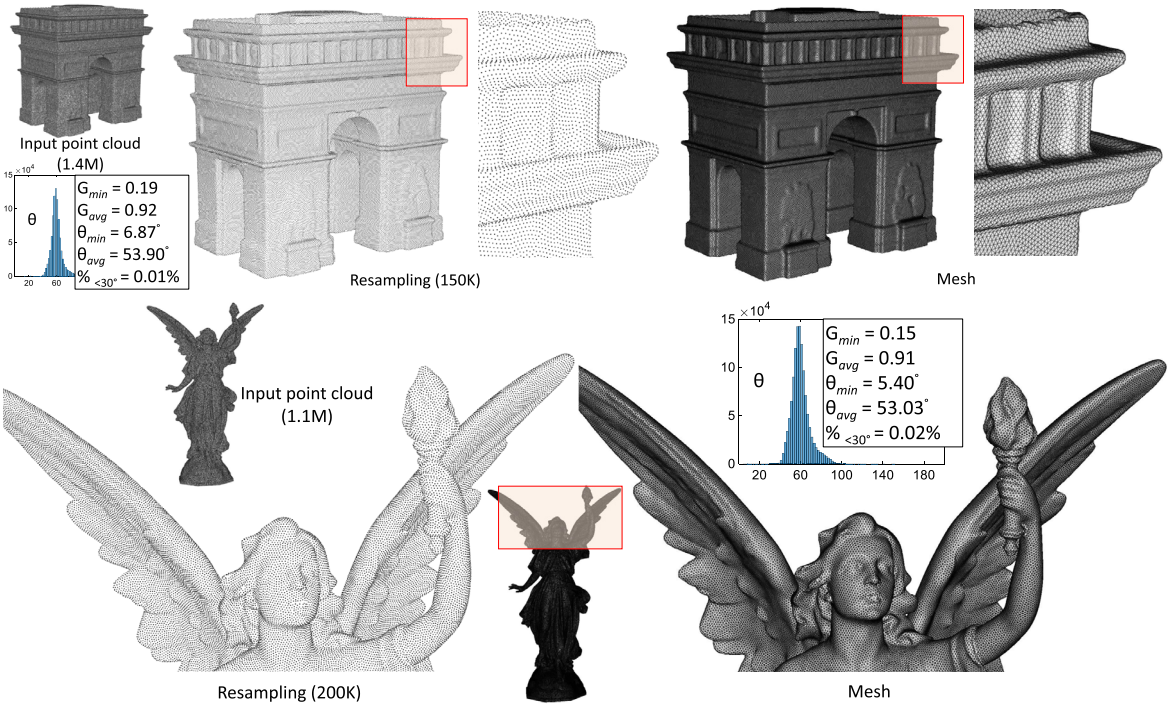

Fig. 3. More uniform hexagonal resampling and isotropic triangular meshing results on large-size Arch and Lucy point cloud models (more closeups of Lucy result are given in Appendix).

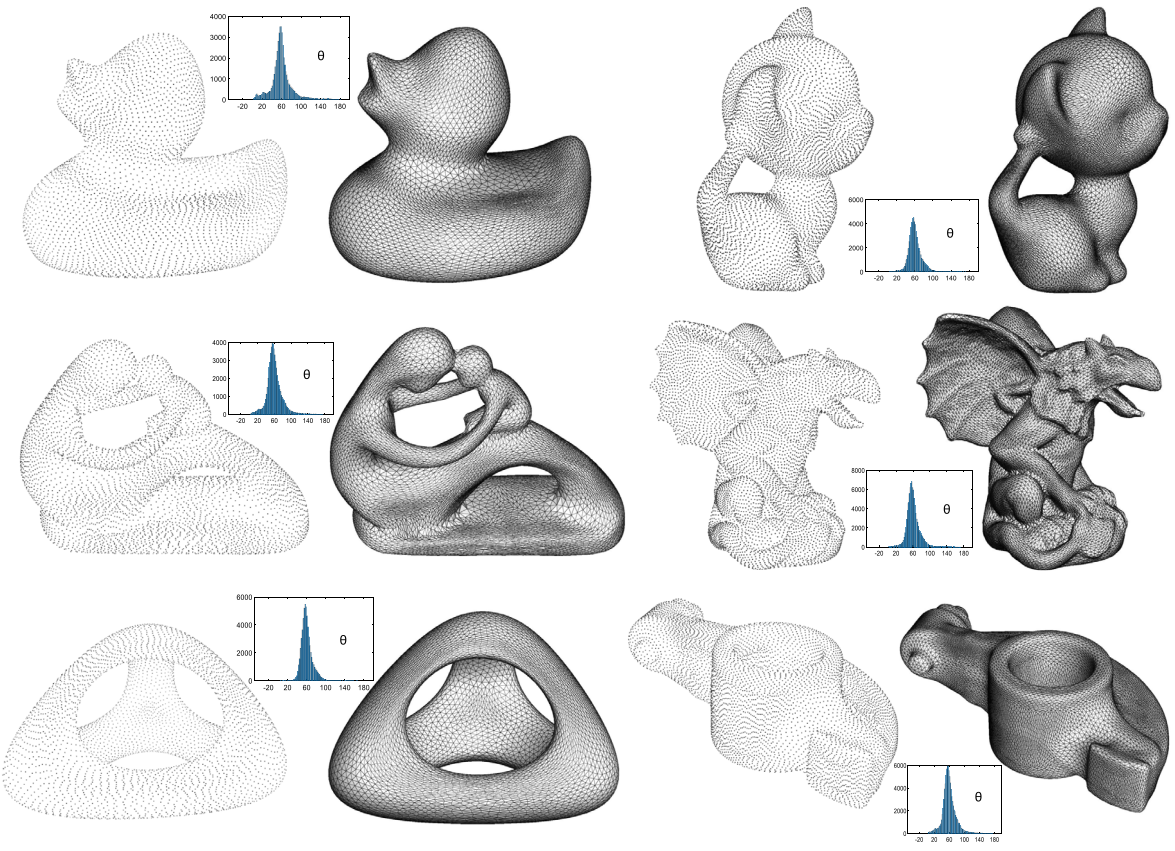

Fig. 4. Anisotropic resampling and anisotropic triangular meshing results on several point cloud models with curvature-based anisotropies.

Table 2
Statistics and timings for anisotropic resampling and surface reconstruction by our particle-based method.

Model	#Points	#Samples	Stretch	T_R (Para.)	T_M (Para.)
Duck	95, 990	8000	[1, 7.4]	7.48 s	0.23 s
Kitten	96, 000	10, 000	[1, 7.2]	8.76 s	0.23 s
Fertility	268, 357	10, 000	[1, 8.3]	12.52 s	0.25 s
Gargo	240, 144	15, 000	[1, 6.5]	12.61 s	0.30 s
Genus3	141, 994	10, 000	[1, 6.2]	8.21 s	0.23 s
Rocker Arm	172, 032	13, 000	[1, 6.9]	9.62 s	0.29 s

Note: #Points: the number of points in the input point clouds. #Samples: the number of output samples. T_R (Para.): timings for resampling computation in parallel design with 50 iterations. T_M (Para.): timings for meshing computation in parallel design.

