A Fast and Simple Surface Reconstruction Algorithm[*]

Siu-Wing Cheng[†] Jiongxin Jin[‡] Man-Kit Lau[†]

Abstract

We present an algorithm for surface reconstruction from a point cloud. It runs in $O(n \log n)$ time, where n is the number of sample points, and this is optimal in the pointer machine model. The only existing $O(n \log n)$-time algorithm is due to Funke and Ramos, and it uses some sophisticated data structures. The key task is to extract a locally uniform subsample from the input points. Our algorithm is much simpler and it is based on a variant of the standard octree. We built a prototype that runs an implementation of our algorithm to extract a locally uniform subsample, invokes Cocone to reconstruct a surface from the subsample, and adds back the samples points absent from the subsample via edge flips. In our experiments with some non-uniform samples, the subsample extraction step is fast and effective, and the prototype gives a 51% to 68% speedup from using Cocone alone. The prototype also runs faster on locally uniform samples.

[*]Research supported by the Research Grants Council, Hong Kong, China (project no. 612109). A preliminary version appeared in Proceedings of the 28th Annual Symposium on Computational Geometry, 2012.

[†]Department of Computer Science and Engineering, HKUST, Hong Kong, `{scheng,lmkaa}@cse.ust.hk`

[‡]Google Inc., jamesjjx@google.com

1 Introduction

The problem of reconstructing surfaces from an unorganized point cloud in $\mathbb{R}^3$ arises in many applications such as reverse engineering, medical imaging, computer graphics and cartography. Given a point cloud sampled from object surfaces, which is often captured by using laser range scanners or medical images like CT or MRI scans, a reconstruction of the surfaces is sought that is faithful. The problem was first studied by researchers in computer graphics and computer vision (e.g., [1, 17, 26]). Many of such algorithms are effective in practice, but do not come with theoretical guarantees.

Amenta and Bern [2] proved that restricted Delaunay triangulation is a good reconstruction. Amenta, Choi, Dey and Leekha [3] developed the first provably correct algorithm, and the output reconstruction consists of Delaunay triangles. When the sample is dense enough, the output approximates the unknown surfaces well in the sense that their normal deviation and Hausdorff distance are small, and they are homeomorphic. The algorithm of Amenta, Choi, Dey, and Leekha [3], known as the Cocone algorithm, also has a successful implementation. There is another Delaunay-based reconstruction algorithm by Amenta, Choi and Kolluri that uses power diagram and weighted Delaunay triangulation [4]. Simplicial reconstructions of manifolds in higher dimensions have been studied [8, 9, 14]. Researchers have also studied reconstructions as zero-sets of implicit functions [7, 13]. More results on surface reconstruction can be found in [18].

The algorithms in [2, 3, 4] compute the 3D Delaunay triangulation or Voronoi diagram or their weighted versions. Let n denote the number of input sample points. The 3D Voronoi diagram can be constructed in $O(n^2)$ time [12], and the fastest output-sensitive algorithm runs in $O((n+f)\log^2 n)$ time, where f is the output size [11]. If the surface is smooth and generic[1] and the sample is uniform, Attali, Boissonnat, and Lieutier [6] showed that the 3D Voronoi diagram has $O(n \log n)$ complexity. However, in the case of non-generic smooth surfaces (e.g. sphere, a smooth surface with a cylindrical part), Erickson [22] proved that the 3D Voronoi diagram can have $\Omega(n\sqrt{n})$ complexity even if the sampling is uniform.

Given a locally uniform sample, Dey, Funke, and Ramos [19] showed that the Cocone algorithm [3] can be modified to run in $O(n \log n)$ time. This modified Cocone algorithm uses an approximate nearest neighbor searching data structure. For a locally uniform sample, Dumitriu, Funke, Kutz and Milosavljević [20, 21] developed a reconstruction algorithm that requires very little geometry (only the distances among the sample points), and therefore, the robustness is greatly enhanced. Subsequently, Funke and Ramos [24] obtained a reconstruction algorithm that runs in $O(n \log n)$ time, which is optimal in the pointer machine model. The algorithm employs several sophisticated data structures such as the well-separated pair decomposition, approximate directional nearest neighbor searching, and approximate range searching. No implementation has been reported so far. The key task of the algorithm of Funke and Ramos is to extract a locally uniform subsample from the input points.

Our main result is an algorithm to extract a locally uniform subsample from a dense input sample that does not use any sophisticated data structure. After building a variant of the standard octree in $O(n \log n)$ time, the extraction works in $O(n)$ time by traversing the octree. Then, reconstructing from the subsample takes $O(n \log n)$ time using the modified Cocone algorithm [19], and adding back the sample points not in the locally uniform subsample can be done in $O(n \log n)$ time as shown by Funke and Ramos [24]. The underlying surfaces are assumed to be smooth, but they need not be generic. In reconstructing from the locally uniform subsample, the octree also allows us to do without the approximate nearest neighbor searching

[1]There are several technical conditions for genericity as stated in [6], one of which is that if a ball is interior-disjoint from the surface, the ball does not touch the surface at more than a constant number of points.

data structure used in the modified Cocone algorithm [24]. We built a prototype that runs an implementation of our subsample extraction algorithm, invokes Cocone to reconstruct a surface from the subsample, and adds back the remaining samples points via edge flips. In our experiments with non-uniform samples, the subsample extraction step is fast and effective, and the prototype is 51% to 68% faster than running Cocone alone. The prototype also runs faster on locally uniform samples.

2 Background and Overview

2.1 Background

We use $d(\cdot,\cdot)$ to denote the Euclidean distance between two geometric objects. Let $B(c,r)$ denote the ball centered at a point c with radius r. Let $B_\infty(c,r)$ denote the axis parallel box with center c and side length $2r$.

Let $\mathcal{M}$ be an m-dimensional smooth compact manifold without boundary in $\mathbb{R}^d$ for some $k \in [1, d-1]$. The manifold $\mathcal{M}$ may have multiple connected components. The *medial axis* of Σ is the closure of the set of points in $\mathbb{R}^3$ that have two or more closest points in $\mathcal{M}$ [2, 23]. The *local feature size* of a point $x \in \Sigma$, denoted by $f(x)$, is the distance between x and the medial axis.

Definition 1. *Let P be a discrete point set drawn from $\mathcal{M}$. We call P an ε-sample for some $\varepsilon \in (0,1)$ if $d(x,P) \leq \varepsilon f(x)$ for every point $x \in \Sigma$.*

Definition 2. *Let P be an* ε-sample *of $\mathcal{M}$. For all $p \in P$, define r_p to be the radius of the largest ball that is centered at some point in $\mathcal{M}$, contains p in its boundary, and does not contain any point in P in its interior. We call P locally uniform if there exist constants $\kappa_{\mathrm{uni}} > 1$ and $c \geq 1$ such that for every point $p \in P$, $|B(p, \kappa_{\mathrm{uni}} r_p) \cap P| \leq c$.*

Our definition of local uniformity is weaker than that in [24], but it suffices for our purposes.

In the case that the manifold is two-dimensional and embedded in $\mathbb{R}^3$, we denote the manifold by Σ, call each connected component in Σ a *surface*, and use $\mathbf{n}_p$ to denote the outward unit normal at a point $p \in \Sigma$. The following results concerning samples of surfaces in $\mathbb{R}^3$ will be useful.

Lemma 2.1 ([3, 5, 18, 25])**.** *Let Σ be a two-dimensional smooth compact manifold without boundary in $\mathbb{R}^3$. Let p, q and r be three arbitrary distinct points in Σ. The following properties hold for sufficiently small ε.*

(i) *If $d(p,q) \leq \varepsilon f(p)$, the acute angle between the support lines of $\mathbf{n}_p$ and pq is at least $\pi/2 - O(\varepsilon)$, and $\angle(\mathbf{n}_p, \mathbf{n}_q) = O(\varepsilon)$.*

(ii) *If the circumradius of pqr is at most $\varepsilon f(p)$, the acute angle between the support lines of $\mathbf{n}_p$ and the normal of pqr is $O(\varepsilon)$.*

(iii) *For all point x on the tangent plane at p, if $d(p,x) \leq \varepsilon f(p)$, then $d(x,\Sigma) = O(\varepsilon d(p,x))$.*

Most provably correct surface reconstruction algorithms in the literature produce triangulated closed surfaces from an ε-sample, and the output surfaces satisfy some desirable properties. To facilitate the subsequent discussion, we define a *faithful reconstruction* as follows.

Definition 3. *Let P be an ε-sample of a 2-manifold Σ in $\mathbb{R}^3$. A faithful reconstruction T is a set of triangulated surfaces with vertices in P that satisfy the following properties: (i) T is homeomorphic to Σ; (ii) the Hausdorff distance between T and Σ is $O(\varepsilon)$; (iii) for every triangle τ in T and for every vertex p of τ, the normal of τ makes an $O(\varepsilon)$ angle with $\mathbf{n}_p$.*

The *restricted Delaunay triangulation* of an ε-sample plays a key role in surface reconstruction as it is known to be a faithful reconstruction of Σ when ε is sufficiently small. A vertex, edge, and triangle in the 3D Delaunay triangulation is a *restricted* Delaunay vertex, edge, and triangle with respect to Σ if its dual Voronoi cell, face, and edge intersects Σ. The restricted Delaunay triangulation cannot be computed though unless Σ is unknown.

For all point $p \in \Sigma$, the *cocone* at p is the set of points $x \in \mathbb{R}^3$ such that the acute angle between px and the support line of the estimated normal at p is at least $\frac{\pi}{2} - \theta$ for some constant $\theta \in (0, \pi/8)$ fixed *a priori*. The Cocone algorithm collects Delaunay triangles whose dual Voronoi edges intersect the cocones at the sample points, followed by a manifold extraction step. The choice of θ depends on the target ε for which the surface reconstruction algorithm works.

2.2 Overview of our algorithm

Our algorithm has two high-level steps as listed below, where were also taken by the algorithm of Funke and Ramos [24].

- Step 1: Select a subset $U_P \subseteq P$ such that U_P is a locally uniform $O(\varepsilon)$-sample of Σ.
- Step 2: For each sample point $p \in U_P$, estimate the surface normal at p and construct the Delaunay triangles incident to p (with respect to U_P) such that the dual Voronoi edge of each such triangle τ intersects the cocones at all three vertices of τ. Then, traverse all such triangles to extract a surface triangulation M by a simple linear time search [3].

Step 1 is the hardest. All aforementioned sophisticated algorithmic tools such as well-separated pair decomposition, approximate direction nearest neighbor searching, and approximate range searching are employed for this purpose. Our first contribution is a variant of the standard octree that allows us to perform step 1 in $O(n \log n)$ time.

In step 2, we need to determine the candidate triangles and extract a manifold from these triangles as in the Cocone algorithm. We want to avoid computing the 3D Delaunay triangulation of U_P because this would take more than $O(n \log n)$ time. We will argue later in Section 4.3 that the candidate triangles can only connect p to other sample points within a distance $\alpha r'_p$ from p for some appropriate constant $\alpha > 1$, where r'_p is the radius of the largest ball that is centered at some point in Σ, contains p in its boundary, and does not contain any point of U_P in its interior. Since U_P is locally uniform, if we can ensure that $\alpha \leq \kappa_{\text{uni}}$, then there are only $O(1)$ points of U_P in $B(p, \alpha r'_p)$. Therefore, by computing the 3D Delaunay triangulation of p and these $O(1)$ points, step 2 takes only $O(1)$ time for each $p \in U_P$, provided that we can estimate the surface normal at p and identify the sample points within a distance $\alpha r'_p$ from p.

The triangulation M is a provably good reconstruction of Σ. The only downside is that it does not include all points in P. Funke and Ramos [24] proposed to include all points in P by computing the Delaunay triangulation of P restricted to M in $O(n \log n)$ time. The readers are referred to [24] for details.[2]

The Cocone algorithm [3] and other quadratic time algorithms in [2, 4] need to invoke a 3D Delaunay triangulation or 3D Voronoi diagram algorithm. In comparison, the 3D Delaunay construction in step 2 is no harder.

In addition to selecting a locally uniform $O(\varepsilon)$-sample $U_P \subseteq P$, our octrees also allow us to: (i) estimate $\mathbf{n}_p$ at each point $p \in P$ in step 2 above, and (ii) collect the set $U_P \cap B(p, \alpha r'_p)$ for every $p \in U_P$ in step 2 above. The algorithmic details will be given later in Section 4.3.

[2]In computing the Delaunay triangulation of P restricted to M as described in [24], the sets $P \cap B(p, \alpha r_p)$ for every $p \in U_P$ are needed, where α is some appropriate constant. Our octree can be used to return these sets efficiently.

Our main result is concerned with step 1, the extraction of a locally uniform subsample U_P from P. The following observation motivates our approach.

