Unbiased Sampling and Meshing of Isosurfaces

Dong-Ming Yan, Johannes Wallner, Peter Wonka

Abstract—In this paper, we present a new technique to generate unbiased samples on isosurfaces. An isosurface, $F(x, y, z) = c$, of a function, F, is implicitly defined by trilinear interpolation of background grid points. The key idea of our approach is that of treating the isosurface within a grid cell as a graph (height) function in one of the three coordinate axis directions, restricted to where the slope is not too high, and integrating / sampling from each of these three. We use this unbiased sampling algorithm for applications in Monte Carlo integration, Poisson-disk sampling, and isosurface meshing.

1 Introduction

Isosurfaces play an important role in many fields, such as medical data processing, scientific visualization, volume rendering, and geometry processing. A three-dimensional (3D) isosurface is implicitly defined as $F(x, y, z) = c$, where c is a constant, called an *isovalue*.

In this paper, we consider isosurfaces given as a 3D discrete volumetric dataset, $I_{n_x \times n_y \times n_z}$, where each sample, $I_{i,j,k}$, is associated with a scalar function value and the function, F, is trilinearly interpolated between samples. This procedure evaluates the value $F(x, y, z)$ by interpolating function values at the eight corners of an axis-aligned box in which the point (x, y, z) is contained.

Since it is difficult to process isosurfaces directly, they are often sampled and converted to triangle meshes before further processing. A major breakthrough in this area was the *marching cubes* (MC) algorithm [1], a simple and elegant method for isosurface extraction. During the last decades, the MC algorithm has been studied extensively and many variations have been proposed to improve the efficiency and the quality of the extracted surfaces, e.g., feature preservation [2], [3], [4], adaptive meshing [5], [6], improved triangulation [7], improved topological consitency [8], etc.

While sampling and meshing are often combined in a single framework, it is interesting to analyze the sampling problem separately. For example, the marching cubes algorithm places samples only on the edges of the sample grid (see Fig. 1), which leads to an undesirable distribution of sample points. For higher-quality sampling, there are generally two strategies: a) optimization based sampling such as centroidal Voronoi tesselation [9], and b) randomized sampling, e.g., maximal Poisson-disk sampling [10]. Both approaches are valuable and have different advantages and disadvantages. The goal of this paper is to contribute to the development of algorithms employing the second of these strategies, i.e., randomized sampling. In this context, one important question to tackle is how to generate a random, unbiased sample on an isosurface. This question does not have a straightforward answer. In this paper, we make the following contributions:

- We propose a new algorithm that randomly chooses samples on an isosurface generated by trilinear interpolation, where "randomly" refers either to a uniform distribution, or to a distribution specified by some density function.
- We build a complete framework around the unbiased sampling algorithm to obtain maximal Poisson-disk sampling and meshing of isosurfaces.

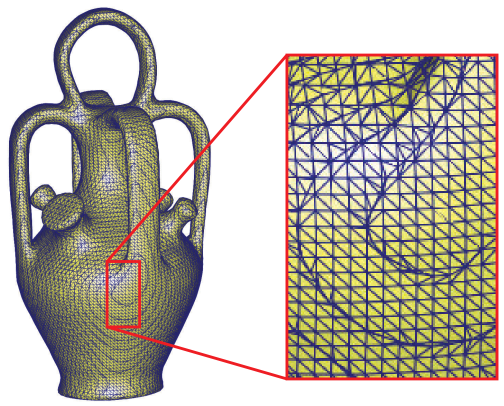


Fig. 1: Sampling pattern of the MC algorithm [1].

2 Related Work

This section reviews the literature on isosurface extraction and Poisson-disk sampling. For more details, we refer the reader to recent comprehensive surveys [11], [10].

- *D.-M. Yan is with KAUST and NLPR, CASIA.*
- *J. Wallner is with TU Graz.*
- *P. Wonka is with KAUST and ASU.*

Isosurface extraction. In their seminal paper, Lorensen and Cline introduced an elegant and robust algorithm for isosurface extraction [1], called Marching Cubes (MC). After that, a huge amount of work investigated polygonalization of implicit surfaces. For example, to resolve the topological ambiguities of the original MC algorithm, Nielson and Hamann proposed the "asymptotic decider" [8]. The accuracy of the MC algorithm was further improved by applying trilinear interpolation [12], [13].

One line of research aims at preserving features of the input data. Kobbelt et al. [2] presented an algorithm to preserve sharp features while extracting surfaces. Ju et al. [3], [4] used the dual grid instead of the primal MC grid for contouring the volumetric data. Schaefer et al. [14] extended dual contouring to generate manifold mesh surfaces.

Another direction of research is to improve the triangle quality of the extracted mesh. Schreiner et al. [5] proposed an active front-based approach to extract isosurfaces from volumetric data. The extracted meshes are near-regular, but artifacts can be observed in the regions where multiple fronts meet. Meyer et al. [6] used particle systems to generate well-distributed points on isosurfaces and to compute their Delaunay triangulation. This led to a significant improvement in the quality of the triangles, with the drawback that the density function estimation used in their work is time consuming. Dietrich et al. [15] introduced an algorithm to analyze the quality of the output of MC. Further, Dietrich et al. [7] proposed to use edge transformations to improve the mesh quality. This approach only slightly changes the original MC algorithm.

Surface sampling. Liu et al. [16] propose a quasi-Monte Carlo approach using integral geometry for computing the area of point-sampled surfaces. Counting the number of intersection points of random lines with the surface, the surface area is estimated by the Cauchy-Crofton formula. Detwiler et al. [17] propose a generic surface sampler for Monte Carlo simulations based on line-surface intersection. However, the line-surface intersection is somehow time consuming.

Poisson-disk sampling. High-quality sampling is a fundamental research problem in both computer graphics and geometry processing. One of the most popular sampling techniques is Poisson-disk sampling, because it exhibits nice properties in both the geometry domain and the frequency domain. Since Cook [18] introduced the classic Poisson-disk sampling algorithm, called "dart throwing", a lot of work has been conducted to generalize and accelerate this algorithm. Poisson-disk sampling has also been generalized to surfaces [19], [20], [21], [22]. More recently, many techniques have been proposed for *maximal Poisson-disk sampling* (MPS), which plays an important role in the quality of the triangulation of a Poisson-disk set [23]. The key to computing MPS is to locate the uncovered regions in an already existing sampling set, e.g., using boundary tracing [24], Voronoi diagrams [25], spatial subdivisions using axis-aligned boxes [26], [27], [28], [29], and regular triangulations [30], [31]. The most recent work of Yan and Wonka [30], [31] generalizes MPS to surfaces and presents a remeshing framework that can guarantee a lower bound of the minimal angle. This property is crucial to many applications, such as finite element simulation [32]. However, to the best of our knowledge, there is no published work that deals with Poisson-disk sampling of isosurfaces directly. In this paper, we show how to adapt a spatial subdivision technique for MPS on isosurfaces.

3 Overview

In the following sections, we first address the problem of generating unbiased uniform or adaptive samples on isosurfaces (Section 4), based on a careful analysis of the trilinear interpolation of a 3D scalar-valued implicit function. Then, we show several applications that can benefit from the presented sampling technique, including Monte Carlo sampling, Poisson-disk sampling, and meshing of isosurfaces (Section 5).

4 Sampling on Isosurfaces

This section describes how to choose a random point on an isosurface, $F(x, y, z) = c$, of a function F that is defined by trilinear interpolation. Here, the meaning of *random* is either that of uniform distribution with respect to the surface's area measure, or, alternatively distribution with respect to to a certain density function (if that density is a constant, then we again obtain uniform distribution with respect to surface area). Our algorithm is based on the availability of random numbers uniformly distributed in an interval, and the fact that the trilinear interpolation can be locally represented by a graph with respect to a reference plane. It consists of the following steps:

1) Set up a way of randomly choosing a point on surfaces that are the graphs of functions, $z = f(x, y)$. Here "random" refers either to uniform distribution, or to distribution with respect to some density function. It is well known that this task can be solved by rejection sampling in the parameter domain [33].
2) Locally represent the isosurface as a graph surface (over the xy plane, or the yz plane, or the xz plane), taking care that the inclination of the isosurface with respect to the reference plane does not exceed a certain threshold.
3) A random point on the isosurface is now chosen by first randomly choosing one of the local graph representations of step 2) and then applying step 1) to it.