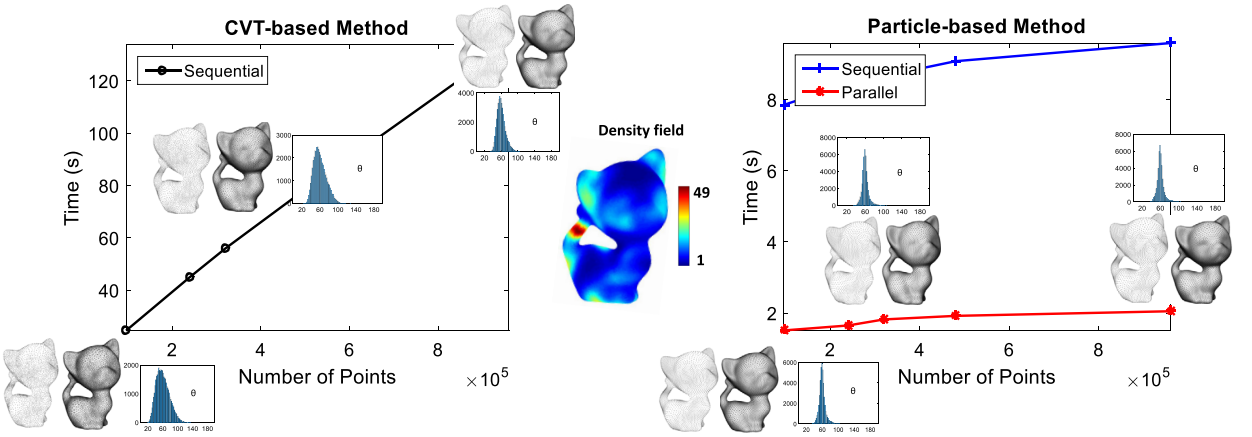

Fig. 5. Comparisons on computational time (50 iterations) with CVT-based method Chen et al. (2018) and our particle-based method on Kitten point cloud models (with different resolutions) for adaptive resampling and adaptive triangular meshing results with 7000 samples.

see that our method is quite efficient on anisotropic resampling and meshing. There are some more anisotropic resampling and meshing results given in Appendix.

Comparison with CVT-based method: Fig. 5 shows the comparisons on computational time with CVT-based method Chen et al. (2018) and our particle-based method (in adaptive L_2-Gaussian kernel) on Kitten point cloud model for adaptive resampling and adaptive triangular meshing results. The number of the output samples is fixed at 7000 and the point numbers of the input cloud points range from 96,000 to 960,000. In CVT-based methods with density function (e.g., Du and Wang (2005), Chen et al. (2018)), they need to compute the Voronoi cells with density in each iteration, whose accuracy highly depends on the numerical computation of the quadrature/integral rule on cells. It is easy to note in Fig. 5 that the denser the input cloud points are, the higher resampling and meshing qualities are. However, our particle-based method does not need to compute the integral, and we only need to compute the inter-particle energy and force, which is much faster with a stable higher accuracy in different resolutions of cloud points. For instance, in order to achieve competitive results, CVT-based method is about 15x times slower than our particle-based method in sequential design (about 80x in parallel design) as shown in Fig. 5 on Kitten model.

6.3. Quadrilateral resampling and mesh reconstruction

In the quadrilateral resampling, we use multi-scale strategy for sampling optimization to achieve better results, due to the L_∞-Gaussian kernel function is easier to be trapped in its local minima compared with L_2-Gaussian kernel function. In all our quad experiments, we use L_4-Gaussian kernel function and two-level multi-scale sampling strategy. The sampling points at first level are split into four subsampling points at the next level. We found that multi-scale strategy can improve both the mesh quality as well as the computational speed. Fig. 6 shows the visualization results of our method on quadrilateral resampling and mesh reconstruction on several point cloud models by using the proposed L_4-Gaussian kernel. Table 3 shows the statistics and timings for quadrilateral resampling computation and surface reconstruction. We can see that our method can efficiently generate high-quality quad-dominant meshes with high quad-element percentages. There are some more (large-size) quadrilateral resampling and quad-dominant meshing results on different 3D point clouds given in Appendix.

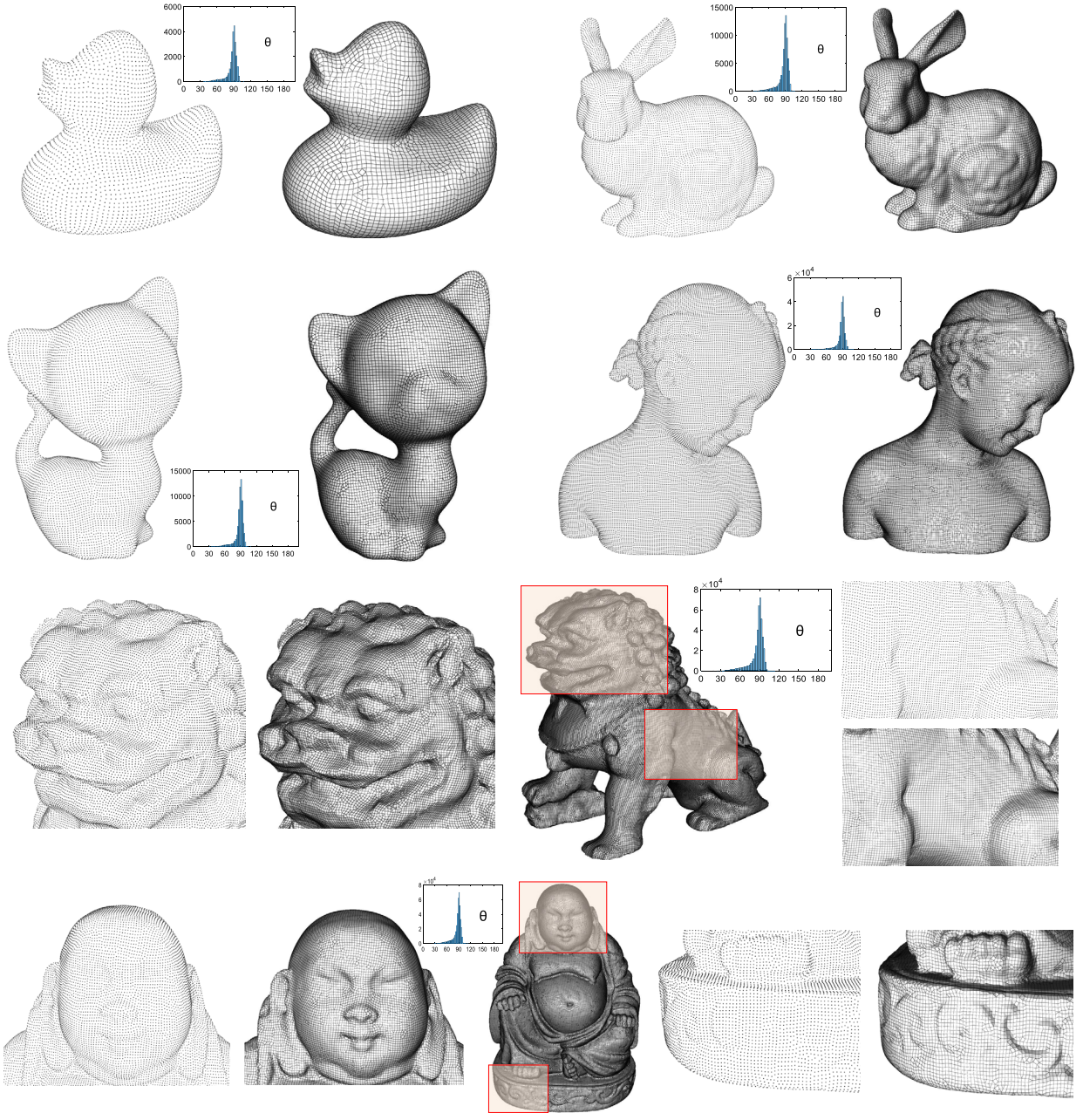

Fig. 6. Quadrilateral resampling and quad-dominant meshing results on several point cloud models.

Table 3
Statistics and timings for quadrilateral resampling and surface reconstruction by our particle-based method.