Lemma 2.2. *Let P be an ε-sample of $\mathcal{M}$. Let R be a compact subset of $\mathbb{R}^d$. Suppose that R contains at least one sample point in P. Then at least one of the following properties is valid for a small enough ε:*

(i) *$d(R,\, P \setminus R) \leq 4\varepsilon f(p)$ for some sample point $p \in P \cap R$.*

(ii) *For each connected component $\mathcal{S}$ of $\mathcal{M}$, either R contains $P \cap \mathcal{S}$, or R is disjoint from $P \cap \mathcal{S}$.*

Proof. Suppose that condition (ii) does not hold for some connected $\mathcal{S}$ of $\mathcal{M}$. That is, $P \cap \mathcal{S}$ has some points in R and some points not in R. Let $\mathcal{U}_\mathcal{S} = \bigcup_{p \in P \cap \mathcal{S}} B(p, 1.5\varepsilon f(p))$.

Since P is an ε-sample, for every point $x \in \Sigma$, $d(x, P) \leq \varepsilon f(x)$. Let p be the nearest neighbor of x in P. As f is 1-Lipschitz, $f(x) \leq f(p) + d(x,p) \leq f(p) + \varepsilon f(x)$, which implies that $f(x) \leq f(p)/(1-\varepsilon) < 1.5 f(p)$ when $\varepsilon < 1/3$. Therefore, $\mathcal{S} \subseteq \mathcal{U}_\mathcal{S}$ and hence $\mathcal{U}_\mathcal{S}$ is connected for a small enough ε.

The connectedness of $\mathcal{U}_\mathcal{S}$ implies the existence of $p, q \in P \cap \mathcal{S}$ such that $p \in R$, $q \notin R$, and $d(p,q) \leq 1.5\varepsilon f(p) + 1.5\varepsilon f(q)$. As $f(q) \leq f(p) + d(p,q)$, we have $d(R,\, P \setminus R) \leq d(p,q) \leq \frac{3\varepsilon}{1-1.5\varepsilon} f(p) \leq 4\varepsilon f(p)$ for $\varepsilon \leq 1/6$, satisfying condition (i) in the lemma. $\square$

In the sequel, R will be some octree cells with a high concentration of sample points, and Lemma 2.2 helps us to decide whether R contains a surface or the sample points in R should be decimated to achieve local uniformity.

Consider a 2-manifold $\Sigma \subset \mathbb{R}^3$ and the special case of Σ being connected (i.e., only one surface). We start with a smallest axis-aligned bounding cube of P and build an octree by repeated splitting. The splitting continues until for every leaf cell C, the sample points in $P \cap C$ concentrate in an octree cube C' of one-eighth the side length of C, called the core of C. Since there is only one surface, Lemma 2.2(ii) cannot hold with $R = C'$. Then, Lemma 2.2(i) applies with $R = C'$. In the simplest situation that C' does not touch the boundary of C, by the octree alignment, the distance between the boundaries of C' and C is at least the side length of C', and this implies that the side lengths of C' and C are $O(\varepsilon)$ times the local feature size of some point in C'. Thus, the side length of C is a local length scale that reflects the local sampling density. We smooth the different local length scales induced by different octree leaf cells, and use the smoothed scales to prune P to a locally uniform subsample U_P. In general, the core C' of a leaf cell C can touch the boundary of C, and it may be in contact with the cores of some other leaf cells. We need to group the cores into clusters, and apply Lemma 2.2 with R equal to a cluster.

When there are several surfaces in Σ, some sample points may concentrate in a cluster X because they lie on some surfaces in Σ that lie (almost) entirely inside X. It is wrong to decimate the sample points in X. Fortunately, Lemma 2.2(i) does not hold with $R = X$. Then Lemma 2.2(ii) implies that the sample points in X can be treated as the input for an independent instance of the surface reconstruction problem, which can be solved recursively. Of course, nobody can foretell whether Lemma 2.2(i) holds or not. We first recursively reconstruct from the sample points in X. If the reconstruction is successful, we ignore the sample points in X because the surfaces containing these points have been reconstructed. If the reconstruction fails, it is safe to decimate the sample points in X.

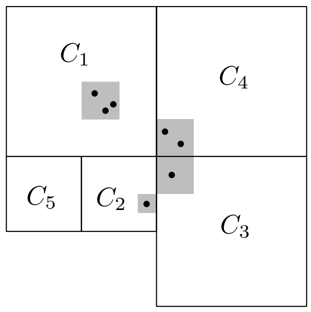


Figure 1: Five leaf cells are shown. C_5 is empty, so it does not have a core. The cores of other leaf cells are shown in gray: $\text{core}(C_1)$ alone is a cluster, and the union $\text{core}(C_2) \cup \text{core}(C_3) \cup \text{core}(C_4)$ is another cluster.

3 Octree Decomposition

For generality, we describe our octree decomposition in $\mathbb{R}^d$ for $d \geq 2$ for a sample P drawn from a smooth, compact and boundaryless manifold $\mathcal{M}$ with dimension $k \in [1, d-1]$. Construct a smallest axis-aligned bounding box of P. By scaling, we assume that this bounding box is $[0,1]^d$. Given a box C in $\mathbb{R}^d$, we denote its side length by ℓ_C. We call a box $C \subseteq [0,1]^d$ *canonical* if $\ell_C = 2^{-i}$ for some integer $i \geq 0$, and the coordinates of the corners of C are integral multiples of 2^{-i}. If we divide C into 2^d equal boxes, we obtain the canonical boxes inside C of the next smaller size 2^{-i-1}.

3.1 The top-level octree

The bounding box $[0,1]^d$ of P is the root and the only leaf cell of the initial octree. A canonical box C is *splittable* if two points in $P \cap C$ lie in the interior of two distinct canonical boxes in C with side length $2^{-d}\ell_C$. Two canonical boxes are *neighbors* if their interiors are disjoint and their boundaries are in contact, possibly at a corner only. The initial octree is grown by invoking the following two rules alternately.

- *Splitting rule:* As long as there is a splittable leaf cell C, split C into 2^d octree cells of side length $\frac{1}{2}\ell_C$ and make them the children of C in the octree.
- *Balancing rule:* As long as there are two neighboring leaf cells C and C' such that $\ell_C > 2\ell_{C'}$, split C as described in the splitting rule.

The splitting rule is applied first until no leaf cell is splittable. The balancing rule is then applied until the tree is *balanced*, i.e., every two neighboring leaf cells differ in side lengths by at most a factor two. We apply the two rules alternately until the tree stops growing, which must happen because, in the worst case, every leaf cell contains a single point and is non-splittable, and then the tree will stop growing after another round of balancing.

Let $\mathcal{T}_P$ be the resulting octree. We add pointers between neighboring cells in $\mathcal{T}_P$ that have the same side length. These pointers provide access to cells of similar sizes near a given cell. A cell is *empty* if it contains no point in P. For every non-empty leaf cell C, since C is not splittable, $P \cap C$ is contained in a canonical box with side length $2^{-d}\ell_C$ inside C. We call it the *core* of C, denoted $\text{core}(C)$. Two cores are connected if their boundaries are in contact, possibly at a corner only. Each connected component of cores is called a *cluster*. An isolated core is a cluster by itself. Figure 1 gives an example.

Lemma 3.1 below proves that every cluster contains a constant number of cores of similar sizes, and every leaf cell is small with respect to the local feature size. The lemma is stated for any subset $Q \subseteq P$ that satisfies a conformity condition in Definition 4 below.

Definition 4. *Let Q be a subset of P. We say that P* conforms *to $\mathcal{T}_Q$ if the following conditions are satisfied.*

(i) *For every cell C in $\mathcal{T}_Q$, either $Q \cap C = \emptyset$ or $Q \cap C = P \cap C$;*

(ii) *If $Q \neq P$, then $d(Q, P \setminus Q)$ is at least 2^{-d-2} times the side length of the root cell of $\mathcal{T}_Q$.*

Note that P conforms to $\mathcal{T}_P$ trivially.

Lemma 3.1. *Let P be an ε-sample of $\mathcal{M}$. Let Q be an arbitrary subset of P. Let $\mathcal{T}_Q$ be the octree constructed for Q by applying the splitting and balancing rules alternately. $\mathcal{T}_Q$ has the following properties.*

(i) *The side lengths of two neighboring leaf cells differ by at most a factor two.*

(ii) *Every pair of leaf cells whose cores are in the same cluster are neighbors. Hence, each cluster consists of at most 2^d cores, and the side lengths of two cores in the same cluster differ by at most a factor two.*

(iii) *Suppose that P conforms to $\mathcal{T}_Q$. For every non-empty leaf cell C of $\mathcal{T}_Q$, let X denote the cluster that contains* $\mathrm{core}(C)$*, if $Q \cap X$ and $Q \setminus X$ contain sample points from the same connected component of $\mathcal{M}$, then for a small enough ε, $\ell_C \leq 2^d \cdot 20\varepsilon f(p)$ for every $p \in Q \cap C$.*

Proof. Property (i) follows from the octree construction. Let X be a cluster. Define $\mathcal{X} = \{\text{non-empty leaf cell } C : \mathrm{core}(C) \subseteq X\}$. We show that every two cells in X are neighbors, and then (ii) follows.

Let $\mathcal{K}$ be a maximal subset of $\mathcal{X}$ such that the cores of cells in $\mathcal{K}$ are connected and every two cells in $\mathcal{K}$ are neighbors. If $\mathcal{K} = \mathcal{X}$, we are done. Otherwise, we can find a cell $C_1 \in \mathcal{X} \setminus \mathcal{K}$ such that $\mathrm{core}(C_1)$ is connected to the core of some cell in $\mathcal{K}$. Pick $C_2 \in \mathcal{K}$ that maximizes $w = d_\infty(C_1, C_2)$, where d_∞ denotes the L_∞ distance function. We have $w > 0$ by the maximality of $\mathcal{K}$. Take an axis-aligned slab L of width w that separates C_1 and C_2. Let $\mathcal{K}_L \subseteq \mathcal{K}$ be the subset of cells that overlap with this slab. Let $\ell_{\min}$ and $\ell_{\max}$ be the side lengths of the smallest and largest cells in $\mathcal{K}_L$, respectively.

We first show that $|\mathcal{K}_L| \leq 2^{d-1}$. Take the hyperplane H that bounds the slab L and contains a boundary facet of C_2. Notice that $\{H \cap \mathrm{core}(C) : C \in \mathcal{K}_L\}$ is a collection of disjoint $(d-1)$-dimensional boxes that have a non-empty common intersection in H. One can pack at most 2^{d-1} boxes in such a configuration. Figure 2(a) shows a two-dimensional example in which 2^{d-1} cells in $\mathcal{K}_L$ share a common vertex of C_2.

We claim that $w \geq \ell_{\max}/2$. Let C be the largest cell in $\mathcal{K}_L$. If C_1 is not a neighbor of C, then $w \geq d_\infty(C, C_1) \geq \ell_C/2 = \ell_{\max}/2$. Suppose that C_1 is a neighbor of C. Then $\ell_{C_1} \geq \ell_C/2$. Also, C_2 and C are neighbors as they belong to $\mathcal{K}$, so $\ell_{C_2} \geq \ell_C/2$. The octree alignment implies that $d_\infty(C_1, C_2) \geq \min\{\ell_{C_1}, \ell_{C_2}\} \geq \ell_C/2 = \ell_{\max}/2$.

Suppose that $|\mathcal{K}_L| < 2^{d-1}$ or $w \geq \ell_{\max}$. Since the cores of the cells in $\mathcal{X}$ are connected, the gap between C_1 and C_2 is bridged by the cores of the cells in $\mathcal{K}_L$. The side lengths of cores of cells in $\mathcal{K}_L$ are at most $2^d\ell_{\max}$. Then $d_\infty(C_1, C_2) \leq |\mathcal{K}_L| 2^{-d}\ell_{\max}$. If $|\mathcal{K}_L| < 2^{d-1}$, then $d_\infty(C_1, C_2) < \ell_{\max}/2 \leq w$. Otherwise, $w \geq \ell_{\max}$ by assumption and $|\mathcal{K}_L| = 2^{d-1}$, so $d_\infty(C_1, C_2) = \ell_{\max}/2 < w$. In either case, we obtain a contradiction to the definition of w. The remaining possibility is that $|\mathcal{K}_L| = 2^{d-1}$ and $w = \ell_{\max}/2$. If $\ell_{\min} < \ell_{\max}$, then $\ell_{\max} = 2\ell_{\min}$.