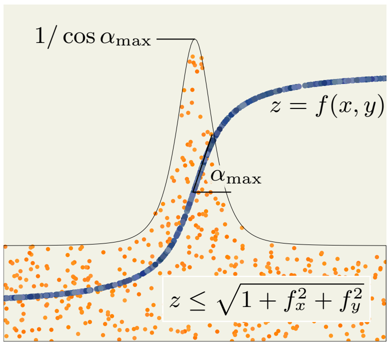


Fig. 2: The uniform distribution of points (x, y, z) (blue) on a graph surface, $z = f(x, y)$, corresponds to the distribution of points (x, y) with respect to the density function, $\delta(x,y) = \sqrt{1 + f_x^2 + f_y^2}$. We simulate the latter via uniform distribution of auxiliary points (red) in the spatial domain $z < \delta(x, y)$, and forgetting the z coordinate.

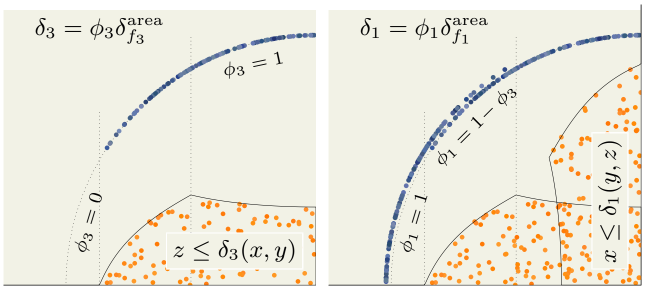


Fig. 3: *Left:* Weighted sampling of a surface, $\Psi : z = f_3(x, y)$, such that the density, ϕ_3, vanishes in the steep part of Ψ. *Right:* Ψ is also represented as $x = f_1(y, z)$. We perform sampling of Ψ using the density function $\phi_1(x, y, z) = 1 - \phi_3(x, y, z)$. Superposition of the two samplings yields a uniform distribution.

4.1 Rejection sampling

The first task is solved by the following well-known observation: choosing a point (x, y) in a domain D randomly according to the density $\delta(x, y)$ is equivalent to choosing a point (x, y, z) in the spatial domain

$$\widetilde{D} = \{(x, y, z) \mid (x, y) \in D,\ 0 \le z \le \delta(x, y)\}$$

randomly according to the uniform distribution, and subsequently forgetting the z coordinate. This taking of marginals is implemented by finding a bounding box, $\mathrm{BB} = [a_1, b_1] \times [a_2, b_2]$, for D, and an upper bound, $\delta_{\max}$, for the density. We now choose random numbers $x \in [a_1, b_1]$, $y \in [a_2, b_2]$ and $z \in [0, \delta_{\max}]$, and repeat this process until (x, y, z) happens to be contained in $\widetilde{D}$. Obviously, the numerical performance of this method depends on the tightness of both the bounding box and the density bound.

The *success rate* of one pass is the volume of the set of acceptable picks (x, y, z), divided by the volume of the box we pick (x, y, z) from:

$$r = \frac{\mathrm{vol}(\widetilde{D})}{\mathrm{area}(\mathrm{BB})\delta_{\max}} = \frac{1}{\mathrm{area}(\mathrm{BB})} \int \frac{\delta}{\delta_{\max}}.$$

The average number of iterations until a random pick (x, y, z) is successful then equals $1/r$.

Normalization of density functions. The outcome of rejection sampling remains the same if δ is multiplied with an arbitrary factor (and $\delta_{\max}$ is adjusted accordingly). A natural way of normalizing δ is to make it a probability density whose integral is 1. Another normalization is to make $\int \delta$ equal r, by multiplying both δ and $\delta_{\max}$ by $r \,/\, \int \delta$. The latter normalization also leads to a probability density, namely the one modeling one pass of rejection sampling where with probability $1 - r$ no point is generated.

4.2 Sampling of graph surfaces

The surface area element of the graph surface, Ψ, with equation $z = f(x, y)$ and the corresponding area element "$dxdy$" in the xy plane are related by the cosine of the *slope angle* $\alpha(x, y)$ between the surface and the plane:

$$dA = \delta_f^{\mathrm{area}}(x, y)\, dxdy, \text{ where} \tag{1}$$

$$\delta_f^{\mathrm{area}} = \frac{1}{\cos\alpha} = \sqrt{1 + f_x^2 + f_y^2} \tag{2}$$

(f_x and f_y are the partial derivatives of f). The equality in (2) is verified by an elementary computation. Summing up, the uniform distribution of points $(x, y, f(x, y))$ in Ψ directly corresponds to the distribution of points (x, y) according to the density, δ_f^{area}. It is therefore not difficult to uniformly sample Ψ: we simply choose $(x, y) \in D$ according to the density defined in (1). Figure 2 illustrates this procedure.

If we wish to sample the surface not in a uniform manner but according to a density, $\phi(x, y, z)$, we can use the same procedure, but with the density function

$$\delta(x, y) = \phi(x, y, f(x, y)) \cdot \delta_f^{\mathrm{area}}(x, y) \tag{3}$$

instead. Rejection sampling has the success rate

$$r = \frac{\int_D \phi\, \delta_f^{\mathrm{area}}}{\mathrm{area}(\mathrm{BB})\delta_{\max}} = \frac{\int_\Psi \phi}{\mathrm{area}(\mathrm{BB})\delta_{\max}}\,. \tag{4}$$

Unfortunately even very simple surfaces can lead to unbounded densities, making rejection sampling unusable in practice. There are, however, special surfaces that are graph surfaces with respect to *all three* coordinate axes, and for which we can escape this predicament – by locally choosing that coordinate plane as a base plane with respect to which the surface enjoys the gentlest slope.

We give a general algorithm for random sampling on such a special surface, which is based on three auxiliary densities, ϕ_1, ϕ_2, ϕ_3, summing up to 1. The procedure is illustrated in Figure 3 in a schematic way (with $\phi_2 = 0$). This algorithm uses the function γ defined by

$$\gamma(x) = \begin{cases} 0 & \text{if} \quad x < n^* \\ 1 & \text{if} \quad x \ge n^* \end{cases} \qquad n^* = \frac{1}{\sqrt{3}}$$

(actually all we need is that $\gamma(x) > 0$ if $x \geq 1/\sqrt{3}$ and $\gamma(x) = 0$ for some value $x < n^*$, where $n^* \leq 1/\sqrt{3}$).

Algorithm 1. *(Uniform random sampling on a surface, Ψ)*

(i) Determine graph representations $z = f_3(x, y)$, $x = f_1(y, z)$, $y = f_2(x, z)$ of the surface, Ψ, and the respective domains, D_i, bounding boxes BB_i and densities $\delta^{\mathrm{area}}_{f_i}$.

(ii) Evaluate the density function ϕ_i ($i = 1, 2, 3$) in a point of the surface Ψ that uses the unit normal vector (n_1, n_2, n_3) in that point: we let

$$\phi_i = \frac{\gamma(|n_i|)}{\gamma(|n_1|) + \gamma(|n_2|) + \gamma(|n_3|)}.$$

(iii) Find an upper bound, $\delta_{i,\max}$, of the function $\delta_i = \delta^{\mathrm{area}}_{f_i} \cdot \phi_i$. One may always use $\delta_{i,\max} = 1/n^*$.

(iv) Choose an index $i \in \{1, 2, 3\}$ randomly, such that the probability of choosing index j equals

$$\frac{w_j}{w_1 + w_2 + w_3}, \text{ where } w_j = \mathrm{area}(\mathrm{BB}_j)\, \delta_{j,\max}. \quad (5)$$

(v) Perform one round of rejection sampling in D_i with density $\phi_i \delta^{\mathrm{area}}_{f_i}$ in order to obtain a point on the graph of the function f_i (which is Ψ). Iterate from (iv) until success.

Well-definedness of Algorithm 1. In the notation employed by Algorithm 1, $n_1^2 + n_2^2 + n_3^2 = 1$ implies that at least one coordinate, n_i, exceeds $1/\sqrt{3}$, so ϕ_i has a nonzero denominator and is well defined. In any point of the surface we have either $|n_i| < n^*$ (so $\phi_i = 0$ there), or $|n_i| \geq n^*$, so the slope angle with respect to the base plane does not exceed $\arccos(n^*)$ and the value of $\delta^{\mathrm{area}}_{f_i}$ does not exceed $1/n^*$. Since $\phi \leq 1$, we have $\delta_i \leq 1/n^*$ as stated.