Model	#Points	#Samples	#Quad%	θ_{min}	θ_{avg}	$\%_{<30^\circ}$	T_R (Para.)	T_M (Para.)	Tri2Quad
Duck	32,000	6000	91.63%	34.17°	80.71°	0	1.89 s	0.23 s	0.10 s
Bunny	83,191	18,000	91.14%	32.24°	80.23°	0	3.40 s	0.26 s	0.26 s
Kitten	96,000	16,000	93.43%	30.11°	82.07°	0	3.66 s	0.24 s	0.23 s
Bimba	254,309	50,000	93.21%	30.60°	81.89°	0	8.24 s	0.50 s	0.71 s
Lion	575,573	100,000	90.18%	19.74°	79.61°	0.01%	19.12 s	0.80 s	1.51 s
Buddha	687,347	100,000	90.02%	15.91°	79.44°	0.009%	15.50 s	0.69 s	1.46 s

Note: #Points: the number of points in the input point clouds. #Samples: the number of output samples. #Quad%: the percentage of quad elements in the output meshes. T_R (Para.): timings for resampling computation in parallel design with 75 iterations (i.e., multi-scale strategy including 40 iterations in the first level and 35 iterations in the second level). T_M (Para.): timings for triangular meshing computation in parallel design. Tri2Quad: timings for merging the triangles into quad-dominant meshes (in sequential).

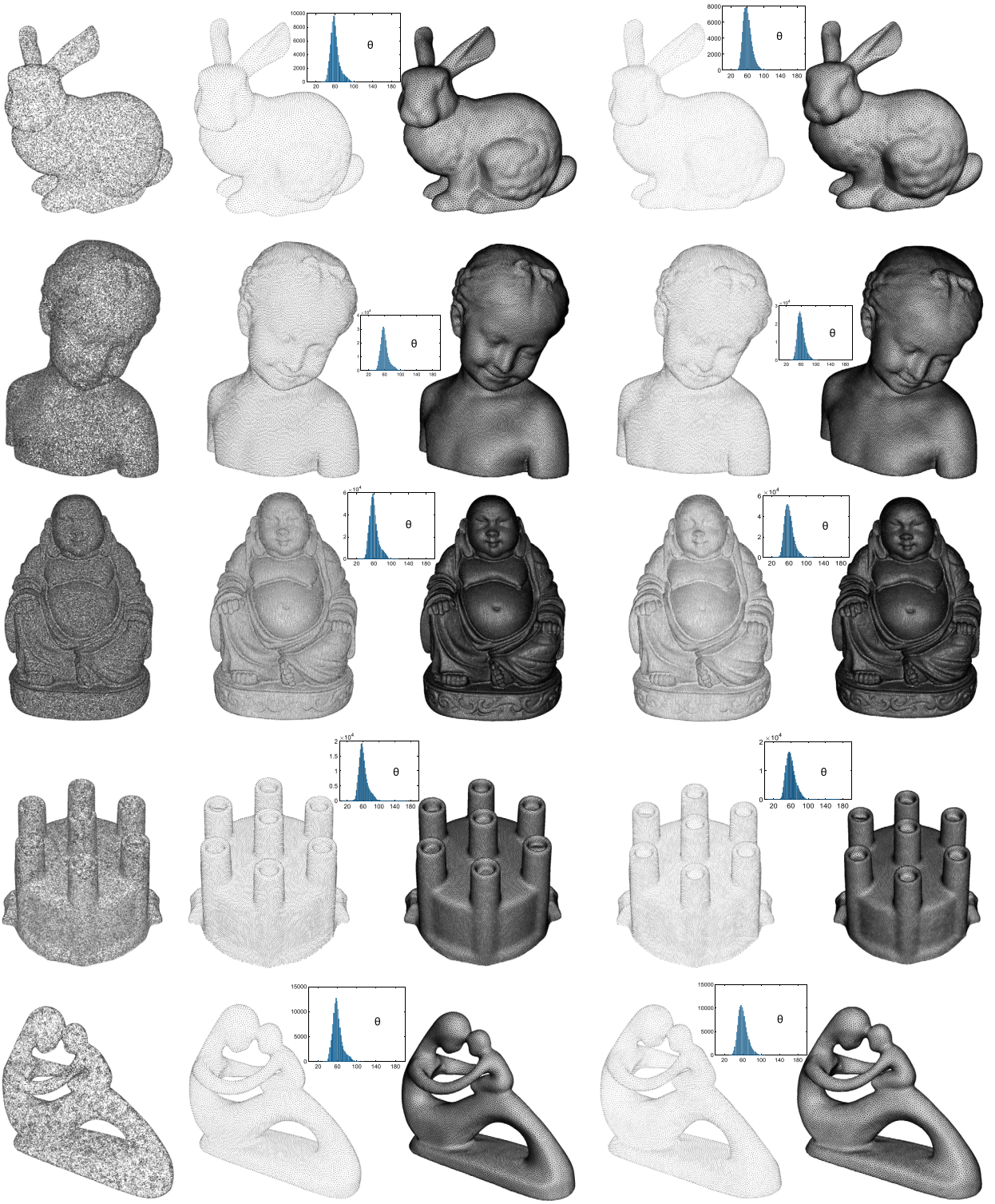

Fig. 7. More comparisons on uniform hexagonal resamplings and isotropic triangular meshes with CVT-based method Chen et al. (2018) and our particle-based method on: Bunny, Bimba, Buddha, Distcap, and Fertility point cloud models.