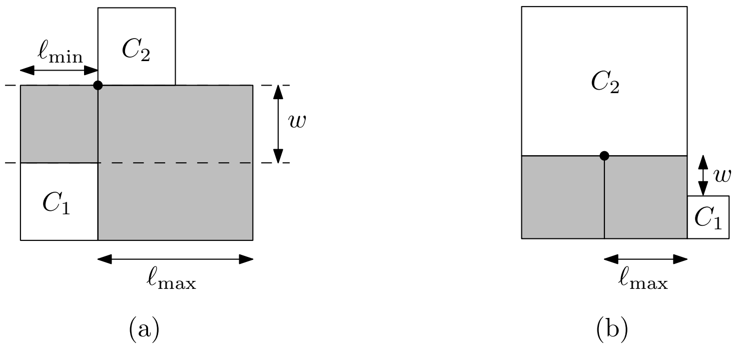


Figure 2: The shaded boxes are cells in $\mathcal{K}_L$. In (a), these cells share the lower left vertex of C_2. In (b), C_1 is at L_∞-distance $\ell_{\max}$ from the left shaded box.

At least one cell in $\mathcal{K}_L$ has side length $\ell_{\min}$ and its core has side length at least $2^{-d}\ell_{\min}$. But then

$$\begin{aligned} d_\infty(C_1, C_2) &\leq (|\mathcal{K}_L| - 1)2^{-d}\ell_{\max} + 2^{-d}\ell_{\min} \\ &\leq \frac{2^{d-1} - 1 + 1/2}{2^d}\ell_{\max} \\ &< \ell_{\max}/2 \\ &= w. \end{aligned}$$

Again, this contradicts the definition of w. If $\ell_{\min} = \ell_{\max}$, then since $|\mathcal{K}_L| = 2^{d-1}$, the cores in $\{\text{core}(C) : C \in \mathcal{K}_L\}$ form a full packing of disjoint boxes of equal sizes around a single point. Therefore, there exits a cell in $\mathcal{K}_L$ that is at L_∞ distance $\ell_{\max} > w$ from C_1. Refer to Figure 2(b) for an illustration. This contradicts the choice of C_2 to maximize $d_\infty(C_1, C_2)$. This establishes property (ii).

Consider (iii). If $Q \cap X$ and $Q \setminus X$ contain sample points from the same connected component $\mathcal{S}$, then $P \cap X$ and $P \setminus X$ contain sample points from $\mathcal{S}$ too as P conforms to $\mathcal{T}_Q$ by assumption. Lemma 2.2(i) implies that $d(X, P \setminus X) \leq 4\varepsilon f(q)$ for some $q \in P \cap X = Q \cap X$. Let C be an arbitrary non-empty leaf cell such that $\text{core}(C) \subseteq X$. By (ii), the smallest core in X has side length at least $\frac{1}{2}\ell_{\text{core}(C)}$. Since the distance between X and any core not in X is at least half of the side length of the smallest core in X, we have $\frac{1}{4}\ell_{\text{core}(C)} \leq d(X, P \setminus X) \leq 4\varepsilon f(q)$, or equivalently,

$$\ell_{\text{core}(C)} \leq 2^4\varepsilon f(q). \tag{3.1}$$

Take any $p \in Q \cap C = Q \cap \text{core}(C)$. By (i) and (ii), the cells containing p and q are neighbors, and their side lengths are within a factor of two. Therefore, the difference between p and q in any coordinate is at most $2^{d+1}\ell_{\text{core}(C)} \leq 2^{d+5}\varepsilon f(q)$. It follows that $d(p, q) \leq 2^{d+4}\sqrt{d}\varepsilon f(q)$. Then, the 1-Lipschitzness of local feature size implies that $f(q) \leq f(p) + d(p, q) \leq f(p) + 2^{d+4}\sqrt{d}\varepsilon f(q)$. Therefore, for a small enough ε, we get $f(q) \leq \frac{5}{4}f(p)$. Substituting this inequality into (3.1) gives $\ell_C = 2^d\ell_{\text{core}(C)} \leq 2^d \cdot 20\varepsilon f(p)$. ◻

3.2 Other octrees

If we know that $\mathcal{M}$ is connected, then by Lemma 3.1(iii), the leaf cells of $\mathcal{T}_P$ are small enough (with respect to local feature size) for the purpose of obtaining a locally uniform sample by decimation. However, since we do not know whether $\mathcal{M}$ is connected, we need to build an octree for each cluster of $\mathcal{T}_P$ in a recursive manner. In general, after building an octree $\mathcal{T}_Q$

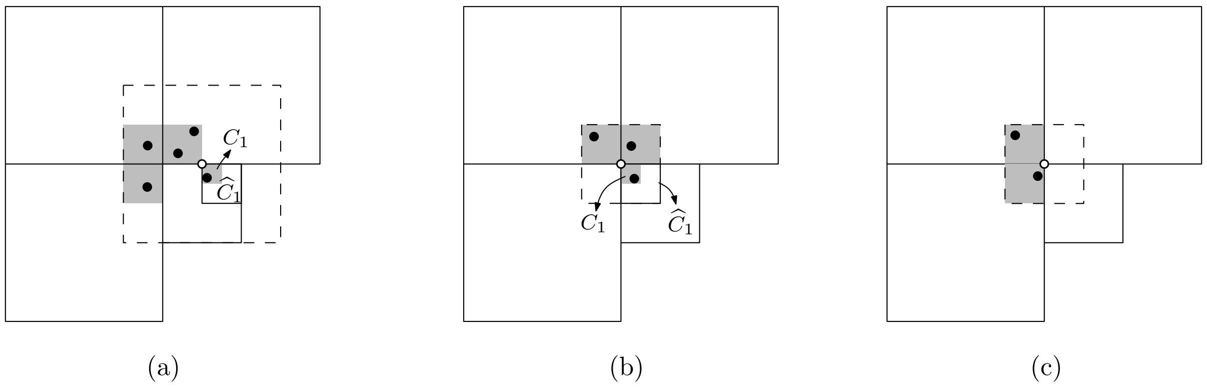


Figure 3: The shaded cores form a cluster. In (a) and (b), there is a core C_1 that is smaller than other cores, so $\ell_{\widehat{C}_1} = 2\ell_{C_1}$. In (a), $\mathcal{I} = \emptyset$; in (b), $\mathcal{I}$ is a point; in (c), $\mathcal{I}$ is a segment. The box with dashed boundary is C_X, whose center is shown as a white dot.

for a subset $Q \subseteq P$, for each cluster X in $\mathcal{T}_Q$ that contains more than one sample point, we recursively build $\mathcal{T}_{Q\cap X}$ by initializing an octree of $O(1)$ levels, followed by alternate splitting and balancing. The initialization is important. It ensures that no cell of $\mathcal{T}_{Q\cap X}$ contains sample points from different cores in X, which helps to achieve the $O(n \log n)$ running time stated in Lemma 3.4. The initialization works as follows.

By Lemma 3.1(ii), there are at most 2^d cores in X. Label them C_i for $i = 1, 2, \cdots$. Let $\ell_{\max}$ be the side length of the largest core in X. Lemma 3.1(ii) implies that every core C_i has side length $\ell_{\max}$ or $\ell_{\max}/2$. For each core C_i, let $\widehat{C}_i$ be the canonical box (with respect to $\mathcal{T}_Q$) such that $\ell_{\widehat{C}_i} = \ell_{\max}$, and $C_i \subseteq \widehat{C}_i$. The box $\widehat{C}_i$ is contained in the leaf cell of $\mathcal{T}_Q$ whose core is C_i, so $\widehat{C}_i$ and $\widehat{C}_j$ have disjoint interiors for $i \neq j$. Let $\mathcal{I} = \bigcap_i \widehat{C}_i$, which can be computed in $O(1)$ time. See Figure 3. We construct the root cell C_X of $\mathcal{T}_{Q\cap X}$ and the initial octree according to the type of $\mathcal{I}$ as follows.

If $\mathcal{I}$ is a d-dimensional cube (i.e., X contains only one core), set C_X to be a smallest axis-aligned d-dimensional bounding cube of $Q \cap X$. And C_X alone is the initial octree.

The remaining possibilities are that $\mathcal{I}$ is empty or a k-dimensional cube for some $k \in [0, d-1]$. We define a special point v as follows. If $\mathcal{I}$ is empty, then set v to be an arbitrary corner of an arbitrary $\widehat{C}_i$; otherwise, set v to be an arbitrary vertex of $\mathcal{I}$ which must be a vertex of $\widehat{C}_i$ for some i. Then, set C_X to be a smallest axis-aligned d-dimensional cube with v as the center such that C_X contains all $\widehat{C}_i$. Figure 3 shows some examples. By Lemma 3.1(ii), there are at most 2^d $\widehat{C}_i$ and and they are connected. So $\ell_{C_X} \leq 2^d \ell_{\max}$. We use C_X as the octree root cell and then split C_X repeatedly until each leaf cell has size $\ell_{\max}/2$. This gives the initial octree, which has $O(1)$ levels.

After building the initial octree, we split and balance alternately to obtain the final octree $\mathcal{T}_{Q\cap X}$. Pointers are added between neighboring octree cells in $\mathcal{T}_{Q\cap X}$ with the same side length. We then recurse on the clusters in $\mathcal{T}_{Q\cap X}$ that contain more than one sample point.

Lemma 3.2. *Given $\mathcal{T}_Q$ and a cluster X in $\mathcal{T}_Q$, the following properties are valid.*

(i) *The initial octree for $\mathcal{T}_{Q\cap X}$ has $O(1)$ levels. For each leaf cell of the initial octree, either it is empty of $Q \cap X$ or it contains points of $Q \cap X$ from exactly one core in X.*

(ii) *There are two or more clusters in $\mathcal{T}_{Q\cap X}$, or the sample points in some core in X are divided into two or more leaf cells of $\mathcal{T}_{Q\cap X}$.*

(iii) *All leaf cells of the initial octree are not larger than the smallest core in X.*

Proof. Recall that the C_i's denote the cores in X, $\ell_{\max}$ is the side length of the largest core in X, $\widehat{C}_i$ denotes the canonical box that contains C_i and has side length $\ell_{\max}$, and $\mathcal{I} = \bigcap_i \widehat{C}_i$. Also, C_X denotes the root cell of the initial octree.

Consider (i). The number of levels follows from the preceding description. If $\mathcal{I}$ is a d-dimensional cube, then X contains only one core and so property (i) is trivially true. If $\mathcal{I}$ is empty or a k-dimensional cube for some $k \in [0, d-1]$, then C_X is a d-dimensional cube that contains all $\widehat{C}_i$ and C_X is aligned with the hyperplanes that contain the boundaries of cells in $\mathcal{T}_Q$. By Lemma 3.1(ii), every C_i has side length $\ell_{\max}/2$ or $\ell_{\max}$. Therefore, after splitting C_X into leaf cells with side lengths $\ell_{\max}/2$, for every such leaf cell C, either $C = C_i$ for some i, or $C \subset C_i$ for some i, or the interior of C is disjoint from the interior of C_i for all i. This establishes property (i).

Consider (ii) in the case that $\mathcal{I}$ is empty. We show that there are two or more clusters in $\mathcal{T}_{Q\cap X}$. Assume to the contrary that there is only one cluster in $\mathcal{T}_{Q\cap X}$. Then, by Lemma 3.1(ii), the non-empty leaf cells of $\mathcal{T}_{Q\cap X}$ are neighbors of each other. This implies that the $\widehat{C}_i$'s must also be neighbors of each other. But then $\mathcal{I}$ should be non-empty, a contradiction.

Consider (ii) in the case that $\mathcal{I}$ is non-empty. If $\mathcal{I}$ is a d-dimensional cube, then X has only one core and C_X is a smallest axis-aligned d-dimensional bounding cube of $Q \cap X$. It implies that C_X is splittable and that when we split C_X, the sample points in X will be divided into two more more children of C_X. The sample points will subsequently go into two or more leaf cells of $\mathcal{T}_{Q\cap X}$. If $\mathcal{I}$ is a k-dimensional cube for some $k \in [0, d-1]$, the center of C_X is a vertex v of $\mathcal{I}$ and C_X is aligned with the hyperplanes that contain the boundaries of cells in $\mathcal{T}_Q$. Since $\mathcal{I} = \bigcap_i \widehat{C}_i$, there exists an axis-aligned hyperplane through $\mathcal{I}$ that separates $\widehat{C}_i$ and $\widehat{C}_j$ for some i and j. Therefore, when we divide C_X into 2^d cells of equal size, the sample points in C_X cannot lie in only one of these 2^d cells. It implies that when we split C_X repeatedly into leaf cells of size $\ell_{\max}/2$ to form the initial octree, the sample points in X will be divided into two or more leaf cells of the initial octree. In particular, the sample points in the largest core in X (i.e., with size $\ell_{\max}$) fall into two or more leaf cells of the initial octree. Subsequently, these samples points go into two or more leaf cells of $\mathcal{T}_{Q\cap X}$.

Consider (iii). If $\mathcal{I}$ is a d-dimensional cube, then there is only one core in X, C_X is the only leaf cell of the initial octree, and C_X is a a smallest axis-aligned bounding cube of $Q \cap X$. Therefore, C_X is not larger than the core in X. In other cases, the smallest core in X has side length $\ell_{\max}/2$ or more and the leaf cells of the initial octree have side lengths $\ell_{\max}/2$. ◻

Next, we show that P conforms to all octrees obtained in the above recursive construction.