Correctness of Algorithm 1. Rejection sampling with base domain D_i generates points distributed with density ϕ_i. From Equations (5) and (4), we derive the success rate, $r_i = \int_\Psi \phi_i / w_i$, so the point generated by one pass of rejection sampling is distributed according to the probability density, $\phi_i r_i \,/\, \int \phi_i = \phi_i / w_i$. Since D_i is chosen with probability $\frac{w_i}{w_1+w_2+w_3}$, the total probability density in Ψ of the point generated by one pass of the algorithm is

$$\sum_{i=1}^{3} \frac{w_i}{\sum w_j} \frac{\phi_i}{w_i} = \frac{\phi_1 + \phi_2 + \phi_3}{\sum w_j} = \frac{1}{\sum w_j}. \quad (6)$$

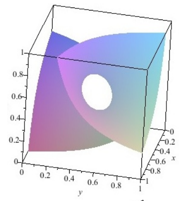
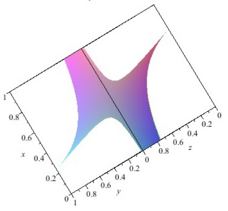
It is a constant function, implying correctness. The success rate of one pass is found by integration:

$$r = \int_\Psi \frac{1}{\sum w_j} = \frac{\mathrm{area}(\Psi)}{\sum_j \mathrm{area}(\mathrm{BB}_j)\delta_{j,\max}} = \frac{n^*\, \mathrm{area}(\Psi)}{\sum_j \mathrm{area}(\mathrm{BB}_j)}.$$

Non-uniform sampling in surfaces. Note that sampling guided by some density function, ϕ, can be achieved by simply replacing ϕ_1, ϕ_2, ϕ_3 by the products $\phi\phi_1, \phi\phi_2, \phi\phi_3$, respectively. We assume, without loss of generality, that ϕ has values in $[0, 1]$, so there is no need to adjust the density bounds, $\delta_{i,\max}$, in the algorithm (unless, of course, it is possible to increase the efficiency of the algorithm if a tighter bound can be found). The success rate of one pass of sampling according to density ϕ is given by

$$r = \int_\Psi \frac{\phi}{\sum w_j} = \frac{\int_\Psi \phi}{\sum_j \mathrm{area}(\mathrm{BB}_j)\delta_{j,\max}} = \frac{n^* \int_\Psi \phi}{\sum_j \mathrm{area}(\mathrm{BB}_j)}.$$

4.3 Sampling of isosurfaces

We cannot apply §4.2 to arbitrary isosurfaces. However, any *trilinear interpolant*, $F(x, y, z)$, defined by function values on a grid has the property that, within a grid cell, its isosurfaces are graphs over each of the three coordinate planes. This statement is perhaps unexpected, since such isosurfaces may exhibit features like tunnels (see inset) or pinch points, but is proven directly from construction of the trilinear interpolant: Within a grid cell, F is a linear function in each variable separately, e.g., $F(x, y, z) = \alpha(x, y) + z \cdot \beta(x, y)$. Consequently, the level set $\{F = c\}$ is the graph of the function $z = f_3(x, y) = \frac{c - \alpha(x,y)}{\beta(x,y)}$. Similarly, we get functions $f_1(y, z)$ and $f_2(x, z)$. There are, however, the following limitations:

- If the eight values of F in the vertices of a grid cell equal the same constant, c, then the entire grid cell is part of the level set, $\{F = c\}$. We must exclude such parts of level sets, since they are not surfaces, and the concept of uniform distribution with respect to the area measure breaks down.
- The denominator of f_i might be zero, e.g., if the level set contains a line parallel to the i-th coordinate axis. This phenomenon does not disturb Algorithm 1: The probability of randomly hitting a singularity of the level set is zero, whereas Algorithm 1 deals with regular areas of the surface by means of those graph representations whose slope does not exceed a certain threshold.

The above-mentioned graph properties of trilinear interpolants lead to the following procedure for sampling isosurfaces of trilinear interpolants according to a given density function $\phi(x, y, z)$:

Algorithm 2. *Sampling of a trilinear interpolant isosurface, Ψ, according to a density function, ϕ, defined on Ψ.*

(i) For each grid cell C intersecting Ψ, prepare the data needed for Algorithm 1: graph representations, $f_{i,C}$, of the surface, $\Psi \cap C$, bounding boxes, $\mathrm{BB}_{i,C}$, and product densities, $\delta^{\mathrm{area}}_{f_{i,C}}\, \phi\, \phi_{i,C}$.

(ii) Randomly choose a pair (i, C), $i \in \{1, 2, 3\}$, such that the probability of picking that pair is proportional to $\mathrm{area}(\mathrm{BB}_{i,C})$.

(iii) *Perform one pass of rejection sampling within cell C, representing the isosurface as graph of the respective function, $f_{i,C}$. Iterate from (ii) until success.*

Correctness of Algorithm 2. The correctness proof is a direct continuation of the respective proof for Algorithm 1: Equation (6) still applies, only $\sum w_j$ is replaced by $\sum_{i,j} w_{C,j}$. Correctness follows from the fact that the point generated by one pass enjoys a constant density independent of the choice of the cell.

Success rates of Algorithm 2. In order to compute the success rate of one pass, we assume that $1/n^*$ is the density upper bound (in the notation of Algorithm 1). Averaging individual success rates yields

$$r = \sum_C \left(\frac{\sum_i \text{area}(\text{BB}_{i,C})}{\sum_{i,C} \text{area}(\text{BB}_{i,C})} \right) \frac{n^* \int_{\Psi \cap C} \phi}{\sum_i \text{area}(\text{BB}_{i,C})}$$
$$= \frac{n^*}{\sum_{i,C} \text{area}(\text{BB}_{i,C})} \sum_C \int_{\Psi \cap C} \phi = \frac{n^*}{\sum_{i,C} \text{area}(\text{BB}_{i,C})} \int_\Psi \phi.$$

If one does not want to implement optimal bounding boxes for the individual graph representations, then of course the entire face of a grid cell may be used as a bounding box. If in addition grid cells are cubes, step *(ii)* of Algorithm 2 is considerably simplified: We may choose a cell C and an integer $i \in \{1, 2, 3\}$ independently, according to uniform distribution.

5 Applications

In this section, we present three applications of the proposed isosurface sampling technique: Monte Carlo integration, Poisson-disk sampling, and meshing of isosurfaces. All the experimental results were conducted on an Intel X5680 with a 3.33GHz CPU, 4GB memory and a 64-bit Windows 7 operating system.

5.1 Monte Carlo integration

The first application we would like to show is that the sampling algorithm presented in Section 4.3 is useful for Monte-Carlo integration (e.g., for global illumination). In the following examples, the probability density that guides the sampling is constant; the sampling is also uniform. These tests are the most straightforward application of our algorithm, because we generate independent, uniformly distributed (i.e., un-biased) samples on the isosurface. Figure 4 shows the result of sampling 200,000 points on a piece of isosurface inside a unit cube. However, we do not attempt to uniformly sample the boundary of the isosurface. Figure 5 illustrates sampling on a sphere, which is implicitly defined as $F(x, y, z) = x^2+y^2+z^2-1$. In this example, we use a sampling grid with size 0.01 to sample the function into a regular grid. For each grid cell, we try to generate a sample 100 times (several of the tries will be rejected). The color coding indicates the number of neighboring points of each sample within a fixed distance. The histograms in the figures show that our sampling results are uniformly distributed. We can also determine the efficiency of our method by computing the acceptance rate of the samples. We can classify two reasons for rejecting a sample. First, a sample can be generated on a part of a graph that is outside of the grid cell (56.6%). Second, the sample can be rejected based on rejection sampling (19.9%). Finally, 23.5% of the samples are accepted leading to an efficient algorithm. The speed of uniform sampling is about 1M accepted points/sec.