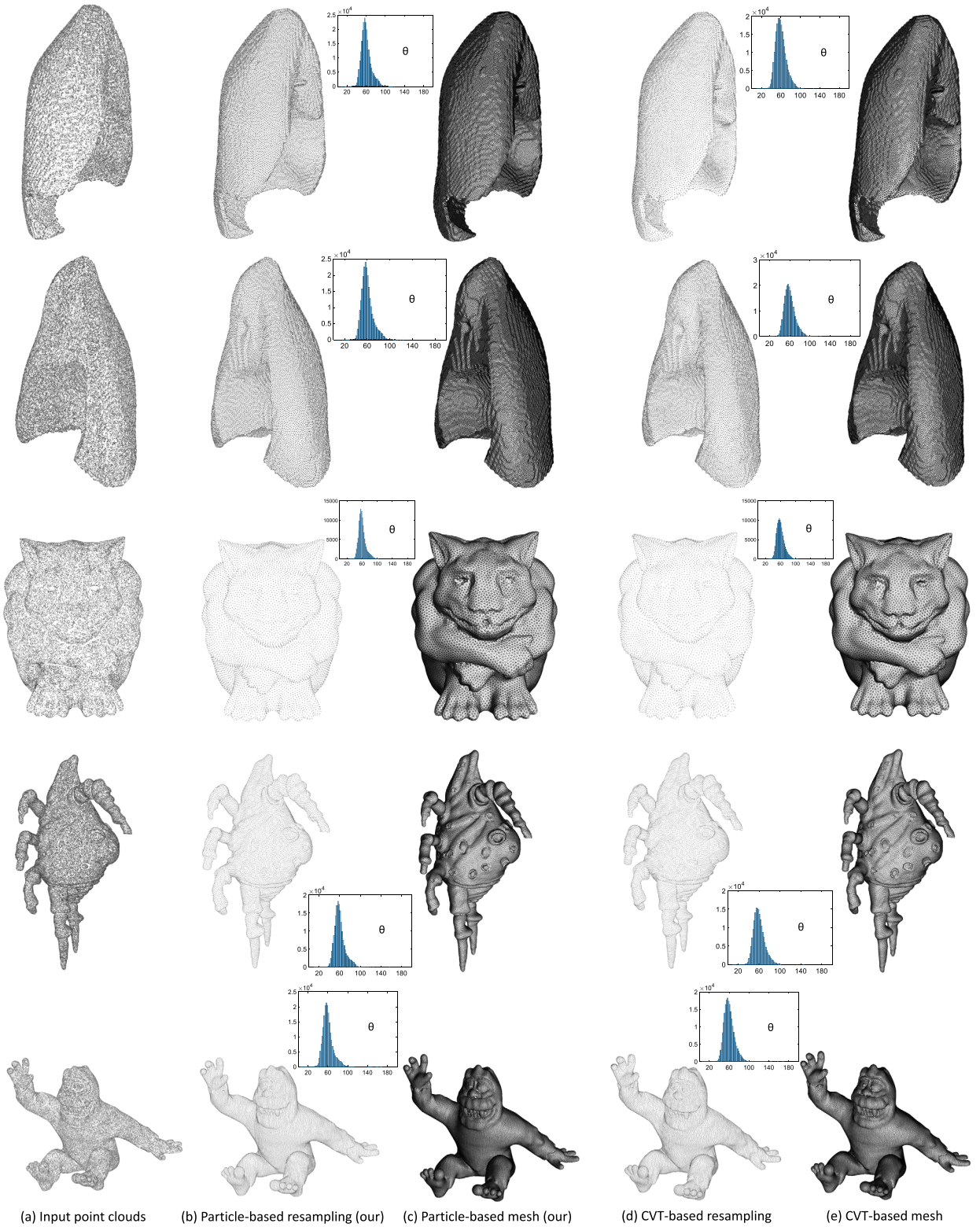

Fig. 8. More comparisons on uniform hexagonal resamplings and isotropic triangular meshes with CVT-based method Chen et al. (2018) and our particle-based method on: Left Lung, Right Lung, GOYLE, Lectroid1, and Mumble Sitting point cloud models.

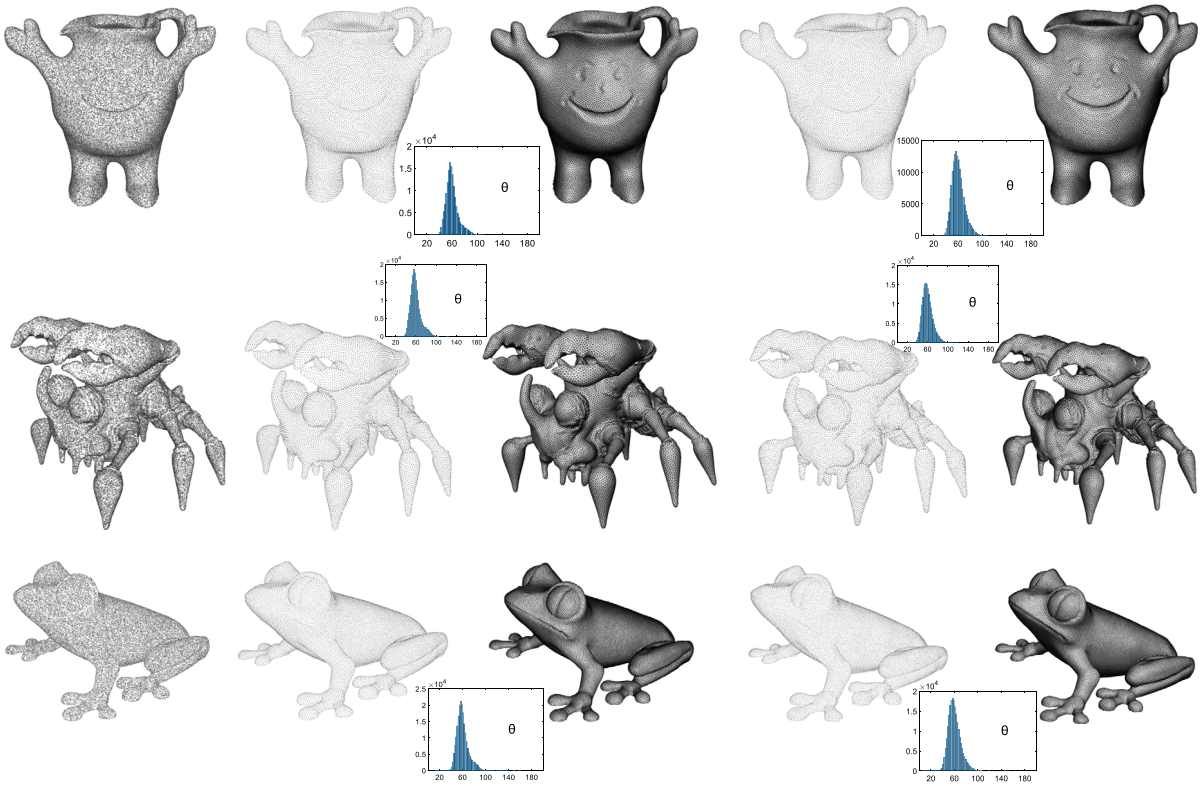

Fig. 9. More comparisons on uniform hexagonal resamplings and isotropic triangular meshes with CVT-based method Chen et al. (2018) and our particle-based method on: Koolaidman, Thundercrab, and Treefrog point cloud models.

7. Discussion and future work

In this work, we have proposed a unified L_p particle-based method to resample the points with different sampling patterns, and generalized the particle-based framework from surface meshes to point clouds, and from the sequential to the parallel. Finally, we can efficiently generate the high-quality isotropic/adaptive/anisotropic hexagonal and quadrilateral samplings and meshes. We can further improve the computational speed by using GPU-based parallel algorithm and implementation, since the current GPUs have thousands of CUDA cores, especially beneficial to the parallel computations. In practice, if the point cloud model has sharp features, such as feature edges or feature corners in CAD models, this requires the user's tagging as an extra input, due to the difficulty of detecting accurate sharp features from the raw point clouds. We will consider these cases in our future work. Moreover, we will also investigate the approaches to further optimize current quad-dominant meshing framework for generating the high-quality all-quad (anisotropic) meshes.

Acknowledgements

We would like to thank the reviewers for their valuable comments. We are grateful to authors of Zhou and Jacobson (2016) and others for sharing 3D model datasets. This work was partially supported by the NSF IIS-1816511, CNS-1647200, OAC-1657364, OAC-1845962, Wayne State University Subaward 4207299A of CNS-1821962, NIH 1R56AG060822-01A1, and ZJNSF LZ16F020002.

Appendix A. Appendix of surface reconstruction by parallel and unified particle-based resampling from point clouds

A.1. More results

A.1.1. Isotropic hexagonal resampling and mesh reconstruction

In order to further demonstrate the better performance of our method compared with CVT-based method Chen et al. (2018), we provide more experiments including different topology genera and point sizes, in addition to results in the paper.

Fig. 7, Fig. 8, and Fig. 9 show the visualization and mesh angle histogram comparisons with the CVT-based method and our method (in L_2-Gaussian kernel) on several point clouds. Table 4 shows the statistics and timings for resampling

Table 4
More comparison results with CVT-based method Chen et al. (2018) and our particle-based method on statistics and timings for resampling computation and surface reconstruction.