Lemma 3.3. *P conforms to all octrees constructed by the recursive procedure described above.*

Proof. We prove the lemma by induction. P conforms to $\mathcal{T}_P$, so the base case holds. Suppose that P conforms to $\mathcal{T}_Q$ for a subset $Q \subseteq P$. Let X be any cluster in $\mathcal{T}_Q$. By the construction of $\mathcal{T}_{Q\cap X}$, it is clear that the first condition in Definition 4 is satisfied. We check the second condition of Definition 4.

Let C be the smallest core in X. Let C_X be the root cell of $\mathcal{T}_{Q\cap X}$. By Lemma 3.1(ii), X contains at most 2^d cores and each core has side length ℓ_C or $2\ell_C$. It follows that $\ell_{C_X} \le 2^{d+1}\ell_C$. The distance between X and any other cluster in $\mathcal{T}_Q$ is at least $\ell_C/2$ by the octree alignment. Therefore,

$$d(X, Q \setminus X) \ge \ell_C/2 \ge 2^{-d-2}\ell_{C_X}. \tag{3.2}$$

Since P conforms to $\mathcal{T}_Q$ by induction assumption and the root cell of $\mathcal{T}_Q$ is at least as large as C_X, the second condition in Definition 4 implies that

$$d(Q, P \setminus Q) \ge 2^{-d-2}\ell_{C_X}. \tag{3.3}$$

Recall that $Q \cap X = P \cap X$. Therefore,

$$\begin{aligned} d(Q \cap X, P \setminus (Q \cap X)) &= d(Q \cap X, P \setminus X) \\ &= \min\{d(Q \cap X, Q \setminus X), d(Q \cap X, P \setminus Q)\} \\ &\geq \min\{d(X, Q \setminus X), d(Q, P \setminus Q)\} \\ &\geq 2^{-d-2}\ell_{C_X}. \qquad (\because (3.2) \ \& \ (3.3)) \end{aligned}$$

Thus, $\mathcal{T}_{Q \cap X}$ also satisfies the second condition in Definition 4. ◻

We use the properties of the octrees established above to prove that the total size of the octrees is $O(n)$ and it takes only $O(n \log n)$ time to construct them, as stated in Lemma 3.4 below. The proof is quite long though. Therefore, we postpone it to section 6 and move on to the surface reconstruction algorithm in the next section.

Lemma 3.4. *The total size of the octrees is $O(n)$. The total construction time is $O(n \log n)$.*

4 Surface Reconstruction

In this section, Σ denotes a 2-manifold in $\mathbb{R}^3$ that may consist of multiple surfaces. Let P denote an ε-sample drawn from Σ.

We first compute $\mathcal{T}_P$ and the other octrees as described in the previous section. Then we call Reconstruct($\mathcal{T}_P$) to perform the surface reconstruction. The pseudocode of Reconstruct is given in Algorithm 1. Given an octree $\mathcal{T}_Q$ constructed for some $Q \subseteq P$, Reconstruct($\mathcal{T}_Q$) performs three steps.

Algorithm 1 Reconstruct($\mathcal{T}_Q$)

```
1:  for all cluster $X$ in $\mathcal{T}_Q$ do
2:    flag ← Reconstruct($\mathcal{T}_{Q \cap X}$)
3:    if flag = true then
4:      Remove $Q \cap X$ from $\mathcal{T}_Q$.
5:    end if
6:  end for
7:  if $\mathcal{T}_Q$ still contains some sample points then
8:    (flag,$\overline{\mathcal{T}}_Q$) ← Trim($\mathcal{T}_Q$)   // $\overline{\mathcal{T}}_Q$ is the tree after the trimming
9:    if flag = true then
10:     $U_Q$ ← Extract($\overline{\mathcal{T}}_Q$)   // $U_Q$ is a locally uniform subsample
11:     Reconstruct the surfaces from which $U_Q$ is sampled.
12:     return true  // reconstruction succeeds
13:   else
14:     return false  // reconstruction fails
15:   end if
16: else
17:   return true  // nothing else to process
18: end if
```

First, for every cluster X in $\mathcal{T}_Q$, we call Reconstruct($\mathcal{T}_{Q \cap X}$) recursively to reconstruct the surfaces whose sample points are completely contained in $\mathcal{T}_{Q \cap X}$. We will show later in Lemma 4.1 that if a cluster contains a partial sample, it contains a partial sample of exactly one surface and no sample point from any other surface. Therefore, if $T_{Q \cap X}$ contains some complete samples, then $T_{Q\cap X}$ contains complete samples only and the reconstruction of the corresponding surfaces will be successful. The sample points on these surfaces are then removed from $\mathcal{T}_Q$. The remaining sample points can be treated as an independent input. This explains the removal of $Q \cap X$ in line 4.

Second, we call TRIM($\mathcal{T}_Q$), which returns **true** if and only if the remaining sample points in $\mathcal{T}_Q$ form complete samples of some surfaces. In addition, TRIM($\mathcal{T}_Q$) prunes $\mathcal{T}_Q$ to remove the cells that are too small with respect to the local sampling density. Pruning also helps to decide whether TRIM should return **true** or **false**. Intuitively, if the remaining sample points contain a partial sample from some surface, then since the root cell of $\mathcal{T}_Q$ is induced by a cluster in the "parent octree", there must be a large gap between points in this cluster and other sample points from the same surface. This gap detection is done in a bottom-up manner in $\mathcal{T}_Q$. If a non-empty leaf cell C is near a gap on the estimated tangent plane for points in C and this gap is large relative to ℓ_C, then C is "too small" with respect to the local sampling density. So we remove C and make parent(C) a new leaf cell. Repeating the above prunes $\mathcal{T}_Q$ to $\overline{\mathcal{T}}_Q$. We will prove that TRIM($\mathcal{T}_Q$) should return **true** if and only if $\overline{\mathcal{T}}_Q$ contains more than one node.

Third, if TRIM($\mathcal{T}_Q$) returns **false**, then RECONSTRUCT($\mathcal{T}_Q$) aborts and returns **false**. On the other hand, if TRIM($\mathcal{T}_Q$) returns **true**, we proceed to reconstruct the surfaces with complete samples in $\overline{\mathcal{T}}_Q$. The pruning by TRIM ensures that all non-empty cells in $\overline{\mathcal{T}}_Q$ have the right sizes with respect to the local sampling density. Thus, we can traverse $\overline{\mathcal{T}}_Q$ to smooth the side lengths of non-empty leaf cells by pruning the tree further. Finally, one arbitrary point is picked from each non-empty leaf cell to form a locally uniform sample. We then perform steps 2 and 3 described in the overview in beginning of Section 2.2 to reconstruct the surfaces.

Before elaborating on the procedures TRIM and EXTRACT in lines 8 and 10 in RECONSTRUCT in the following subsections, we prove Lemma 4.1 which has been alluded to earlier.

Lemma 4.1. *Let P be an ε-sample of Σ. Let Q be a arbitrary subset of P. Suppose that P conforms to $\mathcal{T}_Q$ and ε is sufficiently small. For every cluster X in $\mathcal{T}_Q$, if X contains a partial sample of some surface, then X does not contain any sample point from any other surface in Σ.*

Proof. Let S be a surface with a partial sample in X. Pick a point $p \in S \cap X$. By Lemma 3.1(iii), the non-empty leaf cell that contains p has size at most $160\varepsilon f(p)$. Lemma 3.1(ii) implies that the size of the largest core in X is at most $40\varepsilon f(p)$, and the diameter of X is at most $320\sqrt{3}\varepsilon f(p)$. Assume to the contrary that X contains a sample point q from another surface in Σ. Since the medial axis of Σ separates the surfaces in Σ, the line segment pq must cross the medial axis. But then $320\sqrt{3}\varepsilon f(p) \geq d(p,q) \geq f(p)$, which is impossible for a small enough ε. ◻

4.1 Tree Trimming

Before calling TRIM($\mathcal{T}_Q$) in RECONSTRUCT($\mathcal{T}_Q$), we have recursively reconstructed surfaces whose sample points are contained in the clusters of $\mathcal{T}_Q$ and removed these sample points.

Consider a non-empty leaf cell C of $\mathcal{T}_Q$. It contains only a partial sample from some surface because the sample points in C would have been removed otherwise. Thus, $\ell_C \leq 160\varepsilon f(q)$ for every point $q \in Q \cap C$ by Lemma 3.1(iii).

Refer to the pseudocode of TRIM in Algorithm 2. The algorithm puts non-empty leaf cells in a list L and then examine them one by one. Let C be the current non-empty leaf cell. Partition $B_\infty(c, 2.5\ell_C)$ into a set $\{R_i : 1 \leq i \leq 5^3\}$ of canonical cubes of side length ℓ_C.

As shown in steps 10–15, we first estimate the surface normal at a point $p \in Q \cap C$: find two sample points p_i and p_j in some R_i and R_j, respectively, such that $\angle p_i p p_j$ is neither too large nor too small so that $\mathbf{n}_{pp_ip_j}$ is a good estimate by Lemma 2.1(ii). If the surface normal estimation fails, we call MAKELEAF to make parent(C) a leaf cell. Otherwise, we look for a

Algorithm 2 TRIM($\mathcal{T}_Q$)

```
1: $\overline{\mathcal{T}}_Q \leftarrow \mathcal{T}_Q$
2: $L \leftarrow$ list containing all non-empty leaf cells in $\overline{\mathcal{T}}_Q$ (in any order)
3: while $L$ is not empty do
4:   $C \leftarrow$ EXTRACTFIRST($L$)
5:   if $C$ is the root of $\overline{\mathcal{T}}_Q$ then
6:     return $\overline{\mathcal{T}}_Q$  //$\overline{\mathcal{T}}_Q$ only contains the root
7:   end if
8:   $c \leftarrow \text{center}(C)$
9:   //$B_\infty(x, r)$ is the axis-aligned cube with center $x$ and side length $2r$.
10:  Partition $B_\infty(c, 2.5\ell_C)$ into a set $\{R_i : 1 \leq i \leq 5^3\}$ of canonical cubes of side length $\ell_C$.
11:  For each non-empty $R_i$, $p_i \leftarrow$ an arbitrary point in $Q \cap R_i$.
12:  Note that $C \in \{R_i : 1 \leq i \leq 5^3\}$. Let $p$ be the point picked in step 11 from $Q \cap C$.
13:  $\theta \leftarrow \arccos(0.97)$
14:  if $\exists\, i, j \in [1, 5^3]$ such that $\angle p_i p p_j$ lies in $[\theta, \pi - \theta]$ then
15:    $\tilde{\mathbf{n}}_p \leftarrow \mathbf{n}_{p_i p p_j}$
16:  else  //Normal estimation fails
17:    Remove the non-empty leaf cells in the subtree of parent($C$) from $L$.
18:    MAKELEAF(parent($C$))
19:    $L \leftarrow L \cup \{\text{parent(C)}\}$
20:    continue //skip the following and go to line 4 for the next cell
21:  end if
22:  $H \leftarrow$ plane through $p$ and orthogonal to $\tilde{\mathbf{n}}_p$
23:  $\mathcal{U} \leftarrow$ union of empty cubes $R_i$
24:  if $\exists\, x \in H$ such that $B_\infty(x, \frac{1}{16}\ell_C) \subseteq \mathcal{U}$ then
25:    Remove the non-empty leaf cells in the subtree of parent($C$) from $L$.
26:    MAKELEAF(parent($C$))
27:    $L \leftarrow L \cup \{\text{parent}(C)\}$
28:  end if
29: end while
30: return $\overline{\mathcal{T}}_Q$
```

Algorithm 3 MAKELEAF(C) //Turn C to a leaf

```
1: Discard the subtree rooted at $C$.
2: Regard the sample points in the descendants of $C$ as sample points in $C$.
```

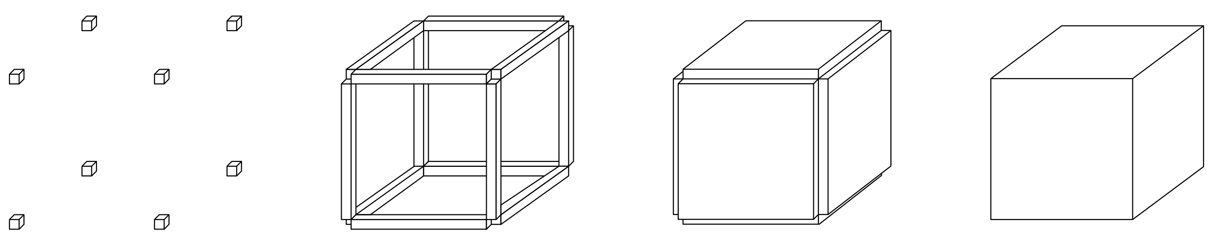


Figure 4: R_i is the disjoint union of $G_{i,0}$, $G_{i,1}$, $G_{i,2}$ and $G_{i,3}$. The first figure is $G_{i,0}$. The second figure is $G_{i,1}$. The third figure is $G_{i,2}$, which consists of six slabs. There is a rectangular hole in the middle. The last figure is $G_{i,3}$, which fits into the hole in $G_{i,2}$.