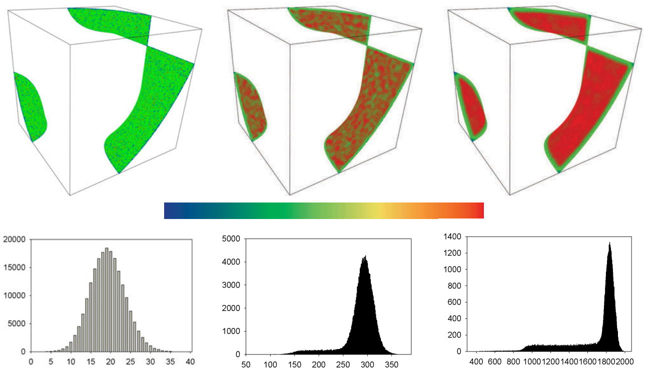


Fig. 4: Uniform sampling inside a unit cube. The isovalues at the vertices are {-3, 3, 1, -1, 2, -1, 12, 1}. Left: d=0.005; middle: d=0.02; right: d=0.05. The color-coding indicates the number of neighboring samples inside a sphere centered at each sample point, where green indicates a smaller value and red indicates a larger value. The histograms show the distribution of neighborhood sizes where neighborhood size is defined by the number of points inside it. The flat regions in the histogram stem from the points on the boundaries.

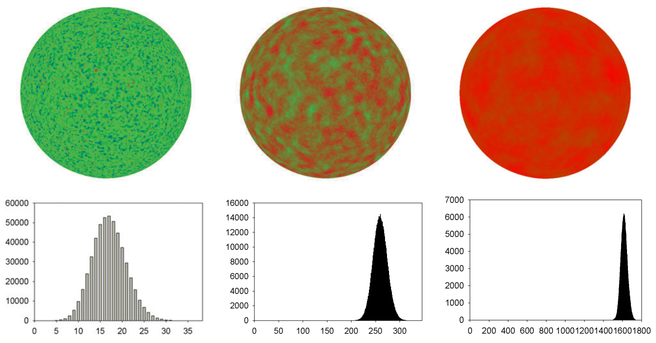


Fig. 5: Uniform sampling on a sphere. From left to right: The neighborhood sizes of the analysis are 0.005, 0.02, and 0.05. See the caption of Fig. 4 for an explanation.

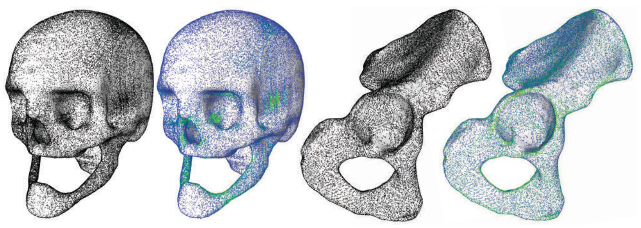


Fig. 6: Adaptive sampling of isosurfaces. The color-coding indicates the curvatures.

Our sampling algorithm also works well for the non-uniform case. In our tests, we estimate the maximal curvature at the sample point as the density function. Figure 6 shows the results of adaptive sampling.

5.2 Poisson-disk sampling

Our sampling algorithm can also be used for Poisson-disk sampling. The goal of Poisson-disk sampling is the un-biased distribution of a set of points on the surface, such that each pair of points has a distance greater than a predefined value. This value can be constant for the complete point set (in which case it is called the sampling radius) or it can be location dependent (thereby controlling the local density of the sampling). We use Euclidean distances in our framework for simplicity and efficiency.

The input of our system is a 3D volumetric dataset, I, together with an isovalue, c, i.e., an isosurface. The resolution of the input 3D image is $n_x \times n_y \times n_z$ voxels, and the width, height, and depth of I are denoted by l_x, l_y, and l_z, respectively. We assume that the longest edge of the input 3D image is normalized to have length one. The user input consists of a sampling radius, r, the cutoff sampling radius, $\lambda\, r$ for the maximal radius (typically we use $\lambda = 8$ for adaptive sampling), and a density function that indicates the sampling radius at a given sample point for adaptive sampling. The cutoff sampling radius is the largest allowable radius. If the density function requests a larger radius, then the cutoff radius will be used instead.

We propose a framework for maximal Poisson-disk sampling on isosurfaces, which consists of the three steps *initialization*, *initial sampling*, and *gap filling*, see Figure 7(b-e). In the following, we first explain the details of the framework for uniform sampling. Then we extend the framework to adaptive maximal Poisson-disk sampling.

5.2.1 Uniform MPS

Initialization. We first build a uniform grid, G, for accelerating the Poisson-disk sampling [26]. The grid size is set to $\frac{r}{\sqrt{3}}$, such that each grid cell can contain at most one sample point. We only need to store the grid cells that intersect the isosurface, "$F(x, y, z) = c$". Those cells are called boundary cells. Each grid cell stores a flag to indicate whether or not the cell is occupied. This flag is initialized as *false*. Further, each boundary cell stores the values of F at the eight corners of the cell. The values inside the cell are computed by trilinear interpolation of the input 3D image, I, at the grid positions. Note that for a boundary cell, the eight function values at vertices cannot have the same sign. The grid, G, is also used for conflict checking during the Poisson-disk sampling.

Initial sampling. Once the boundary cells of the grid are extracted, we store them in a flat array and we preform rejection sampling as described in Section 4.1 on the pieces of the isosurface contained in the boundary cells. Each time a boundary cell is randomly chosen, a sample point inside the boundary is randomly generated using the method presented in Section 4.3. If the newly generated sample is inside the boundary cell, we perform the conflict check with its $5 \times 5 \times 5$ neighboring boundary cells. If the new dart does not contain any other existing sampling and vice versa, the dart is accepted and the occupied flag of the containing boundary cell is marked as *true*. This initial sampling is repeatedly performed until a consecutive number of rejections is observed (e.g., 300 in our implementation). Note that there are other choices of the conflict checking that can affect the angles of the meshing [34].

Gap filling. The initial sampling results in a non-maximal sampling of the isosurface. As shown in [35], [31], the maximal sampling has many nice properties such as angle bounds of the extracted triangle mesh. Hence, we perform another gap-filling step to achieve maximality. We adapt the technique proposed by Ebeida et al. [29] for this purpose. We first update the flat array by subdividing the boundary cells that are not fully covered. We call a subdivided cell a fragment. If a fragment is still not fully covered, we keep it in the flat array; otherwise, the fragment is discarded. Then, we perform dart throwing against the flat array of the fragments. The fragment updating step and the dart throwing step are repeated until there are no more uncovered fragments. While this process is limited by machine precision, we observed convergence in 12-15 iterations in our examples (see Figure 8).

5.2.2 Adaptive MPS

The framework for adaptive MPS is quite similar to that for uniform MPS. The only difference is that in the initial sampling stage, each time when a new sample, $\mathbf{x}$, is generated, we first evaluate the sampling radius, $r(\mathbf{x})$, based on the input density function. The radius is bounded between $[r, \lambda\, r]$ in our experiments. Then, we perform the conflict check using the subgrid that is covered by the sphere, $(\mathbf{x}, r(\mathbf{x}))$, of the new sample point. If the new sample is covered by other spheres centered at existing samples or it covers other existing samples, the new sample is rejected; otherwise, it is accepted. Note that the resolution of the subgrid used for conflict checking is much larger than that used for uniform sampling, which slows down the performance of the adaptive sampling. The readers are referred to [31] for more details on adaptive MPS.

5.2.3 Performance analysis

We analyze the efficiency of our approach. Given the normalized input data set (as well as the isovalue) and a sampling radius, r, we measure the speed of generating valid points on the isosurfaces, and the

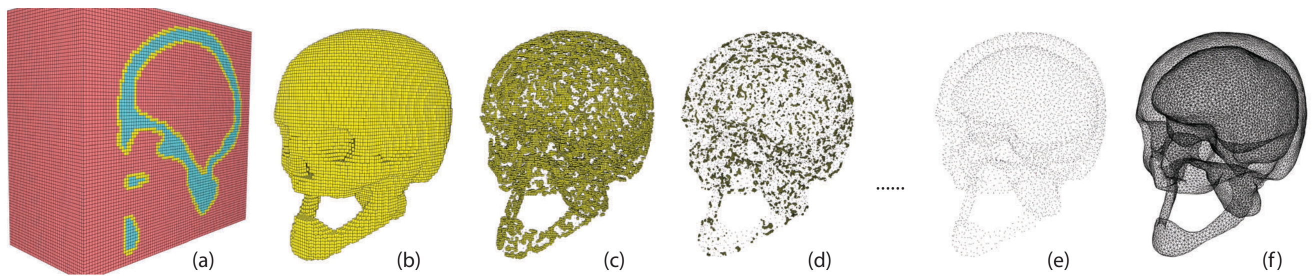


Fig. 7: Main steps of the isosurface sampling/meshing framework. (a) A uniform grid is built from the input 3D image. The cells are classified as inside (cyan), outside (red) and boundary (yellow); (b) initial uncovered fragments (boundary cells); (c) fragments after one iteration of dart throwing and subdivision; (d) fragments after two steps of repetition; (e) final maximal sampling result; (f) meshing result from the sample points.

speed of the accepted points by Poisson-disk sampling. Table 1 lists those statistics. We can see that about 66.7% of the randomly generated points are outside the box, and the acceptance ratio decreases when the sampling radius decreases. We can observe that Poisson-disk sampling has a significantly lower acceptance ratio than the uniform sampling procedure discussed in Section 5.1. This is partly due to the fact that the efficiency of Poisson-disk sampling decreases over time, as it is difficult to find small gaps. However, the sampling efficiency is similar to other state-of-the-art Poisson-disk sampling algorithms.