Model	Method	#Points	#Samples	G_{min}	G_{avg}	θ_{min}	θ_{avg}	$\%_{<30°}$	T_R (Seq.)	T_M (Seq.)	T_R (Para.)	T_M (Para.)
Bunny	CVT	83, 191	15, 000	0.56	0.88	**32.52°**	51.21°	0	13.85 s	0.20 s	–	–
Bunny	Particle	83, 191	15, 000	**0.58**	**0.90**	31.59°	**52.45°**	0	**5.87** s	0.20 s	**1.03** s	**0.14** s
Bimba	CVT	254, 309	50, 000	**0.57**	0.88	**31.56°**	51.20°	**0**	54.16 s	0.69 s	–	–
Bimba	Particle	254, 309	50, 000	0.49	**0.89**	22.51°	**52.33°**	0.005%	**22.01** s	0.69 s	**4.11** s	**0.45** s
Buddha	CVT	687, 347	100, 000	0.41	0.88	**21.56°**	50.87°	0.023%	91.06 s	1.15 s	–	–
Buddha	Particle	687, 347	100, 000	**0.44**	**0.89**	19.43°	**51.73°**	**0.012%**	**36.75** s	1.15 s	**7.52** s	**0.64** s
Distcap	CVT	208, 578	35, 000	0.02	0.87	0.56°	50.20°	0.28%	24.71 s	0.53 s	–	–
Distcap	Particle	208, 578	35, 000	**0.09**	**0.88**	**5.10°**	**51.31°**	**0.25%**	**11.25** s	0.53 s	**2.11** s	**0.32** s
Fertility	CVT	110, 209	20, 000	**0.55**	0.88	**30.54°**	51.06°	**0**	25.10 s	0.31 s	–	–
Fertility	Particle	110, 209	20, 000	0.44	**0.89**	18.97°	**52.26°**	0.012%	**10.17** s	0.31 s	**1.75** s	**0.18** s
Left Lung	CVT	199, 533	40, 000	0.16	0.87	5.76°	50.40°	0.1%	37.02 s	0.59 s	–	–
Left Lung	Particle	199, 533	40, 000	**0.23**	**0.89**	**8.87°**	**51.88°**	**0.03%**	**15.55** s	0.59 s	**2.94** s	**0.39** s
Right Lung	CVT	191, 471	40, 000	**0.38**	0.88	**19.88°**	50.61°	0.1%	37.77 s	0.52 s	–	–
Right Lung	Particle	191, 471	40, 000	0.32	**0.89**	12.48°	**51.86°**	**0.01%**	**15.62** s	0.52 s	**2.94** s	**0.31** s
GOYLE	CVT	136, 132	20, 000	0.31	0.88	17.44°	50.93°	0.03%	16.45 s	0.40 s	–	–
GOYLE	Particle	136, 132	20, 000	**0.44**	**0.90**	**20.09°**	**52.36°**	**0.02%**	**5.76** s	0.40 s	**1.54** s	**0.26** s
Lectroid1	CVT	195, 639	30, 000	**0.19**	0.88	8.54°	50.84°	0.08%	36.16 s	0.52 s	–	–
Lectroid1	Particle	195, 639	30, 000	0.17	**0.89**	**10.10°**	**51.99°**	**0.06%**	**14.42** s	0.52 s	**3.03** s	**0.37** s
Mumble Sitting	CVT	198, 023	35, 000	0.08	0.88	**5.17°**	50.88°	0.2%	39.89 s	0.59 s	–	–
Mumble Sitting	Particle	198, 023	35, 000	**0.10**	**0.89**	5.00°	**52.01°**	0.2%	**16.44** s	0.59 s	**3.42** s	**0.41** s
Koolaidman	CVT	129, 623	25, 000	**0.46**	0.88	**22.44°**	51.10°	0.01%	28.05 s	0.43 s	–	–
Koolaidman	Particle	129, 623	25, 000	0.26	**0.90**	9.36°	**52.39°**	0.01%	**11.78** s	0.43 s	**2.41** s	**0.31** s
Thundercrab	CVT	178, 540	30, 000	**0.37**	0.88	**18.65°**	50.95°	0.05%	25.87 s	0.48 s	–	–
Thundercrab	Particle	178, 540	30, 000	0.24	**0.89**	12.24°	**52.08°**	**0.03%**	**9.10** s	0.48 s	**2.34** s	**0.31** s
Treefrog	CVT	154, 975	35, 000	0.07	0.88	2.45°	50.86°	0.4%	49.11 s	0.64 s	–	–
Treefrog	Particle	154, 975	35, 000	**0.10**	**0.89**	**5.21°**	**51.88°**	**0.3%**	**19.54** s	0.64 s	**4.06** s	**0.41** s

Note: #Points: the number of points in the input point clouds. #Samples: the number of output samples. T_R (Seq.) and T_R (Para.): timings for resampling computations in sequential and parallel designs with 35 iterations. T_M (Seq.) and T_M (Para.): timings for meshing computations in sequential and parallel designs. The best values are highlighted in bold for each group. It is noted that the timings for meshing computations on CVT and Particle methods are the same, since both of them use Geogram to compute the final mesh.

computation and surface reconstruction with CVT-based method Chen et al. (2018) and our particle-based method. Our results demonstrate that we can consistently yield better sampling, mesh angle, and triangle quality (especially G_{avg}, θ_{avg}), as well as the faster computational speed.

Fig. 10 shows more closeup visualization of the uniform hexagonal resampling and isotropic triangular meshing results on large-size Lucy point cloud model.

A.1.2. Anisotropic resampling and mesh reconstruction

Fig. 11 shows two more visualization results of our method (in anisotropic L_2-Gaussian kernel) on anisotropic resampling and mesh reconstruction on Bunny and GOYLE point cloud models with curvature-based anisotropic metrics. The final mesh angle histograms show that our method has good mesh quality to match the input curvature-based anisotropy. Table 5 shows the statistics and timings for anisotropic resampling computation and surface reconstruction by using our particle-based method.

A.1.3. Quadrilateral resampling and mesh reconstruction

Fig. 12 shows the visualization results of our method on quadrilateral resampling and mesh reconstruction on some more point cloud models by using the proposed L_4-Gaussian kernel. Table 6 shows the statistics and timings for quadrilateral resampling computation and surface reconstruction. We can see that our method can efficiently generate high-quality quad-dominant meshes with high quad-element percentages.

In order to demonstrate the scalability of the proposed method on large-size resampling and meshing of point clouds, Fig. 13 shows the final results with quadrilateral resamplings ranging from 160K to 220K of the original point clouds with more than 1M points. It is noted that our particle-based method is quite efficient on the large-size models, and it takes 66.42 s and 77.33 s totally for resampling and meshing (using three-level multi-scale strategy with 150 iterations) for Arch and Lucy models (in parallel implementation), respectively. The final resampling and mesh quality are quite good as well.