"big gap" on the estimated tangent plane and if there is one, we also make parent(C) a leaf cell. MakeLeaf and steps 17–19 of Trim work by removing the descendants of parent(C) from the list L and make parent(C) a leaf.[3] Let H denote the estimated tangent plane at p. Let $\mathcal{U}$ denote the union of non-empty R_i's. We say that there is a "big gap" on H if there is a point $x \in H$ such that $B_\infty(x, \frac{1}{16}\ell_C) \subseteq \mathcal{U}$ (step 24 of Trim).

The detection of such a $B_\infty(x, \frac{1}{16}\ell_C)$ in $\mathcal{U}$ can be done in $O(1)$ time as follows. Divide the empty R_i's into canonical cubes of side length $\frac{1}{16}\ell_C$. Put these smaller empty cubes into four groups $G_{i,m}$ for $m \in [0,3]$ such that $G_{i,m}$ is the union of the cubes that intersect an m-dimensional face of R_i but not any face of R_i with dimension less than m. Figure 4 shows an example. Each $G_{i,m}$ consists of several elements of the same kind. $G_{i,3}$ is one cube in the interior of R_i with side length $\frac{14}{16}\ell_C$, $G_{i,2}$ consists of six rectangular slabs in contact with the facets of R_i, $G_{i,1}$ consists of twelve rectangular rods in contact with the edges of R_i, and $G_{i,0}$ consists of eight cubes in contact with the vertices of R_i. Assume that the plane H intersects an empty R_i. Then, H intersects an element K of $G_{i,m}$ for some $m \in [0,3]$ (i.e., a cube for $m = 0$, a rod for $m = 1$, a slab for $m = 2$, and a cube for $m = 3$). There exists a point $x \in H \cap K$ such that $B_\infty(x, \frac{1}{16}\ell_C) \subseteq \mathcal{U}$ if and only if there are 2^{3-m} cubes in the set $\{R_j \subseteq \mathcal{U} : R_j \text{ touches or contains } K\}$. Figure 5 shows an example. So testing whether there is such an empty cube $B_\infty(x, \frac{1}{16}\ell_C)$ takes $O(1)$ time.

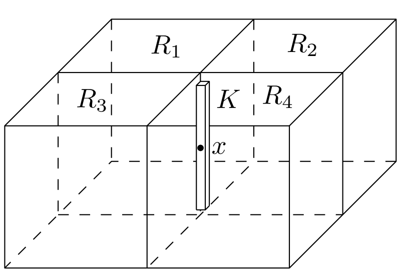
A leaf cell C is trimmed when the normal estimation for the point picked in C fails (line 16 of Trim) or when a "big gap" is found (line 24 of Trim). Lemmas 4.2 and 4.3 below show that C is small in either case, and so it is safe to trim C and perform at parent(C) the normal estimation and the check for a "big gap".

We first state and prove Lemma 4.2 that applies when the normal estimation for point picked in C fails in line 16 of Trim.

Lemma 4.2. *Consider the call* Trim($\mathcal{T}_Q$) *for some* $Q \subseteq P$ *such that* P *conforms to* $\mathcal{T}_Q$. *Let* C *be the octree cell currently being processed in the while-loop in lines 3–29 of* Trim. *Suppose that there exists a constant* $\lambda \geq 1$ *such that* $\ell_C \leq \lambda\varepsilon f(q)$ *for every point* $q \in Q \cap C$. *If the normal estimation in* Trim *fails for* C *and* ε *is small enough, then* $\ell_C \leq 20\varepsilon f(q)$ *for every point* $q \in Q \cap C$.

Proof. Let p be the sample point in C that is picked for the normal estimation. For $1 \leq i \leq 5^3$, let R_i denote the canonical cubes in the partition of $B_\infty(c, 2.5\ell_C)$ into cubes of side length ℓ_C. According to the algorithm, we have picked at most one sample point p_i from every non-empty R_i. There are two cases.

[3]Every octree cell keeps a linked list of samples points in it. When a cell A is removed during the trimming of an octree, the linked list of sample points at A is appended to the linked list of sample points at parent(A) in $O(1)$ time.



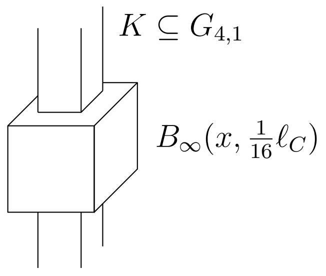


Figure 5: The point x lies in a rod K in $G_{4,1}$. R_1, R_2, R_3 and R_4 are the four cubes in contact with K. Since the base side length of K is $\ell_C/16$, the box $B_\infty(x, \frac{1}{16}\ell_C)$ must intersect all these four cubes. Moreover, it is contained in the union of these four cubes. Thus, $B_\infty(x, \frac{1}{16}\ell_C) \subseteq \mathcal{U}$ if and only if these four cubes belong to $\mathcal{U}$.

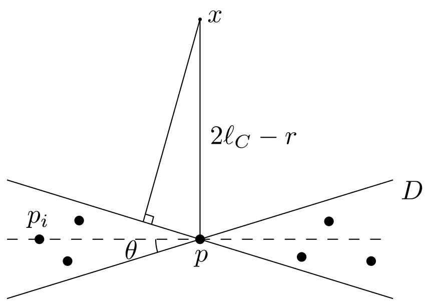


Figure 6: The distance between x and any pick is at least $d(x,p)\cos\theta$.

The first possibility is that there exists $i \in [1, 5^3]$ such that $R_i \neq C$ and a sample point p_i is picked successfully for R_i. Let x be a point in Σ such that px is perpendicular to pp_i and $d(p,x) = 2\ell_C - r$, where $r = 0.1\ell_C$. Such a point x exists in the neighborhood of p in the intersection of Σ and the plane through p perpendicular to pp_i.

Suppose that $B(x,r) \cap P = \emptyset$. It follows from ε-sampling that $\varepsilon f(x) > r = 0.1\ell_C$, or equivalently $\ell_C < 10\varepsilon f(x)$. For every sample point $q \in C$,

$$d(x,q) \leq d(x,p) + d(p,q) \leq 2\ell_C - r + \sqrt{3}\ell_C < 4\ell_C < 40\varepsilon f(x).$$

Then, $f(q) \geq f(x) - d(x,q) > (1 - 40\varepsilon)f(x)$. Therefore, when ε is small enough,

$$\ell_C < 10\varepsilon f(x) < \frac{10\varepsilon f(q)}{1 - 40\varepsilon} < 20\varepsilon f(q).$$

Suppose that $B(x,r) \cap P \neq \emptyset$. We first show that $B(x,r) \cap P \subseteq P \setminus Q$. The ball $B(x,r)$ is completely outside C, and the furthest distance between a point in $B(x,r)$ and C is at most $r + d(p,x) = 2\ell_C$. It follows that that $B(x,r)$ lies inside $B_\infty(c, 2.5\ell_C)$, where c is the center of C. Therefore, if $B(x,r)$ contains any sample point in Q, that sample point must lie in R_j for some j. In the following, we argue that the sample points in $Q \cap R_j$ for all j are outside $B(x,r)$. Let $\theta = \arccos(0.97)$. Consider the double cone with p as the apex, aperture $\pi - 2\theta$, and the support line of px as the axis. Let D be the complement of this double cone. See Figure 6. Since we cannot find a pick p_j such that $\angle p_i p p_j \in [\theta, \pi - \theta]$, D must contain the pick p_j for all non-empty R_j. Thus, for every non-empty R_j,

$$d(x,p_j) \geq d(p,x)\cos\theta = (2\ell_C - r)\cos\theta.$$

Every sample point in $Q \cap R_j$ is at distance $\sqrt{3}\ell_C$ or less from p_j. Therefore, the distance from x to any sample point of Q in any non-empty R_j (including C) is at least

$$d(x,p_j) - \sqrt{3}\ell_C \geq (2\ell_C - r)\cos\theta - \sqrt{3}\ell_C > r.$$

This shows that $B(x,r)$ cannot contain any sample point in Q. As a result, $B(x,r) \cap P \subseteq P \setminus Q$.

We are ready to show that $\ell_C \leq 20\varepsilon f(q)$ for all sample point $q \in Q \cap C$ in the case that $B(x,r) \cap P \neq \emptyset$. Let p' be a sample point in $B(x,r) \cap P$. Let C_Q be the root cell of $\mathcal{T}_Q$. Since P conforms to $\mathcal{T}_Q$, $d(Q, P \setminus Q) \geq 2^{-5}\ell_{C_Q}$ by condition (ii) in Definition 4. Then

$$d(x,p) \geq d(p,p') - d(x,p') \geq d(Q, P\setminus Q) - r \geq 2^{-5}\ell_{C_Q} - r. \tag{4.1}$$

Since $x \in B_\infty(c, 2.5\ell_C)$ and $p \in C$, the distance between p and x is at most $3\sqrt{3}\ell_C$. Combining with (4.1) gives

$$2^{-5}\ell_{C_Q} - r \leq 3\sqrt{3}\ell_C \quad \Rightarrow \quad \ell_C > 2^{-8}\ell_{C_Q}. \tag{4.2}$$

By assumption of the lemma, $\ell_C \leq \lambda\varepsilon f(q)$ for every point $q \in Q \cap C$. By (4.2), C_Q has side length $\ell_{C_Q} < 2^8\ell_C < 2^8 \cdot \lambda\varepsilon f(q))$ for every point $q \in Q \cap C$. Therefore, for a small enough ε, C_Q cannot contain a complete sample from any surface in Σ. By the recursive construction of octrees, C_Q contains at most eight cores $\{\text{core}(C_1), \text{core}(C_2), \ldots\}$ in some non-empty leaf cells $C_1, C_2, \ldots$ in some "parent" octree tree. None of these C_i's contains a complete sample from any surface in Σ because C_Q does not. By Lemma 3.1(iii), $\ell_{C_i} \leq 160\varepsilon f(q)$ for every point $q \in Q \cap C_i$. By Lemma 3.2(i) and (iii), for each leaf cell of the initial octree rooted at C_Q, it contains sample points from only one core(C_i) and it is not larger than the smallest core(C_i). The cell C is a descendant of a leaf cell of the initial octree rooted at C_Q. Therefore, $\ell_C \leq (160\varepsilon f(q))/8 = 20\varepsilon f(q)$ for every point $q \in Q \cap C$.

The remaining possibility is that for all $i \in [1, 5^3]$ such that $R_i \neq C$, we cannot pick a sample point p_i from R_i. That is, $Q \cap R_i = \emptyset$ for all $R_i \neq C$. We set x to be an arbitrary point in Σ such that $d(p,x) = 2\ell_C - r$. Then, we can repeat the previous analysis for the possibility that we successfully pick a sample point p_i from R_i for some $i \in [1, 5^3]$. Since $Q \cap R_i = \emptyset$ for all $R_i \neq C$, the analysis for the case of $B(x,r) \cap P \neq \emptyset$ can be simplified. $\square$

Next, we state and prove Lemma 4.3 that shows that the cell C is small when a "big gap" is detected in line 24 of Trim. Recall that $\mathcal{U}$ is the union of the non-empty cubes R_i's in the partition of $B_\infty(c, 2.5\ell_C)$ into cubes of side lengths ℓ_C.