Model	r	#V	oob%	rej_1%	acc%	rej_2%	speed
Skull	0.010	15.9k	66.68	14.35	1.65	16.38	5.7k/s
	0.008	24.4k	66.66	14.60	1.11	16.04	4.4k/s
	0.006	43.3k	66.62	14.48	0.64	16.12	3.3k/s
	0.005	63.0k	66.60	14.46	0.42	16.29	2.8k/s
Botijo	0.010	6.0k	66.66	14.36	4.05	16.54	7.9k/s
	0.008	9.4k	66.70	14.28	2.69	16.49	6.5k/s
	0.006	16.4k	66.58	14.48	1.68	16.17	4.7k/s
	0.005	23.4k	66.70	14.37	1.26	16.00	4.0k/s

TABLE 1: Performance analysis of Poisson-disk sampling. r is the sampling radius. #V is the total number of generated samples. *oob* is the out-of-box ratio when attempting to generate a random point inside a box; *rej_1* is the rejection ratio of generating a random point inside a box; *acc* means the final accepted point by Poisson-disk sampling over the total generated darts, and *rej_2* means a rejection of the Poisson-disk sampling over the total generated points. The *speed* is measured in points per second.

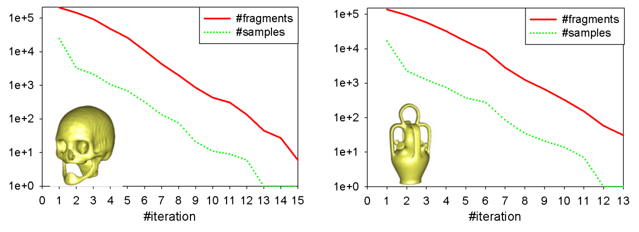


Fig. 8: Convergence of the gap-filling step. Left: Skull (r=0.008); right: Botijo (r=0.006).

Convergence of the gap-filling procedure. The examples of Figure 8 illustrate that the gap-filling step converges in practice.

Blue-noise properties. Since the standard Fourier analysis cannot be directly applied to point sets on surfaces, Wei and Wang [36] introduced the *Differential Domain Analysis* (DDA) technique. We use this technique to measure the blue-noise properties of our results. One example is shown in Fig. 9. The spectrum properties are very similar to that of the original paper [36], which means that our sampling results exhibit the typical blue-noise properties.

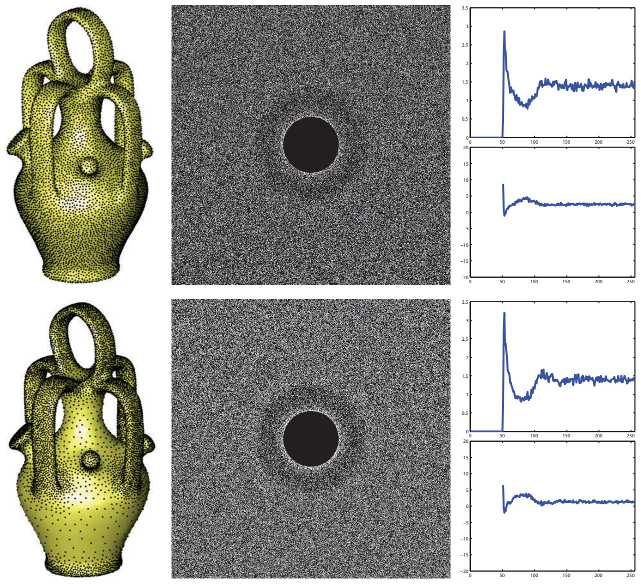


Fig. 9: Spectrum analysis of the uniform (top row) and the adaptive (bottom row) sampling of the Botijo model. Left: samples; middle: power spectrum, and right: radial average (top) and anisotropy (bottom) of the power spectrum.

5.3 Mesh generation

As discussed in [31], the sampled Poisson-disk set is already a very good candidate for mesh generation. There are multiple ways to extract a 3D mesh from a point set. We use the algorithm proposed in [37], [31] to compute a restricted Delauney triangulation. The generated meshes satisfy the Delaunay property and have high quality; see the discussion below.

We have tested our algorithm on various data sets, including 3D CT images (Figure 10) and computationally generated volumetric data sets (Figure 11). The input data sets are scaled into a bounding box with the longest edge length equal to one. We use a fixed sampling radius, $r = 0.008$, for all the results listed in Table 2.

Quality of results. We measure the quality of the resulting meshes in terms of triangle quality, approximation error, and vertex valences. We measure triangle quality by the following values: $Q_1(t)$ is defined as $\frac{6}{\sqrt{3}}\frac{|t|}{p(t)\,h(t)}$, where $|t|$ is the area of the triangle t, $p(t)$ is the half-perimeter of t, and $h(t)$ is the longest edge length of t [38]. The value $Q_2(t)$ is defined as ratio of the radii of incircle and circumcircle. Finally, $\theta_{\min}/\theta_{\max}$ is the ratio of the minimal and maximal angles in a triangle.

The approximation quality is measured by the Hausdorff distance and the Root Mean Square (RMS) distance between each generated mesh and the corresponding reference mesh. The reference meshes are extracted from the input data by the classic marching cubes algorithm [1] with a finer resolution. We adapted the code of [39] as the MC reference algorithm.

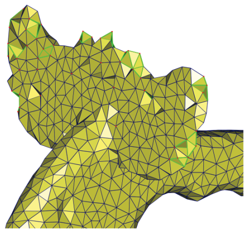
Comparison with other methods. We compare our approach with marching cubes [1] and an active front (AF) algorithm [5]. The results of our comparison are shown in the middle columns of Figures 10 and 11.

In the case of uniform sampling/meshing, the angles of the extracted mesh are bounded between $[30°, 120°]$, and the edge lengths are between $[r, 2\,r]$ (with r as the sampling radius). In adaptive sampling/meshing, our algorithm can always generate meshes with $\theta_{\min}$ around $24°$, which outperforms the result of AF. The AF method generates meshes that are more regular than the meshes generated by other methods, but artifacts can be observed in the regions where multiple fronts meet. Moreover, our algorithm can generate meshes with much better values for Q_1, Q_2. Marching cubes always exhibits better Hausdorff and RMS distances to the reference mesh, which is explained by the fact that many vertices of the new mesh are actually vertices of the reference mesh. Both the AF method and our method have better distributions of vertex valences (which can even further be improved by randomized optimization as proposed in [31]. We did not do that in this paper because it is too slow for some realtime applications).

Timings. The tables below also provide detailed timings for all three methods used for meshing. Note that these timings do not include the time spent for loading the input volumetric data. The MC algorithm is still the most efficient one, but it yields the worst results. In summary, our numerical experiments show that our algorithm is capable of generating high-quality meshes, while sacrificing a little bit of efficiency.

6 Conclusions

We have presented an algorithm for unbiased sampling on isosurfaces, as well as a framework for maximal Poisson-disk sampling and meshing of isosurfaces. We demonstrated that our results have better meshing quality than previous approaches. This is important for applications like rendering, physical simulation, etc.



We rely on existing methods for mesh-extraction (triangulation) after sampling and we therefore inherit all limitations of the chosen mesh-extraction algorithm. For example, we cannot extract a valid mesh from the Elk data set (right figure) due to the large sampling radius and the presence of the thin features. In this case, the mesh can have non-manifold edges (green edges shown in the figure). Another issue is that the efficiency of the adaptive sampling is slower since the subgrid resolution for conflict checking is much larger than uniform sampling. In the future, we would like to address these issues by using an adaptive grid. We are also interested in finding more applications of the presented technique such as topology verification for isosurface extraction [40].