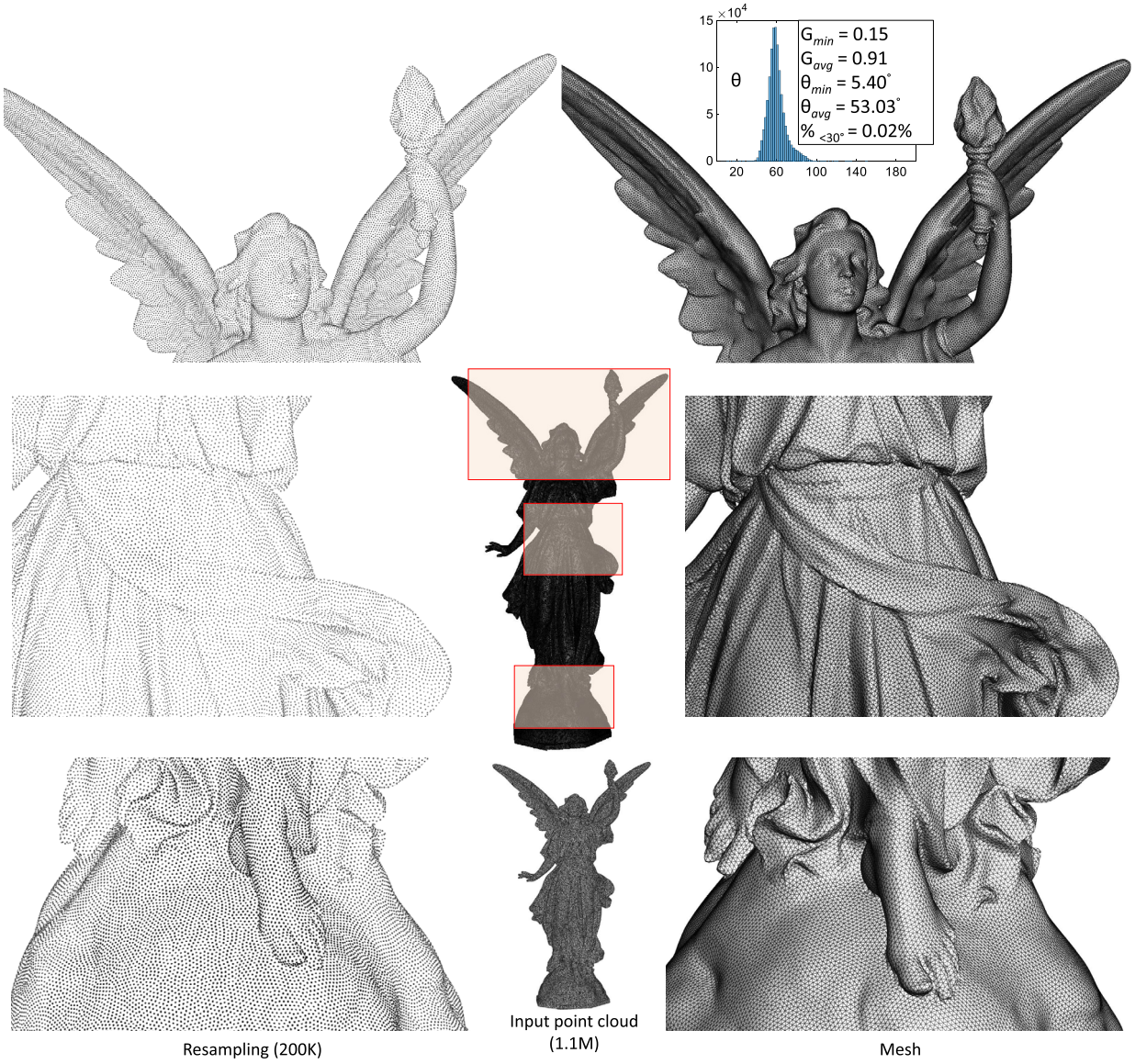

Fig. 10. More closeup visualization of the uniform hexagonal resampling and isotropic triangular meshing results on large-size Lucy point cloud model.

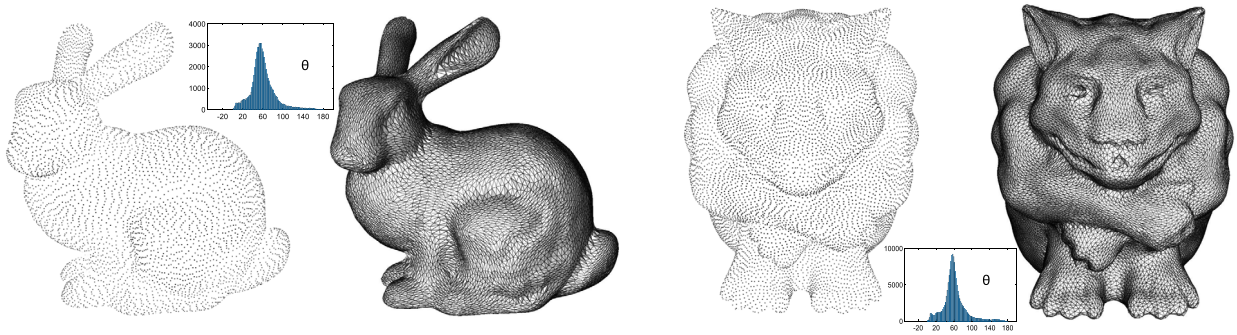

Fig. 11. More anisotropic resampling and anisotropic triangular meshing results on Bunny and GOYLE point cloud models with curvature-based anisotropies.

Table 5
More statistics and timings for anisotropic resampling and reconstruction by our particle-based method.

Model	#Points	#Samples	Stretch	T_R (Para.)	T_M (Para.)
Bunny	124, 800	10, 000	[1, 6.1]	10.90 s	0.28 s
GOYLE	95, 966	25, 000	[1, 5.4]	21.84 s	0.36 s

Note: #Points: the number of points in the input point clouds. #Samples: the number of output samples. T_R (Para.): timings for resampling computation in parallel design with 50 iterations. T_M (Para.): timings for meshing computation in parallel design.

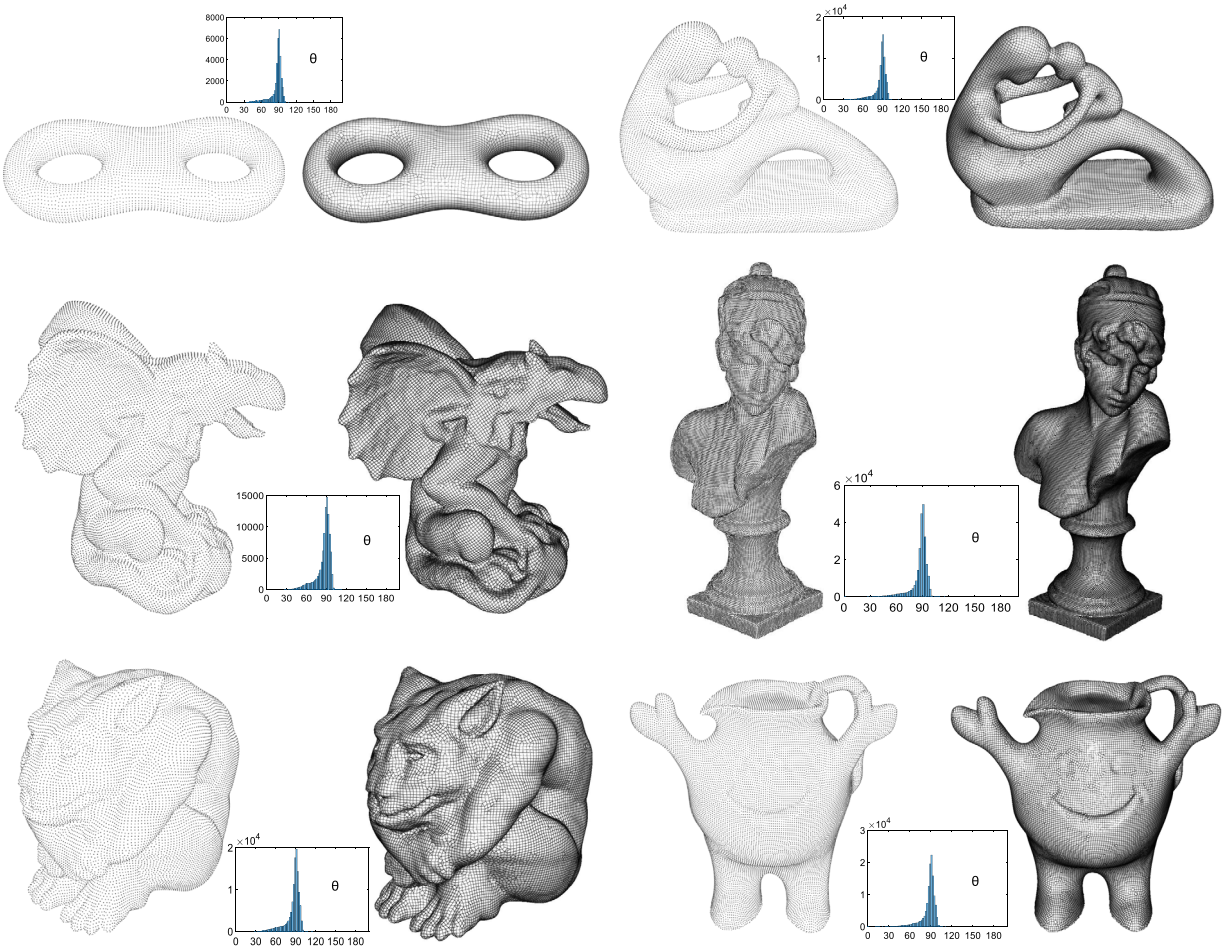

Fig. 12. More quadrilateral resampling and quad-dominant meshing results on point cloud models.