Lemma 4.3. *Consider the call* Trim$(\mathcal{T}_Q)$ *for some $Q \subseteq P$ such that P conforms to $\mathcal{T}_Q$. Let C be the octree cell currently being processed in the while-loop in lines 3–29 of* Trim. *Suppose that there exists a constant $\lambda \geq 1$ such that $\ell_C \leq \lambda\varepsilon f(q)$ for every point $q \in Q \cap C$. If* Trim *obtains an estimated tangent plane H in line 22 and $\mathcal{U}$ contains an empty $B_\infty(x, \frac{1}{16}\ell_C)$ for some $x \in H$ and ε is small enough, then $\ell_C \leq 20\varepsilon f(q)$ for every point q in $Q \cap C$.*

Proof. Let p denote the point picked for C for normal estimation. Since $p \in C$ and $x \in B_\infty(c, 2.5\ell_C)$, where c is the center of C, we obtain $d(p,x) = O(\ell_C) = O(\varepsilon f(p))$ by the assumption of the lemma that $\ell_C \leq \lambda\varepsilon f(q)$ for all $q \in Q \cap C$. Let x' be the projection of x in the plane tangent to Σ at p. Let $\tilde{x}$ be the point in Σ nearest to x'. By Lemma 2.1(ii), H makes an $O(\varepsilon)$ angle with the tangent plane at p, so

$$d(x,x') = O(\varepsilon\, d(p,x)) = O(\varepsilon\ell_C) = O(\varepsilon^2 f(p)).$$

Therefore,

$$d(p,x') \leq d(p,x) + d(x,x') = O(\ell_C) = O(\varepsilon f(p)).$$

By Lemma 2.1(iii),

$$d(x', \tilde{x}) = O(\varepsilon d(p, x')) = O(\varepsilon \ell_C) = O(\varepsilon^2 f(p)).$$

It follows that

$$d(x, \tilde{x}) \le d(x, x') + d(x', \tilde{x}) = O(\varepsilon \ell_C) = O(\varepsilon^2 f(p)).$$

Suppose that $B(\tilde{x}, \frac{1}{16}\ell_C - d(x, \tilde{x})) \cap P = \emptyset$. It follows from ε-sampling that $\varepsilon f(\tilde{x}) \ge \frac{1}{16}\ell_C - d(x, \tilde{x})$, implying that $\ell_C \le 16\varepsilon f(\tilde{x}) + 16\, d(x, \tilde{x})$. Since $d(p, \tilde{x}) \le d(p, x) + d(x, \tilde{x}) = O(\varepsilon f(p))$, we obtain $f(\tilde{x}) \le f(p) + d(p, \tilde{x}) \le (1 + O(\varepsilon)) f(p)$. It follows that $\ell_C \le (16\varepsilon + O(\varepsilon^2)) f(p)$. For every point $q \in Q \cap C$, by the assumption of the lemma that $\ell_C \le \lambda \varepsilon f(q)$, we obtain $d(p, q) \le \sqrt{3}\ell_C = O(\varepsilon f(q))$, and so $f(p) \le f(q) + d(p, q) = (1 + O(\varepsilon)) f(q)$. Hence, for a small enough ε,

$$\ell_C \le (16\varepsilon + O(\varepsilon^2)) f(p) \le 16\varepsilon (1 + O(\varepsilon))^2 f(q) \le 20\varepsilon f(q).$$

Suppose that $B(\tilde{x}, \frac{1}{16}\ell_C - d(x, \tilde{x})) \cap P \ne \emptyset$. Observe that $B\big(\tilde{x}, \frac{1}{16}\ell_C - d(x, \tilde{x})\big) \subset B\big(x, \frac{1}{16}\ell_C\big) \subset B_\infty\big(x, \frac{1}{16}\ell_C\big)$. As $B_\infty(x, \frac{1}{16}\ell_C)$ is empty by assumption, we conclude that $B\left(\tilde{x}, \frac{1}{16}\ell_C - d(x, \tilde{x})\right) \cap Q = \emptyset$. It follows that $B(\tilde{x}, \frac{1}{16}\ell_C - d(x, \tilde{x})) \cap P \subseteq P \setminus Q$. Let C_Q be the root cell of $\mathcal{T}_Q$. Notice that C contains a point in Q, namely p, and $B_\infty(c, 2.5\ell_C)$ contains a point in $P \setminus Q$, namely a point in $B(\tilde{x}, \frac{1}{16}\ell_C - d(x, \tilde{x})) \cap P$. Therefore, the diameter of $B_\infty(c, 2.5\ell_C)$ is at least $d(Q, P \setminus Q) \ge 2^{-5}\ell_{C_Q}$ by condition (ii) in Definition 4. That is, $5\sqrt{3}\ell_C \ge 2^{-5}\ell_{C_Q}$, or equivalently $\ell_C \ge \ell_{C_Q}/(2^5 5\sqrt{3})$. Therefore, we are in a similar setting as equation (4.2) in the proof of Lemma 4.2. We can apply the argument in the rest of the paragraph following equation (4.2) in the proof of Lemma 4.2 to show that for a small enough ε, $\ell_C \le 20\varepsilon f(q)$ for every point $q \in Q \cap C$. □

Before calling TRIM($\mathcal{T}_Q$), RECONSTRUCT($\mathcal{T}_Q$) has recursively reconstructed from every complete sample that reside in a cluster of $\mathcal{T}_Q$ and subsequently removed such samples. That is, every cluster of the current $\mathcal{T}_Q$ contains a partial sample of exactly one surface by Lemma 4.1. At the beginning of the call TRIM($\mathcal{T}_Q$), Lemma 3.1(iii) implies that every non-empty leaf cell satisfies the size condition in Lemmas 4.2 and 4.3. We will show that if C is trimmed, then the size condition in Lemmas 4.2 and 4.3 will be satisfied again by parent(C). Therefore, Lemmas 4.2 and 4.3 are applicable to parent(C) when we repeat the normal estimation and possibly trimming at parent(C).

Lemma 4.4 below shows several properties of the tree $\overline{\mathcal{T}}_Q$ returned by TRIM($\mathcal{T}_Q$). First, each non-empty leaf cell is small with respect to the local feature sizes at the sample points in it. Second, the non-empty leaf cells are not too small relative to local sampling density. Third, $\overline{\mathcal{T}}_Q$ includes a partial sample on some surface in Σ if and only if TRIM($\mathcal{T}_Q$) returns **false**.

Lemma 4.4. *Consider the call* TRIM($\mathcal{T}_Q$) *for some $Q \subset P$ invoked by* RECONSTRUCT. *Let $\overline{\mathcal{T}}_Q$ be the tree returned by* TRIM($\mathcal{T}_Q$). *Assume that ε is small enough.*

(i) *The call runs in $O(|\mathcal{T}_Q|)$ time. For every non-empty leaf cell C of $\overline{\mathcal{T}}_Q$ and every point $q \in Q \cap C$, $\ell_C \le 50\varepsilon f(q)$. Hence, the sample points in C form a partial sample of exactly one surface.*

(ii) *Let x be a point in Σ. Let p be the nearest point in $\overline{\mathcal{T}}_Q$ to x. If $d(p, x) = O(\varepsilon f(x))$ and $\overline{\mathcal{T}}_Q$ contains more than one node, the non-empty leaf cell C in $\overline{\mathcal{T}}_Q$ that contains p satisfies $d(p, x) < 2\ell_C$.*

(iii) *If* TRIM($\mathcal{T}_Q$) *returns* **false**, *then the sample points in $\overline{\mathcal{T}}_Q$ form a partial sample of exactly one surface. Conversely, if $\overline{\mathcal{T}}_Q$ contains a partial sample of a surface, then the sample points in $\overline{\mathcal{T}}_Q$ form a partial sample of that surface only and* TRIM($\mathcal{T}_Q$) *returns* **false**.

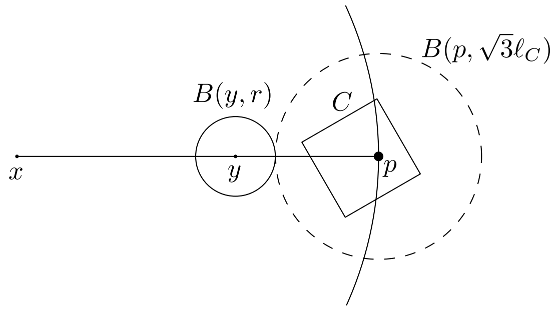


Figure 7: Proof of Lemma 4.4(ii).

Proof. Consider (i). The running time follows from the description of the algorithm. At the beginning of the call TRIM($\mathcal{T}_Q$), every non-empty leaf cell C' satisfy $\ell_{C'} \leq 160\varepsilon f(q)$ for every point $q \in Q \cap C'$ by Lemma 3.1(iii), satisfying the conditions in Lemmas 4.2 and 4.3. These two lemmas imply that $\ell_{C'} \leq 20\varepsilon f(q)$ for every $q \in Q \cap C'$. Therefore, $\ell_{\text{parent}(C')} = 2\ell_{C'} \leq 40\varepsilon f(q)$ for every point $q \in Q \cap C'$. For every point q' in parent(C') but not in C', $d(q, q') \leq 2\sqrt{3}\ell_{C'} = O(\varepsilon f(q))$. Therefore, $f(q') \geq f(q) - d(q, q') \geq (1 - O(\varepsilon))f(q)$, which implies that for a small enough ε,

$$\ell_{\text{parent}(C')} \leq 40\varepsilon f(q) \leq 40(1 + O(\varepsilon))f(q') < 50\varepsilon f(q').$$

Hence parent(C'), being a new non-empty leaf cell, satisfy the size conditions in Lemmas 4.2 and 4.3. Inductively, we conclude that $\ell_C \leq 50\varepsilon f(q)$ for every non-empty leaf cell C of $\overline{\mathcal{T}}_Q$ and for all $q \in Q \cap C$.

The size of C is too small for it to contain a complete sample. So C contains a partial sample of a surface. If C contains sample points p and q from two different surfaces, then $d(p, q) \geq f(q)$ because the line segment pq must then intersect the medial axis of Σ. But this is impossible when ε is sufficiently small because $\ell_C \leq 50\varepsilon f(q)$.

Consider (ii). Suppose that $d(p, x) \geq 2\ell_C$, where C is the non-empty leaf cell of $\overline{\mathcal{T}}_Q$ that contains p. We will show that C is the only node in $\overline{\mathcal{T}}_Q$, a contradiction to the assumption that $\overline{\mathcal{T}}_Q$ contains more than one node.

Refer to Figure 7. Let y be the point on the segment px such that $d(p, y) = \sqrt{3}\ell_C + r$, where $r = (1 - \sqrt{3}/2)\ell_C$. The distance between any point in $B(y, r)$ and C is at most $r + d(p, y) = 2r + \sqrt{3}\ell_C = 2\ell_C$. Therefore, $B(y, r) \subset B_\infty(c, 2.5\ell_C)$, where c is the center of C.

Let L be the plane tangent to Σ at p. Let y' be the projection of y in L. Since $d(p, x) = O(\varepsilon f(x))$, $f(x) \leq 2f(p)$ for a small enough ε. Therefore, $d(p, x) = O(\varepsilon f(x)) = O(\varepsilon f(p))$ and Lemma 2.1(i) implies that px makes an $O(\varepsilon)$ angle with L. This gives $d(y, y') = O(\varepsilon d(p, y)) = O(\varepsilon \ell_C)$, which is less than $r - (\sqrt{3}/16)\ell_C$ for a small enough ε. So

$$B_\infty\Big(y', \frac{1}{16}\ell_C\Big) \subset B\Big(y', \frac{\sqrt{3}}{16}\ell_C\Big) \subseteq B\Big(y, \frac{\sqrt{3}}{16}\ell_C + d(y, y')\Big) \subseteq B(y, r) \subset B_\infty(c, 2.5\ell_C).$$

Recall the partition of $B_\infty(c, 2.5\ell_C)$ into the set $\{R_i : 1 \leq i \leq 5^3\}$ of canonical cubes of side length ℓ_C. If R_i intersects $B_\infty(y', \frac{1}{16}\ell_C)$, it also intersects $B(y, r)$, which implies that

$$R_i \subset B(y, \sqrt{3}\ell_C + r) = B(y, d(p, y)) \subset B(x, d(p, x)).$$

Since p is the nearest point of x in Q, $B(x, d(p, x)) \cap Q = \emptyset$. It follows that $R_i \cap Q = \emptyset$. So $B_\infty(y', \frac{1}{16}\ell_C)$ is contained in the union of the empty R_i's. But then this should have caused the leaf cell C to be trimmed unless C is the only node in $\overline{\mathcal{T}}_Q$. This completes the proof of (ii).

Consider (iii). Suppose that $\textsc{Trim}(\mathcal{T}_Q)$ returns **false**. In this case, $\overline{\mathcal{T}}_Q$ contains the root node only. By (i), the root (and hence $\overline{\mathcal{T}}_Q$) contains the partial sample of exactly one surface.