Acknowledgments

We thank the anonymous reviewers for their valuable comments and suggestions. We are grateful to Takashi Michikawa and Hiromasa Suzuki for providing us an implementation of marching cubes, the authors of “afront” [5] for making it publicly available, Miriah Meyer for sharing data with us, and Jianwei Guo for helping on the DDA [36] software. This work was supported by the KAUST Visual Computing Center, the National Natural Science Foundation of China (no. 61372168, 61331018, 61271431, and 61272327), and the U.S. National Science Foundation.

References

[1] W. E. Lorensen and H. E. Cline, “Marching cubes: A high resolution 3D surface construction algorithm,” in *Computer Graphics (Proc. SIGGRAPH)*, vol. 21, 1987, pp. 163–169.

[2] L. Kobbelt, M. Botsch, U. Schwanecke, and H.-P. Seidel, “Feature-sensitive surface extraction from volume data,” in *Proc. ACM SIGGRAPH*, 2001, pp. 57–66.

[3] T. Ju, F. Losasso, S. Schaefer, and J. Warren, “Dual contouring of Hermite data,” in *Proc. ACM SIGGRAPH*, 2002, pp. 339–346.

[4] T. Ju, “Robust repair of polygonal models,” *ACM Trans. Graphics (Proc. SIGGRAPH)*, vol. 23, no. 3, pp. 888–895, 2004.

[5] J. Schreiner, C. Scheidegger, and C. Silva, “High-quality extraction of isosurfaces from regular and irregular grids,” *IEEE Trans. Vis. Comp. Graphics*, vol. 12, no. 5, pp. 1205–1212, 2006.

[6] M. D. Meyer, R. M. Kirby, and R. T. Whitaker, “Topology, accuracy, and quality of isosurface meshes using dynamic particles,” *IEEE Trans. Vis. Comp. Graphics*, vol. 13, no. 6, pp. 1704–1711, 2007.

[7] C. Dietrich, C. Scheidegger, J. Schreiner, J. Comba, L. Nedel, and C. Silva, “Edge transformations for improving quality of marching methods,” *IEEE Trans. Vis. Comp. Graphics*, vol. 15, no. 1, pp. 150–159, 2009.

[8] G. M. Nielson and B. Hamann, “The asymptotic decider: Removing the ambiguity in marching cubes,” in *Visualization ’91*, 1991, pp. 83–91.

Model	Res.	Method	$\#V$	Q_1	Q_2	$\theta_{\min}/\theta_{\max}$	Hdist	RMS	$\theta_{<30^\circ}$	v_{567}	Time
Pelvis	271x390x282	MC	26.9k	$7.8 \cdot 10^{-4}$/0.59	$5.8 \cdot 10^{-5}$/0.66	0.03/177.2	0.226	0.014	41.0	79.6	**0.16**
		MPS	12.5k	**0.48**/0.81	**0.49**/0.89	**30.3/118.1**	0.261	0.019	**0**	96.4	6.8/0.5
		AF	12.5k	0.10/0.88	0.04/0.94	4.34/162.0	0.232	0.018	3.8	**99.2**	27.8
		AMPS	13.5k	0.44/0.81	0.43/0.90	23.7/122.3	0.204	0.017	0.38	95.6	8.2/0.5
Brain	149x188x148	MC	65.9k	$6.3 \cdot 10^{-4}$/0.59	$1.1 \cdot 10^{-4}$/0.67	0.02/178.7	0.443	0.025	39.3	81.7	**0.38**
		MPS	30.9k	**0.48**/0.81	**0.49**/0.90	**30.1/118.0**	0.556	0.037	**0**	95.9	11.1/1.5
		AF	59.1k	0.02/0.90	$7.4 \cdot 10^{-4}$/0.95	1.09/177.8	0.512	0.027	2.66	**99.2**	80.6
		AMPS	30.6k	0.48/0.81	0.48/0.90	27.8/118.3	0.547	0.032	0.02	95.7	12.5/1.4
Skull	64 x 64 x 64	MC	68.2k	$7.0 \cdot 10^{-4}$/0.59	$8.7 \cdot 10^{-5}$/0.67	0.02/178.9	0.288	0.015	39.3	84.2	**0.24**
		MPS	32.3k	**0.48**/0.81	**0.49**/0.89	**30.1/118.1**	0.356	0.023	**0**	96.3	10.8/1.4
		AF	38.4k	0.01/0.95	$3.1 \cdot 10^{-4}$/0.97	0.71/178.6	0.312	0.017	1.13	**99.8**	45.7
		AMPS	11.6k	0.42/0.81	0.41/0.89	23.3/124.3	0.321	0.018	0.38	95.4	12.8/0.5
Dente	156x256x138	MC	29.8k	$6.4 \cdot 10^{-4}$/0.59	$7.1 \cdot 10^{-5}$/0.67	0.02/177.7	0.180	0.010	38.6	84.0	**0.16**
		MPS	14.2k	**0.49**/0.81	**0.51**/0.90	**30.2/116.7**	0.240	0.016	**0**	96.5	4.00/0.5
		AF	13.5k	0.32/0.96	0.26/0.98	16.1/134.6	0.314	0.018	1.06	**99.9**	9.5
		AMPS	6.63k	0.41/0.81	0.39/0.89	24.2/119.7	0.260	0.017	0.23	95.6	8.4/0.25
Bunny	512x508x397	MC	43.8k	$7.7 \cdot 10^{-4}$/0.60	$1.1 \cdot 10^{-4}$/0.67	0.02/179.2	0.352	0.014	37.8	85.8	**0.23**
		MPS	20.9k	**0.48**/0.81	**0.49**/0.90	**30.0/118.2**	0.335	0.017	**0**	96.3	7.58/0.9
		AF	21.8k	0.17/0.95	0.07/0.98	9.1/157.5	0.234	0.018	1.15	**99.8**	21.55
		AMPS	18.4k	0.38/0.81	0.35/0.89	22.5/124.8	0.344	0.018	0.28	95.1	28.6/0.7.55
Genus3	512x430x291	MC	38.2k	$7.4 \cdot 10^{-4}$/0.59	$3.1 \cdot 10^{-5}$/0.66	0.02/179.1	0.254	0.007	40.3	84.3	**0.21**
		MPS	17.8k	**0.47**/0.81	**0.47**/0.89	**30.1/119.5**	0.311	0.014	**0**	96.5	5.26/0.8
		AF	22.8k	0.05/0.96	0.006/0.98	3.1/173.6	0.206	0.087	0.90	**99.9**	18.2
		AMPS	8.7k	0.42/0.81	0.40/0.89	23.9/122.5	0.306	0.025	0.11	96.1	12.5/0.3
Maxpl.	303x512x385	MC	32.1k	$7.6 \cdot 10^{-4}$/0.59	$1.1 \cdot 10^{-4}$/0.67	0.03/178.8	0.396	0.008	39.0	83.9	**0.17**
		MPS	15.3k	**0.48/0.81**	**0.49**/0.90	**30.3/118.2**	0.451	0.013	**0**	96.3	4.8/0.6
		AF	15.2k	$5.2 \cdot 10^{-4}$/0.92	$7.1 \cdot 10^{-7}$/0.96	0.03/179.9	0.228	0.029	39.0	**99.4**	31.2
		AMPS	14.8k	0.44/0.81	0.42/0.89	24.4/123.3	0.448	0.016	0.33	95.6	7.6/0.5
Botijo	266x512x276	MC	25.7k	$6.4 \cdot 10^{-4}$/0.59	$7.1 \cdot 10^{-5}$/0.67	0.02/177.7	0.218	0.017	38.6	83.8	**0.14**
		MPS	12.2k	**0.49**/0.81	**0.51**/0.90	**30.2/116.7**	0.277	0.027	**0**	96.4	3.32/0.5
		AF	11.8k	0.04/0.95	0.004/0.97	2.45/175.0	0.507	0.030	1.16	**99.6**	10.4
		AMPS	12.6k	0.43/0.81	0.41/0.90	21.7/124.1	0.287	0.031	0.17	96.1	11.2/0.4

TABLE 2: Statistics of the meshing quality compared with previous approaches. MC refers to the marching cubes algorithm [1], MPS refers to our algorithm with uniform sampling, AF refers to the advancing front approach [5], and AMPS refers to our algorithm with adaptive sampling. $\#V$ is the number of sampled points. We give the miminal and average values of the triangle quality measures, Q_1 and Q_2, which are defined in the text. $\theta_{\min}/\theta_{\max}$ are the minimal and maximal angles of the generated meshes; *Hdist* and *RMS* are Hausdorff distance and the root mean square distance, respectively, between the resulting mesh and the reference mesh (% of the diagonal length of the bounding box); v_{567} is the percentage of vertices with valences 5, 6 and 7. The timing of our algorithm is split into two parts, i.e., the timing for sampling and the timing for meshing. The best result is highlighted in bold.