Table 6
More statistics and timings for quadrilateral resampling and reconstruction by our particle-based method.

Model	#Points	#Samples	#Quad%	θ_{min}	θ_{avg}	$\%_{<30^\circ}$	T_R (Para.)	T_M (Para.)	Tri2Quad
Eight	79, 999	8000	92.65%	32.46°	81.57°	0	1.94 s	0.19 s	0.14 s
Fertility	110, 209	20, 000	91.83%	30.08°	80.76°	0	6.01 s	0.25 s	0.29 s
Gargo	90, 898	24, 000	88.29%	23.04°	78.24°	0.01%	5.04 s	0.31 s	0.38 s
Sapphos Head	271, 937	60, 000	92.85%	29.94°	81.56°	0.005%	10.67 s	0.53 s	0.86 s
GOYLE	136, 132	28, 000	89.24%	26.17°	79.22°	0.01%	6.96 s	0.40 s	0.40 s
Koolaidman	129, 623	30, 000	93.27%	31.61°	80.59°	0	6.71 s	0.36 s	0.48 s

Note: #Points: the number of points in the input point clouds. #Samples: the number of output samples. #Quad%: the percentage of quad elements in the output meshes. T_R (Para.): timings for resampling computation in parallel design with 75 iterations (i.e., multi-scale strategy including 40 iterations in the first level and 35 iterations in the second level). T_M (Para.): timings for triangular meshing computation in parallel design. Tri2Quad: timings for merging the triangles into quad-dominant meshes (in sequential).

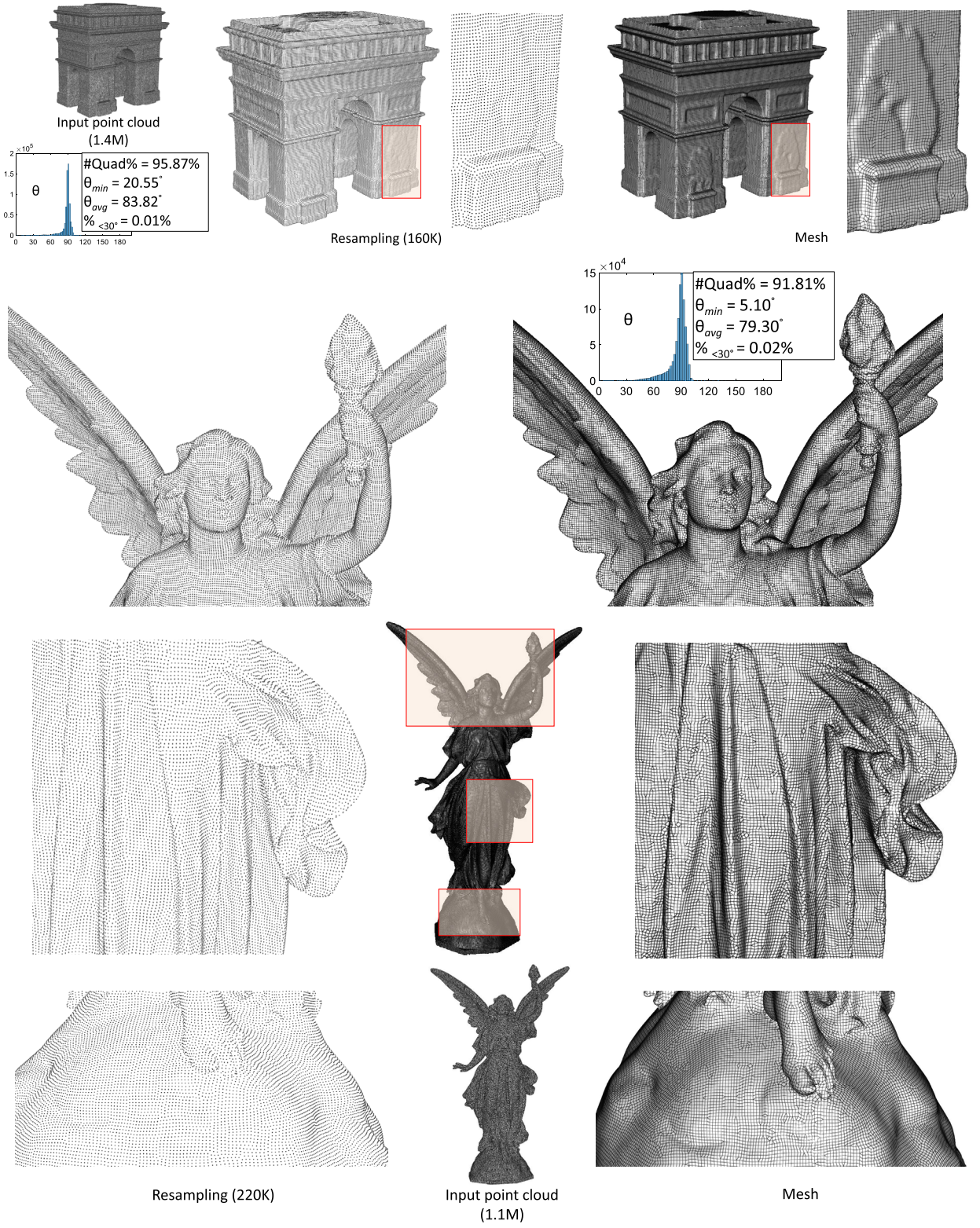

Fig. 13. More quadrilateral resampling and quad-dominant meshing results on large-size Arch and Lucy point cloud models.

References

ALICE/INRIA Nancy, 2018. Geogram. http://alice.loria.fr/software/geogram/doc/html/index.html.

Boltcheva, D., Lévy, B., 2017. Surface reconstruction by computing restricted Voronoi cells in parallel. Comput. Aided Des. 90, 123–134.

Bossen, F., Heckbert, P., 1996. A pliant method for anisotropic mesh generation. In: 5th International Meshing Roundtable, pp. 63–76.

Bronson, J., Levine, J., Whitaker, R., 2012. Particle systems for adaptive, isotropic meshing of CAD models. Eng. Comput. 28, 331–344.

Chen, J., Ge, X., Wei, L., Wang, B., Wang, Y., Wang, H., Fei, Y., Qian, K., Yong, J., Wang, W., 2013. Bilateral blue noise sampling. ACM Trans. Graph. 32, 216:1–216:11.

Chen, Z., Yuan, Z., Choi, Y., Liu, L., Wang, W., 2012. Variational blue noise sampling. IEEE Trans. Vis. Comput. Graph. 18, 1784–1796.

Chen, Z., Zhang, T., Cao, J., Zhang, Y., Wang, C., 2018. Point cloud resampling using centroidal Voronoi tessellation methods. Comput. Aided Des. 102, 12–21.

Dey, T., Ray, T., 2010. Polygonal surface remeshing with Delaunay refinement. Eng. Comput. 26, 289–301.

Du, Q., Faber, V., Gunzburger, M., 1999. Centroidal Voronoi tessellations: applications and algorithms. SIAM Rev. 41, 637–676.

Du, Q., Gunzburger, M., Ju, L., 2003. Constrained centroidal Voronoi tessellations for surfaces. SIAM J. Sci. Comput. 24, 1488–1506.

Du, Q., Wang, D., 2005. Anisotropic centroidal Voronoi tessellations and their applications. SIAM J. Sci. Comput. 26, 737–761.

Edelsbrunner, H., Shah, N., 1994. Triangulating topological spaces. In: Symposium on Computational Geometry, pp. 285–292.