As for the direction, suppose that $\overline{\mathcal{T}}_Q$ contains a partial sample of some surface in Σ. Then, Q is a proper subset of P, so $Q = Q_0 \cap X$ for some cluster X of some octree $\mathcal{T}_{Q_0}$. Let C_X be the root cell of $\mathcal{T}_Q$ and hence of $\overline{\mathcal{T}}_Q$ too. Since the sample points in $\overline{\mathcal{T}}_Q$ contains a partial sample, a surface in Σ contains a sample point in X and a sample point outside X. Then, Lemma 3.1(iii) implies that $\ell_{C_X} \le 160\varepsilon f(q)$ for every point in $q \in Q_0 \cap X = Q$. Note that all sample points in Q lie in C_X. Let x be a point in Σ at distance $2\ell_{C_X}$ from C_X. Then for all $q \in Q$,

$$d(q,x) \le d(x, C_X) + \sqrt{3}\ell_{C_X} = (2+\sqrt{3})\ell_{C_X} < 600\varepsilon f(q).$$

The 1-Lipschitzness of f implies that $f(q) \le f(x) + d(q,x) = f(x) + O(\varepsilon))f(q)$. Therefore, $d(q,x) = O(\varepsilon f(x))$. Take an arbitrary sample point $q \in Q$. Let C be the leaf cell of $\overline{\mathcal{T}}_Q$ that contains q. C_X contains C as C_X is the root cell. Since x lies outside C_X, we have

$$d(x, C) \ge d(c, C_X) = 2\ell_{C_X} \ge 2\ell_C.$$

This is the negation of the implication of (ii). Therefore, one of the two if-conditions is not satisfied. Since $d(q,x) = O(\varepsilon f(x))$, we conclude that $\overline{\mathcal{T}}_Q$ must have only one node, i.e., $\textsc{Trim}(\mathcal{T}_Q)$ returns **false**. Since $\overline{\mathcal{T}}_Q$ contains only the root cell, we conclude by (i) that the sample points in $\overline{\mathcal{T}}_Q$ form a partial sample of exactly one surface. □

By Lemma 4.4(iii), in line 8 of $\textsc{Reconstruct}(\mathcal{T}_Q)$, if $\textsc{Trim}(\mathcal{T}_Q)$ returns **true**, the sample points in $\overline{\mathcal{T}}_Q$ must form complete sample(s) of some surface(s). Indeed, if $\overline{\mathcal{T}}_Q$ contains a partial sample, then $\textsc{Trim}(\mathcal{T}_Q)$ would have returned **false** by Lemma 4.4(iii). Therefore, it is correct to reconstruct the surfaces from the samples in lines 10-11 in $\textsc{Reconstruct}(\mathcal{T}_Q)$. Conversely, if the sample points in $\overline{\mathcal{T}}_Q$ form complete sample(s) of some surface(s), then $\textsc{Trim}(\mathcal{T}_Q)$ cannot return **false** because the sample points in $\overline{\mathcal{T}}_Q$ would form a partial sample otherwise by Lemma 4.4(iii).

4.2 Subsample

In line 10 of $\textsc{Reconstruct}(\mathcal{T}_Q)$, we need to extract a locally uniform sample $U_Q \subseteq Q$ by calling $\textsc{Extract}(\overline{\mathcal{T}}_Q)$ in Algorithm 4. Let $\eta = 3\kappa_{\mathrm{uni}}$, where κ_{uni} is the constant used in Definition 2 (locally uniform sample). In $\textsc{Extract}$, we examine non-empty leaf cells in non-increasing sizes. For each non-empty leaf cell C examined, we find all the cells C' in the current tree such that $\ell_{C'} = \frac{1}{2}\ell_C$ and C' intersects $B_\infty(c, \eta\ell_C)$. There are only $O(\eta^3) = O(1)$ such cells C', and they can be found in $O(1)$ time using the pointers between neighboring octree cells of the same size and the parent-child pointers. For every such C', we make it a new leaf by discarding its descendants. Upon the completion of this tree traversal, we pick an arbitrary point from each remaining non-empty leaf cell to form a locally uniform sample.

Lemma 4.5. *$\textsc{Extract}(\overline{\mathcal{T}}_Q)$ computes in $O(|\overline{\mathcal{T}}_Q|)$ time a locally uniform $O(\varepsilon)$-sample $U_Q \subseteq Q$ of the surfaces in Σ that contain the sample points in $\overline{\mathcal{T}}_Q$.*

Proof. It is clear that $\textsc{Extract}$ runs in $O(|\overline{\mathcal{T}}_Q|)$ time. It remains to show that U_Q is a locally uniform $O(\varepsilon)$-sample.

We first show that U_Q is an $O(\varepsilon)$-sample. Take any point x on the surface that contains a sample point in $\overline{\mathcal{T}}_Q$. We need to identify a point in U_Q at distance $O(\varepsilon f(x))$ from x.

Since $\textsc{Trim}(\mathcal{T}_Q)$ succeeds before the invocation of $\textsc{Extract}$, by Lemma 4.4(iii), Q form complete sample(s) for some surface(s). So Q is an ε-sample of these surfaces. Let p be a sample point in Q such that $d(p,x) \le \varepsilon f(x)$. If p is included in U_Q, we are done. Suppose not. Let C_1 be the non-empty leaf cell in the final $\widehat{\mathcal{T}}_Q$ that contains p. Let $(C_k, C_{k-1}, \ldots, C_1)$ be the

Algorithm 4 Extract($\overline{\mathcal{T}}_Q$)

```
1: $\widehat{\mathcal{T}}_Q \leftarrow \overline{\mathcal{T}}_Q$
2: Start a breadth-first search of $\widehat{\mathcal{T}}_Q$ at its root.
3: while the breadth-first search has not finished do
4:     $C \leftarrow$ current cell encountered in the breadth-first search
5:     if $C$ is a non-empty leaf cell then
6:         $c \leftarrow$ center of $C$
7:         for all octree cell $C'$ in $\widehat{\mathcal{T}}_Q$ such that $\ell_{C'} = \frac{1}{2}\ell_C$ and it intersects $B_\infty(c, \eta\ell_C)$ do
8:             MakeLeaf($C'$)
9:         end for
10:    end if
11: end while
12: $U_Q \leftarrow \emptyset$
13: for all non-empty leaf cell $C$ of $\widehat{\mathcal{T}}_Q$ do
14:     $p \leftarrow$ arbitrary point in $Q \cap C$
15:     $U_Q \leftarrow U_Q \cup \{p\}$
16: end for
17: return $U_Q$
```

maximal sequence of octree cells of T_Q such that for $i \in [2, k]$, the processing of C_i made C_{i-1} a non-empty leaf cell of $\widehat{\mathcal{T}}_Q$ during the execution of Extract. Notice that C_k is a non-empty leaf cell of the input parameter $\overline{\mathcal{T}}_Q$ of Extract. For $i \in [2, k]$, $\ell_{C_{i-1}} = \frac{1}{2}\ell_{C_i}$, and the distance between any two points in C_{i-1} and C_i is at most

$$\sqrt{3}\eta\ell_{C_i} + \frac{\sqrt{3}}{2}\ell_{C_i} + \sqrt{3}\ell_{C_{i-1}} = \sqrt{3}(\eta + 1)\ell_{C_i}.$$

This implies that the distance between any two points in C_1 and C_k is at most

$$\sum_{i=2}^{k} \sqrt{3}(\eta + 1)\ell_{C_i} < 2\sqrt{3}(\eta + 1)\ell_{C_k}.$$

By Lemma 4.4(i), $\ell_{C_k} \leq 50\varepsilon f(q)$, where q is a sample point in $Q \cap C_k$. Then $d(p, q) < 2\sqrt{3}(\eta + 1)\ell_{C_k} \leq 100\sqrt{3}(\eta+1)\varepsilon f(q)$. The Lipschitzness of f implies that $d(p, q) = O(\varepsilon f(q))$. Let q' be the sample point in C_1 that is included in U_Q. We can reason as before to obtain $d(q, q') = O(\varepsilon f(q))$, which further implies that $f(q) = O(f(q'))$. As a result, $d(p, q') \leq d(p, q) + d(q, q') = O(\varepsilon f(q'))$. Therefore,

$$d(q', x) \leq d(p, x) + d(p, q') \leq \varepsilon f(x) + O(\varepsilon f(q')).$$

Using $f(q') \leq f(x) + d(q', x)$, we obtain $d(q', x) = O(\varepsilon f(x))$. Thus, U_Q is an $O(\varepsilon)$-sample.

We prove the local uniformity of U_Q. (Refer to Definition 2.) For every sample point $p \in U_Q$, we need to show that $|B(p, \kappa_{\text{uni}} d(p, x)) \cap U_Q| = O(1)$ for all point $x \in \Sigma$ such that $B(x, d(p, x)) \cap U_Q = \emptyset$. Since $B(x, d(p, x)) \cap U_Q = \emptyset$, p is the nearest sample point in U_Q to x. We have $d(p, x) = O(\varepsilon f(x))$ as U_Q is an $O(\varepsilon)$-sample. Let C and C' be the non-empty leaf cells in $\widehat{\mathcal{T}}_Q$ and $\overline{\mathcal{T}}_Q$, respectively, that contain p. Notice that $C' \subseteq C$ as $\widehat{\mathcal{T}}_Q$ is a subtree of $\overline{\mathcal{T}}_Q$. By Lemma 4.4(ii),

$$d(p, x) \leq 2\ell_{C'} \leq 2\ell_C. \tag{4.3}$$

Consider an arbitrary non-empty leaf cell C_1 of $\widehat{\mathcal{T}}_Q$ that intersects $B(p, \kappa_{\text{uni}} d(p,x))$. Let c be the center of C. We have

$$d(c, C_1) < \kappa_{\text{uni}} d(p,x) + d(p,c) \le \kappa_{\text{uni}} d(p,x) + \frac{\sqrt{3}}{2}\ell_C < (2\kappa_{\text{uni}} + 1)\ell_C,$$

which is less than $\eta\ell_C$ because $\kappa_{\text{uni}} > 1$ and $\eta = 3\kappa_{\text{uni}}$. This implies that C_1 intersects $B_\infty(c, \eta\ell_C)$. We conclude that $\ell_{C_1} \ge \frac{1}{2}\ell_C$; otherwise, C_1 would have been removed in lines 7–10 of Extract. Then, a packing argument shows that $O(1)$ non-empty leaf cells of $\widehat{\mathcal{T}}_Q$ can intersect $B(p, \kappa_{\text{uni}} d(p,x))$, meaning that $|B(p, \kappa_{\text{uni}} d(p,x)) \cap U_Q| = O(1)$ as U_Q contains one sample point from each non-empty leaf cell of $\widehat{\mathcal{T}}_Q$. $\square$

4.3 Putting everything together

We summarize the algorithm and fill in some details in high level description of the reconstruction algorithm in the overview in Section 2.2. Recall that there are two high-level steps:

- Step 1: Select a subset $U_P \subseteq P$ such that U_P is a locally uniform $O(\varepsilon)$-sample of Σ.
- Step 2: For each sample point $p \in U_P$, estimate the surface normal at p and construct the Delaunay triangles incident to p (with respect to U_P) such that the dual Voronoi edge of each such triangle τ intersects the cocones at all three vertices of τ. Then, traverse all such triangles to extract a surface triangulation M by a simple linear time search [3].

We preprocess the input ε-sample P to recursively construct the "hierarchy" of octree trees as described in Section 3. By Lemma 3.4, the octrees can be constructed in $O(|P|\log|P|)$ time and they require $O(|P|)$ storage. (The proof of Lemma 3.4 is given in Section 6.)

Afterwards, we call Reconstruct($\mathcal{T}_P$) to perform the reconstruction. In contrast with the two high-level steps in Section 2.2, Reconstruct does not extract a locally uniform sample from P all at once. Instead, by the recursive nature of Reconstruct, locally uniform samples of surfaces in Σ are extracted from P at different times. Specifically, in Reconstruct, a locally uniform sample of a surface is extracted in line 10 and then the corresponding surface is reconstructed in line 11 by invoking step 2 described above. By Lemmas 4.4 and 4.5, the octrees can be processed in $O(|P|)$ time to extract locally uniform samples of all surfaces. We explain below how to run step 2 above in time linear in the sizes of the locally uniform samples. Then, our main result follows: a faithful reconstruction of Σ can be computed in $O(|P|\log|P|)$ time. (Faithful reconstruction is defined in Definition 3.)

Let $Q \subset P$ denote the subset of samples that lie on a surface in Σ. Let U_Q be a locally uniform sample extracted from Q obtained in line 10 of Reconstruct. Let $\text{Del}(U_Q)$ denote the Delaunay triangulation of U_Q. For every point $p \in U_Q$, we use *C_p-triangles* to refer to the set of triangles in $\text{Del}(U_Q)$ incident to p such that the dual Voronoi edge of each such triangle τ intersects the cocones at all three vertices of τ. It suffices to describe how to compute the C_p-triangles in order to run step 2.

By the results in [3], Delaunay triangles of U_Q restricted to Σ are C_p-triangles, the normal of every C_p-triangle makes an $O(\varepsilon)$ angle with $\mathbf{n}_p$, and the angle between the supporting planes of every pair of C_p-triangles is $O(\varepsilon)$.