[9] Q. Du, V. Faber, and M. Gunzburger, "Centroidal Voronoi tessellations: applications and algorithms," *SIAM Review*, vol. 41, pp. 637–676, 1999.

[10] A. Lagae and P. Dutré, "A comparison of methods for generating Poisson disk distributions," *Computer Graphics Forum*, vol. 27, no. 1, pp. 114–129, 2008.

[11] T. S. Newman and H. Yi, "A survey of the marching cubes algorithm," *Comput. & Graphics*, vol. 30, no. 5, pp. 854–879, 2006.

[12] A. Lopes and K. Brodlie, "Improving the robustness and accuracy of the marching cubes algorithm for isosurfacing," *IEEE Trans. Vis. Comp. Graphics*, vol. 9, no. 1, pp. 16–29, 2003.

[13] G. Nielson, "On marching cubes," *IEEE Trans. Vis. Comp. Graphics*, vol. 9, no. 3, pp. 283–297, 2003.

[14] S. Schaefer, T. Ju, and J. Warren, "Manifold dual contouring," *IEEE Trans. Vis. Comp. Graphics*, vol. 13, no. 3, pp. 610–619, 2007.

[15] C. Dietrich, J. Comba, L. Nedel, C. Scheidegger, and C. Silva, "Edge groups: A new approach to understanding the mesh quality of marching methods," *IEEE Trans. Vis. Comp. Graphics*, vol. 15, no. 1, pp. 150–159, 2009.

[16] Y.-S. Liu, J.-H. Yong, H. Zhang, D. Yan, and J.-G. Sun, "A quasi Monte Carlo method for computing areas of point-sampled surfaces," *Computer-Aided Design*, vol. 38, no. 1, pp. 55–68, 2006.

[17] J. A. Detwiler, R. Henning, R. A. Johnson, and M. G. Marino, "A generic surface sampler for Monte Carlo simulations," *IEEE Trans. Nuclear Science*, vol. 55, no. 4, pp. 2329–2333, 2008.

[18] R. L. Cook, "Stochastic sampling in computer graphics," *ACM Trans. Graphics*, vol. 5, no. 1, pp. 69–78, 1986.

[19] D. Cline, S. Jeschke, A. Razdan, K. White, and P. Wonka, "Dart throwing on surfaces," *Computer Graphics Forum (Proc. EG Symp. Rendering)*, vol. 28, no. 4, pp. 1217–1226, 2009.

[20] J. Bowers, R. Wang, L.-Y. Wei, and D. Maletz, "Parallel Poisson disk sampling with spectrum analysis on surfaces," *ACM Trans. Graphics (Proc. SIGGRAPH Asia)*, vol. 29, no. 6, pp. 166:1–166:10, 2010.

[21] M. Corsini, P. Cignoni, and R. Scopigno, "Efficient and flexible sampling with blue noise properties of triangular meshes," *IEEE Trans. Vis. Comp. Graphics*, vol. 18, no. 6, pp. 914–924, 2012.

[22] Z. Chen, Z. Yuan, Y.-K. Choi, L. Liu, and W. Wang, "Variational blue noise sampling," *IEEE Trans. Vis. Comp. Graphics*, vol. 18, no. 10, pp. 1784–1796, 2012.

[23] M. S. Ebeida, S. A. Mitchell, A. A. Davidson, A. Patney, P. M. Knupp, and J. D. Owens, "Efficient and good Delaunay meshes from random points," *Computer-Aided Design*, vol. 43, no. 11, pp. 1506–1515, 2011.

[24] D. Dunbar and G. Humphreys, "A spatial data structure for fast Poisson-disk sample generation," *ACM Trans. Graphics (Proc. SIGGRAPH)*, vol. 25, no. 3, pp. 503–508, 2006.

[25] T. R. Jones, "Efficient generation of Poisson-disk sampling patterns," *J. Graphics Tools*, vol. 11, no. 2, pp. 27–36, 2006.

[26] K. B. White, D. Cline, and P. K. Egbert, "Poisson disk point sets by hierarchical dart throwing," in *Proceedings of the IEEE Symposium on Interactive Ray Tracing*, 2007, pp. 129–132.

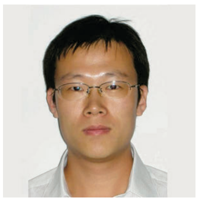
[27] M. N. Gamito and S. C. Maddock, "Accurate multidimensional Poisson-disk sampling," *ACM Trans. Graphics*, vol. 29, no. 1, pp. 8:1–8:19, 2009.

[28] M. S. Ebeida, A. Patney, S. A. Mitchell, A. Davidson, P. M. Knupp, and J. D. Owens, "Efficient maximal Poisson-disk sampling," *ACM Trans. Graphics (Proc. SIGGRAPH)*, vol. 30, no. 4, pp. 49:1–49:12, 2011.

[29] M. S. Ebeida, S. A. Mitchell, A. Patney, A. A. Davidson, and J. D. Owens, "A simple algorithm for maximal Poisson-disk sampling in high dimensions," *Computer Graphics Forum (Proc. Eurographics)*, vol. 31, no. 2, pp. 785–794, 2012.

[30] D.-M. Yan and P. Wonka, "Adaptive maximal Poisson-disk sampling on surfaces," in *ACM SIGGRAPH Asia Technical Briefs*, 2012, pp. 21:1–21:4.

[31] ——, “Gap processing for adaptive maximal Poisson-disk sampling,” *ACM Trans. Graphics*, vol. 32, no. 5, pp. 148:1–148:15, 2013.
[32] J. R. Shewchuk, “What is a good linear element? Interpolation, conditioning, and quality measures,” in *11th Intl. Meshing Roundtable*, 2002, pp. 115–126.
[33] G. Melfi and G. Schoier, “Simulation of random distributions on surfaces,” SIS, 2004.
[34] S. A. Mitchell, A. Rand, M. S. Ebeida, and C. L. Bajaj, “Variable radii Poisson disk sampling,” in *24th Canadian Conference on Computational Geometry (CCCG)*, 2012, pp. 185–190.
[35] L. P. Chew, “Guaranteed-quality mesh generation for curved surfaces,” in *Proceedings of the Ninth Symposium on Computational Geometry*, 1993, pp. 274–280.
[36] L.-Y. Wei and R. Wang, “Differential domain analysis for non-uniform sampling,” *ACM Trans. Graphics (Proc. SIGGRAPH)*, vol. 30, no. 4, pp. 50:1–50:8, 2011.
[37] D.-M. Yan, B. Lévy, Y. Liu, F. Sun, and W. Wang, “Isotropic remeshing with fast and exact computation of restricted Voronoi diagram,” *Computer Graphics Forum*, vol. 28, no. 5, pp. 1445–1454, 2009.
[38] P. Frey and H. Borouchaki, “Surface mesh evaluation,” in *6th Intl. Meshing Roundtable*, 1997, pp. 363–374.
[39] J. Bloomenthal, “An implicit surface polygonizer,” in *Graphics Gems IV*, 1994, pp. 324–349.
[40] R. M. Kirby, C. T. Silva, T. Etiene, V. Pascucci, T. J. Peters, J. Tienry, C. Scheidegger, and L. G. Nonato, “Topology verification for isosurface extraction,” *IEEE Trans. Vis. Comp. Graphics*, vol. 19, no. 6, pp. 952–965, 2012.



Dong-Ming Yan is a research scientist at King Abdullah University of Science and Technology (KAUST). He received his Ph.D. from Hong Kong University in 2010, and his Master’s and Bachelor’s degrees from Tsinghua University in 2005 and 2002, respectively. His research interests include computer graphics, geometric processing and visualization.

Johannes Wallner is the head of the Institute of Geometry at Graz University of Technology. Before that he was with TU Wien, where he also received his Ph.D. in 1997. His research interests are in applied geometry and geometry processing, in discrete differential geometry, and in geometric aspects of approximation theory.