Fei, Y., Rong, G., Wang, B., Wang, W., 2014. Parallel L-BFGS-B algorithm on GPU. Comput. Graph. 40, 1–9.

Frey, P., Borouchaki, H., 1999. Surface mesh quality evaluation. Int. J. Numer. Methods Eng. 45, 101–118.

Fu, X., Liu, Y., Snyder, J., Guo, B., 2014. Anisotropic simplicial meshing using local convex functions. ACM Trans. Graph. 33, 182:1–182:11.

Horn, R., Johnson, C., 1990. Matrix Analysis. Cambridge University Press.

Huang, H., Li, D., Zhang, H., Ascher, U., Cohen-Or, D., 2009. Consolidation of unorganized point clouds for surface reconstruction. ACM Trans. Graph. 28, 176:1–176:7.

Huang, H., Wu, S., Gong, M., Cohen-Or, D., Ascher, U., Zhang, H., 2013. Edge-aware point set resampling. ACM Trans. Graph. 32, 9:1–9:12.

Itoh, T., Shimada, K., 2002. Automatic conversion of triangular meshes into quadrilateral meshes with directionality. Int. J. CAD/CAM 1, 20–38.

Lévy, B., 2016. Robustness and efficiency of geometric programs: the predicate construction kit (PCK). Comput. Aided Des. 72, 3–12.

Lévy, B., Bonneel, N., 2012. Variational anisotropic surface meshing with Voronoi parallel linear enumeration. In: 21st International Meshing Roundtable, pp. 349–366.

Lévy, B., Liu, Y., 2010. L_p centroidal Voronoi tessellation and its applications. ACM Trans. Graph. 29, 119:1–119:11.

Liao, B., Xiao, C., Jin, L., Fu, H., 2013. Efficient feature-preserving local projection operator for geometry reconstruction. Comput. Aided Des. 45, 861–874.

Lipman, Y., Cohen-Or, D., Levin, D., Tal-Ezer, H., 2007. Parameterization-free projection for geometry reconstruction. ACM Trans. Graph. 26, 22:1–22:6.

Liu, D., Nocedal, J., 1989. On the limited memory BFGS method for large scale optimization. Math. Program. 45, 503–528.

Liu, Y., 2010. HLBFGS. https://xueyuhanlang.github.io/software/hlbfgs/.

Liu, Y., Wang, W., Lévy, B., Sun, F., Yan, D., Lu, L., Yang, C., 2009. On centroidal Voronoi tessellation – energy smoothness and fast computation. ACM Trans. Graph. 28, 101:1–101:17.

Lloyd, S., 1982. Least squares quantization in PCM. IEEE Trans. Inf. Theory 28, 129–137.

Luo, C., Ge, X., Wang, Y., 2018. Uniformization and density adaptation for point cloud data via graph Laplacian. In: Computer Graphics Forum, pp. 325–337.

Meyer, M., Georgel, P., Whitaker, R., 2005. Robust particle systems for curvature dependent sampling of implicit surfaces. In: Shape Modeling and Applications, 2005 International Conference, pp. 124–133.

Mount, D., Arya, S., 1998. ANN: Library for Approximate Nearest Neighbour Searching.

Ni, S., Zhong, Z., Huang, J., Wang, W., Guo, X., 2018. Field-aligned and lattice-guided tetrahedral meshing. Comput. Graph. Forum 37, 161–172.

Öztireli, A.C., Alexa, M., Gross, M., 2010. Spectral sampling of manifolds. ACM Trans. Graph., 168:1–168:8.

Preiner, R., Mattausch, O., Arikan, M., Pajarola, R., Wimmer, M., 2014. Continuous projection for fast L_1 reconstruction. ACM Trans. Graph. 33, 47:1–47:13.

Ray, N., Sokolov, D., Lefebvre, S., Lévy, B., 2018. Meshless Voronoi on the GPU. ACM Trans. Graph. 37, 265:1–265:12.

Rong, G., Liu, Y., Wang, W., Yin, X., Gu, D., Guo, X., 2011. GPU-assisted computation of centroidal Voronoi tessellation. IEEE Trans. Vis. Comput. Graph. 17, 345–356.

Rong, G., Tan, T., 2006. Jump flooding in GPU with applications to Voronoi diagram and distance transform. In: Proceedings of the Symposium on Interactive 3D Graphics and Games, pp. 109–116.

Rouxel-Labbé, M., Wintraecken, M., Boissonnat, J.D., 2016. Discretized Riemannian Delaunay triangulations. Proc. Eng. 163, 97–109.

Rusu, R., 2009. Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments. Ph.D. thesis. Computer Science Department, Technische Universitaet Muenchen, Germany.

Rusu, R., Cousins, S., 2011. 3D is here: Point Cloud Library (PCL). In: IEEE International Conference on Robotics and Automation. ICRA, Shanghai, China.

Secord, A., 2002. Weighted Voronoi stippling. In: Proceedings of the 2nd International Symposium on Non-photorealistic Animation and Rendering, pp. 37–43.

Shapiro, P., Martel, H., Villumsen, J., Owen, J., 1996. Adaptive smoothed particle hydrodynamics, with application to cosmology: methodology. Astrophys. J. Suppl. 103, 269–330.

Shimada, K., Gossard, D., 1995. Bubble mesh: automated triangular meshing of non-manifold geometry by sphere packing. In: Proceedings of the Third ACM Symposium on Solid Modeling and Applications, pp. 409–419.

Shimada, K., Yamada, A., Itoh, T., 1997. Anisotropic triangular meshing of parametric surfaces via close packing of ellipsoidal bubbles. In: 6th International Meshing Roundtable, pp. 375–390.

Valette, S., Chassery, J., Prost, R., 2008. Generic remeshing of 3D triangular meshes with metric-dependent discrete Voronoi diagrams. IEEE Trans. Vis. Comput. Graph. 14, 369–381.

Vasconcelos, C., Sá, A., Carvalho, P., Gattass, M., 2008. Lloyd's algorithm on GPU. In: International Symposium on Visual Computing. Springer, pp. 953–964.

Witkin, A., Heckbert, P., 1994. Using particles to sample and control implicit surfaces. In: Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques, pp. 269–277.

Yamakawa, S., Shimada, K., 2000. High quality anisotropic tetrahedral mesh generation via packing ellipsoidal bubbles. In: 9th International Meshing Roundtable, pp. 263–273.

Yan, D., Guo, J., Wang, B., Zhang, X., Wonka, P., 2015. A survey of blue-noise sampling and its applications. J. Comput. Sci. Technol. 30, 439–452.

Yan, D., Lévy, B., Liu, Y., Sun, F., Wang, W., 2009. Isotropic remeshing with fast and exact computation of restricted Voronoi diagram. Comput. Graph. Forum 28, 1445–1454.

Zhong, Z., Guo, X., Wang, W., Lévy, B., Sun, F., Liu, Y., Mao, W., 2013. Particle-based anisotropic surface meshing. ACM Trans. Graph. 32, 99:1–99:14.

Zhong, Z., Hua, J., 2016. Kernel-based adaptive sampling for image reconstruction and meshing. Comput. Aided Geom. Des. 43, 68–81.

Zhong, Z., Shuai, L., Jin, M., Guo, X., 2014. Anisotropic surface meshing with conformal embedding. Graph. Models 76, 468–483.

Zhong, Z., Wang, W., Lévy, B., Hua, J., Guo, X., 2018. Computing a high-dimensional Euclidean embedding from an arbitrary smooth Riemannian metric. ACM Trans. Graph. 37, 62:1–62:16.

Zhou, Q., Jacobson, A., 2016. Thingi10K: a dataset of 10,000 3D-printing models. arXiv preprint. arXiv:1605.04797.