Peter Wonka is Associate Professor at KAUST (King Abdullah University of Science and Technology) and ASU (Arizona State University). His research interests include topics in computer graphics, visualization, computer vision, remote sensing, image processing, and machine learning.

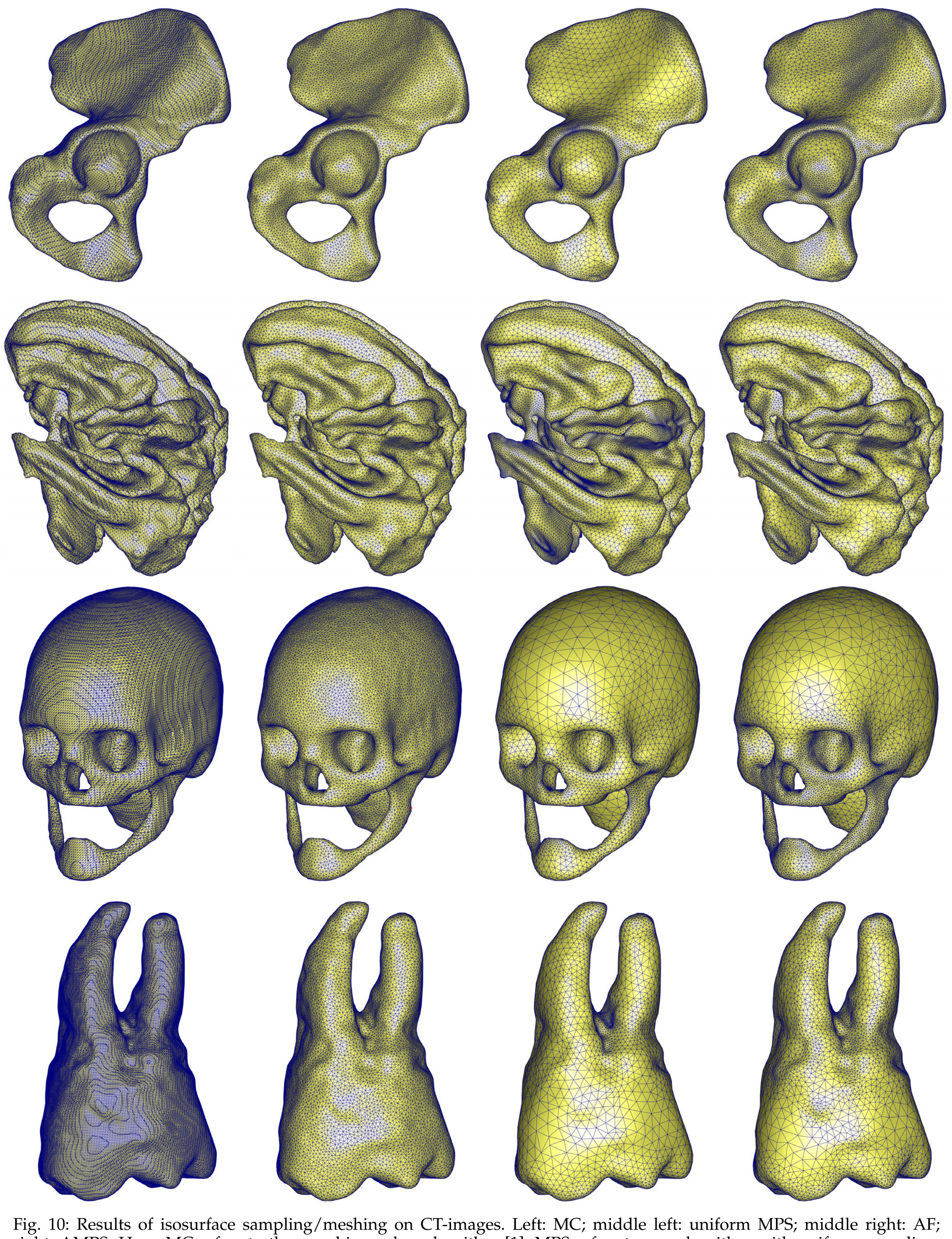


Fig. 10: Results of isosurface sampling/meshing on CT-images. Left: MC; middle left: uniform MPS; middle right: AF; right: AMPS. Here, MC refers to the marching cubes algorithm [1], MPS refers to our algorithm with uniform sampling, AF refers to the advancing front approach [5], and AMPS refers to our algorithm with adaptive sampling.

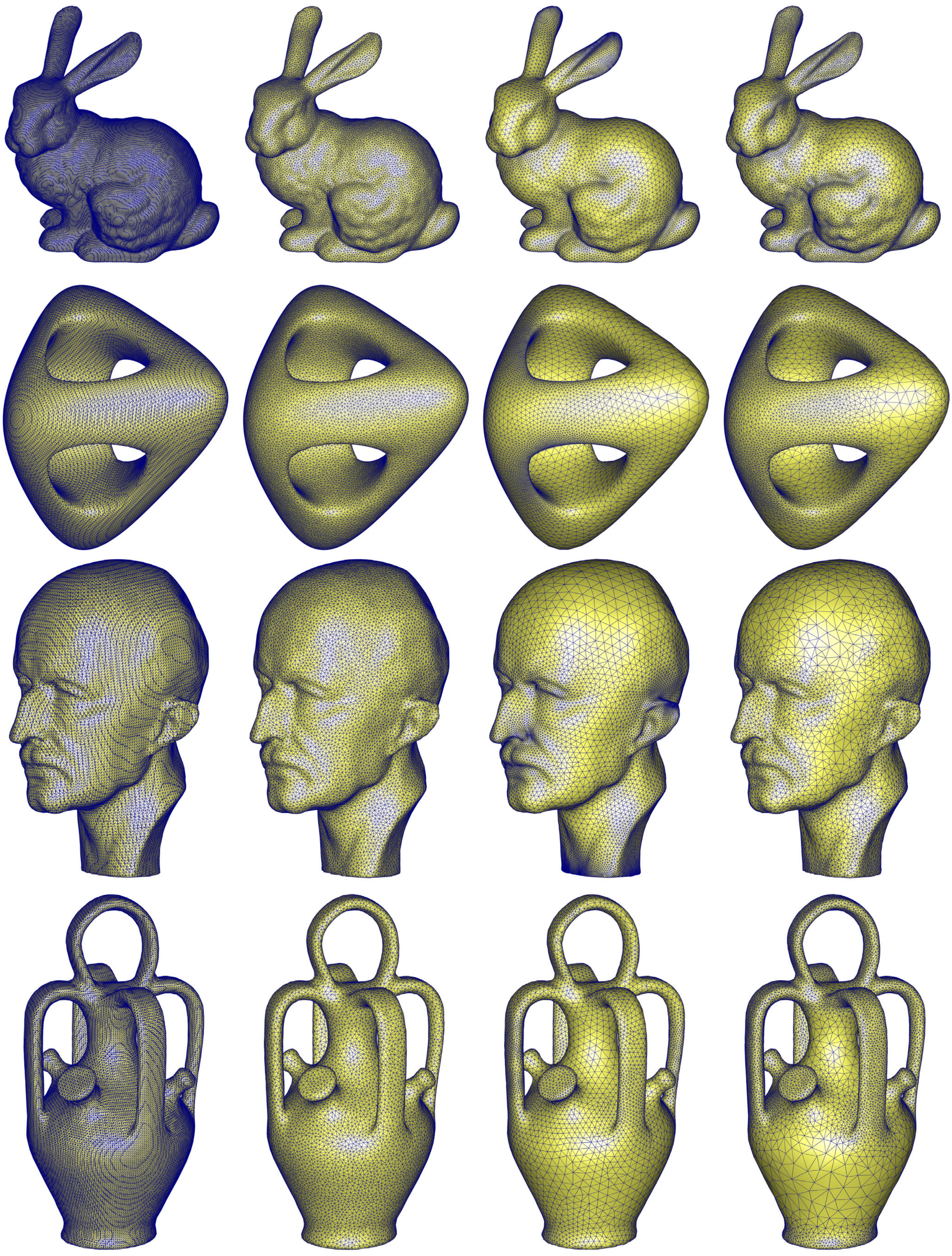


Fig. 11: Results of isosurface sampling/meshing on synthetic data. Left: MC; middle left: uniform MPS; middle right: AF; right: adaptive MPS. Here, MC refers to the marching cubes algorithm [1], MPS refers to our algorithm with uniform sampling, AF refers to the advancing front approach [5], and AMPS refers to our algorithm with adaptive sampling.