Let τ be a C_p-triangle. Since the dual Voronoi edge of τ intersects the cocone at p, there exists a circumball B of τ that is empty of vertices in U_Q such that the center c_B of B lies in the cocone of p. Let H_τ denote the plane of τ. Since c_B lies in the cocone of p, H_τ makes an angle at most $\pi/8 + O(\varepsilon)$ with the ray from p through c_B. Therefore, for a small enough ε, we have

$$\begin{aligned} d(c_B, H_\tau) &< \frac{1}{2}\text{radius}(B), \\ \text{circumradius}(\tau) &= \Omega(\text{radius}(B)). \end{aligned}$$

Consider the ray from p through the circumcenter of τ. Project this ray in the normal direction of τ onto a restricted Delaunay triangle incident to p, say t. Let H_t denote the plane of t. Let γ_t denote the projected ray in H_t. The ray γ_t makes an $O(\varepsilon)$ angle with the diameter of the disk $B \cap H_t$ incident to p. Since γ_t cuts through the interior of t and $B \cap H_t$ is empty of points in U_Q, the circumcircle of t must contain the segment $\gamma_t \cap B \cap H_t$. This implies that

$$\text{circumradius}(t) \geq \frac{1}{2}\,\text{length}(\gamma_t \cap B \cap H_t) = \Omega(\text{radius}(B \cap H_t)).$$

Since $d(c_B, H_\tau) < \frac{1}{2}\text{radius}(B)$ and H_t makes an $O(\varepsilon)$ angle with H_τ, we have $d(c_B, H_t) < 0.6\,\text{radius}(B)$ for a small enough ε. It implies that $\text{radius}(B \cap H_t) = \Omega(\text{radius}(B))$. This allows us to bound the circumradius of τ:

$$\text{circumradius}(t) = \Omega(\text{radius}(B \cap H_t)) = \Omega(\text{radius}(B)) = \Omega(\text{circumradius}(\tau)).$$

Since t is a restricted Delaunay triangle incident to p, there is a circumball B' of t that is centered at a point in Σ. The ball B' is empty of points in U_Q. Let r'_p be the radius of the largest ball that is centered at a point in Σ, contains p in its boundary, and does not contain any point of U_Q in its interior. Therefore,

$$r'_p \geq \text{radius}(B') \geq \text{circumradius}(t) = \Omega(\text{circumradius}(\tau)).$$

By the above discussion, there exists constants $c \geq 1$ and $\kappa_{\text{uni}} \geq c$ such that the C_p-triangles satisfy the following properties:

(i) connect p to points in U_Q that are within a distance of $\frac{\kappa_{\text{uni}}}{2c} r'_p$ from p,

(ii) have circumradii no more than $\frac{\kappa_{\text{uni}}}{2c} r'_p$, and

(iii) have circumballs empty of points in U_Q with radii at most $\frac{\kappa_{\text{uni}}}{2} r'_p$.

Define $V_p = \{q \in U_Q : d(p,q) \leq \kappa_{\text{uni}} r'_p\}$. Then, we only need to compute triangles in $\text{Del}(U_Q)$ that connect p to points in V_p. By property (iii) above, the Delaunayhood of any triangle that satisfies properties (i) and (ii) above can be checked for Delaunayhood using the points in V_p only (instead of the full set U_Q). Consequently, it suffices to compute the Delaunay triangulation of V_p, denoted by $\text{Del}(V_p)$, which must contain all C_p-triangles. By locally uniformity, $|V_p| = O(1)$. Thus, $\text{Del}(V_p)$ can be computed in $O(1)$ time, and the triangles that satisfy properties (i)–(iii) above can be extracted from $\text{Del}(V_p)$ in $O(1)$ time.

It is inconvenient to retrieve the exact V_p using our octrees. Fortunately, any superset of V_p of $O(1)$ size that is within a distance at least $\kappa_{\text{uni}} r'_p$ from p works fine too. We retrieve such a superset of V_p in $O(1)$ time as follows. By (4.3), for all $x \in \Sigma$ such that $B(x, d(p,x)) \cap U_Q = \emptyset$, $d(p,x) \leq 2\ell_C$, where C is the leaf cell in the final $\widehat{\mathcal{T}}_Q$ that contains p. It follows that $r'_p \leq 2\ell_C$. So $B(p, \kappa_{\text{uni}} r'_p) \subseteq B(p, 2\kappa_{\text{uni}}\ell_C) \subset B_\infty(c, \eta\ell_C)$, where c is the center of C and $\eta = 3\kappa_{\text{uni}}$. As a result, the points in V_p are contained in leaf cells of the final $\widehat{\mathcal{T}}_Q$ that lie inside $B_\infty(c, \eta\ell_C)$. By lines 7 and 8 of Extract, every cell in the final $\widehat{\mathcal{T}}_Q$ that intersects $B_\infty(c, \eta\ell_C)$ has side length at least $\frac{1}{2}\ell_C$. Therefore, we are looking for $O(1)$ such leaf cells and we can find them using pointers between neighboring octree cells of the same size and the parent-child pointers. Then, we can find the sample points of U_Q in the cells identified in $O(1)$ time. These sample points contain $B(p, \kappa_{\text{uni}} r_p) \cap U_Q$ and hence form a superset of V_p of $O(1)$ size. Denote this superset by W_p.

We compute $\text{Del}(W_p)$ and extract the triangles incident to p that have empty circumballs with radii at most $\kappa_{\text{uni}} \ell_C$. Because such circumballs are contained in $B(p, 2\kappa_{\text{uni}} \ell_C) \subset B_\infty(c, \eta \ell_C)$, the extracted triangles from $\text{Del}(W_p)$ are Delaunay triangles in $\text{Del}(U_Q)$. Since $r'_p \leq 2\ell_C$, the triangles extracted include all C_p-triangles. For each triangle τ extracted from $\text{Del}(W_p)$, it remains to whether the dual Voronoi edge of τ intersects the cocones at all three vertices of τ. This requires us to compute the cocones at each $p \in U_Q$, which boils down to estimating the surface normals at each $p \in U_Q$. Among the triangles extracted, let τ' be the one with the minimum circumradius. Since all C_p-triangles are also extracted, by property (ii), the circumradius of τ' is no more than $\frac{\kappa_{\text{uni}}}{2c} r'_p = O(\varepsilon f(p))$ as $r'_p = O(\varepsilon f(p))$. When ε is sufficiently small, by Lemma 2.1(ii), the normal of τ' makes an $O(\varepsilon)$ angle with $\mathbf{n}_p$, and so the normal of τ' is a good estimation of $\mathbf{n}_p$.

Theorem 4.1. *Let P be an ε-sample of a closed two-dimensional manifold $\Sigma \subset \mathbb{R}^3$ for a sufficiently small ε. A faithful reconstruction of Σ can be computed in $O(|P| \log |P|)$ time.*

5 Experimental results

Our implementation differs slightly from the preceding description. In line 10 of Trim, we use $B_\infty(c, 1.5\ell_C)$ instead of $B_\infty(c, 2.5\ell_C)$. For normal estimation, we pick two points from each of the canonical cubes of side length ℓ_C in $B_\infty(c, 1.5\ell_C)$, and then use PCA to estimate the surface normal. The PCA-based approach improves the robustness of the surface normal estimation. We perform an extra test in line 24 of Trim before trimming a cell C. We use the estimated surface normal to form a cocone at $p \in C$ of angular radius $\pi/12$. We trim C only if an empty cube is found in line 24 of Trim and this cocone at p contains all sample points in $B_\infty(c, 1.5\ell_C)$. We need this cocone check because some test cases do not meet the sampling requirement of the algorithm. If the cocone at p excludes some sample point in $B_\infty(c, 1.5\ell_C)$, the surface around p is curvy with respect to the local sampling density, so we should not trim C. In line 7 of Extract, we set $\eta = 2$. In lines 13–16 of Extract, for each non-empty leaf cell C in $\widehat{\mathcal{T}}_P$, if C is also a leaf cell in $\overline{\mathcal{T}}_P$, then sample one point in C as described in the algorithm; otherwise, we sample one point from each child of C in $\overline{\mathcal{T}}_P$ instead of C alone. This gives us a denser subsample to facilitate the subsequent surface reconstruction. This implementation still runs in $O(n \log n)$ time.

We run Cocone to reconstruct the first surface from the subsample. Afterwards, the sample points absent from the subsample are added back as follows. For each remaining sample point p, we find the nearest triangle τ in the current reconstruction, connect p to the three sides of τ to split τ into three triangles, and then apply edge flips until no new edge produced is flippable. (The common edge pq between two triangles pqr and pqs is flippable iff the diametric ball of pqr contains s and the diametric ball of pqs contains r. Refer to [15] for theoretical results on edge flips.) We compile our code using `g++-4.1.2` with `O2` flag, and obtain the binary of Cocone from its author's webpage. The experiments were run on a Dell Optiplex 745 with a 4GB RAM and an Intel Core 2 Duo E4600 processor (2.4GHz, 2MB L2 cache, and 800MHz FSB). Floating point number type is used in our code for extracting a locally uniform subsample. Cocone uses filtered predicates that simulate exact arithmetic in an on-demand fashion.

The top table in Table 1 shows the results of our experiments on samples that are not locally uniform. Figure 8 shows the samples and models used. For each model, we randomly pick a subset of triangles, sample extra points in those triangles and their neighboring triangles, and then include the vertices of the model to form a locally non-uniform input sample. The white

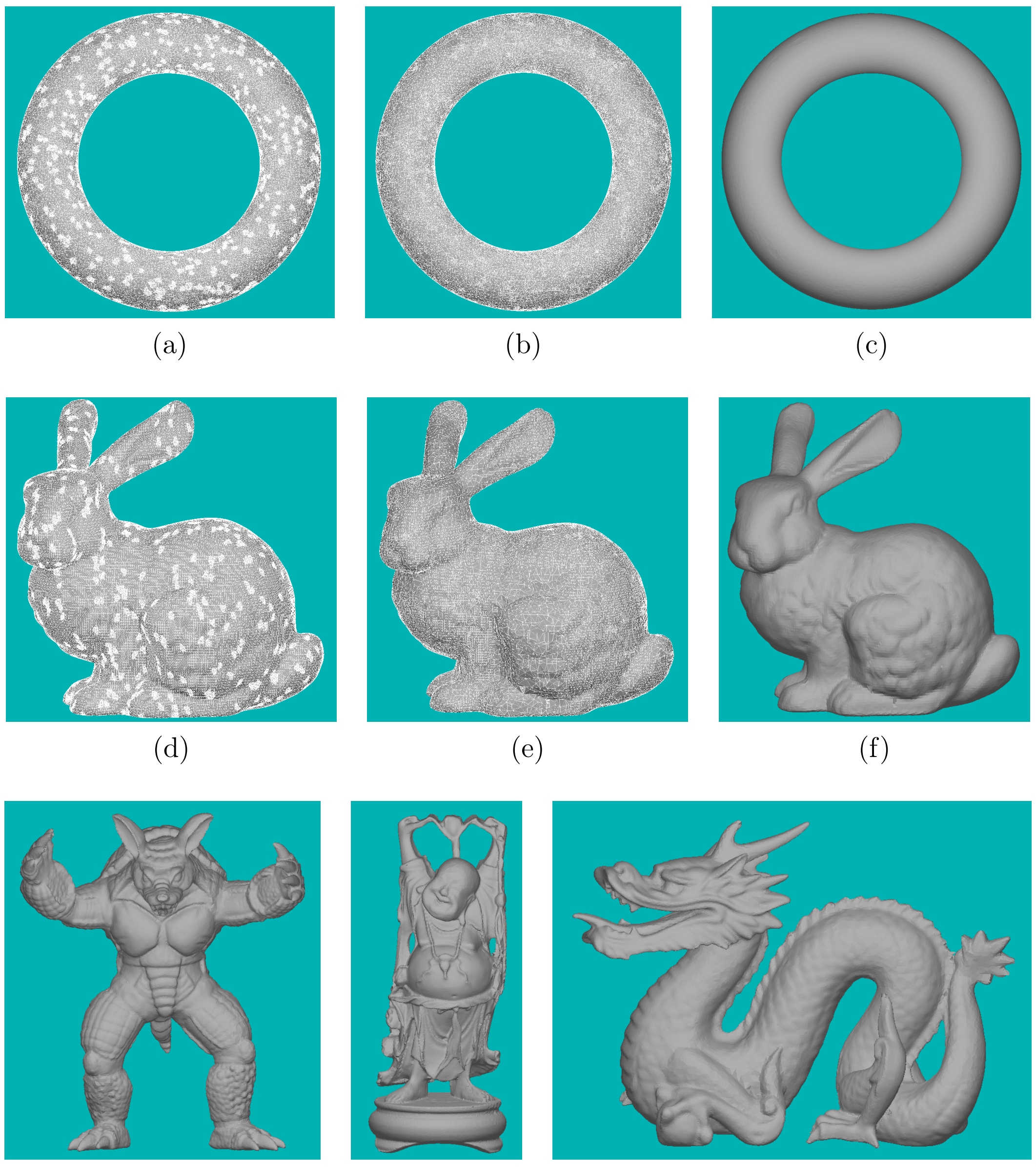


Figure 8: (a) and (d) show the input non-uniform samples. (b) and (e) show the subsamples extracted. (c) and (f)–(i) show the surfaces reconstructed by our prototype.

Non-uniform	I	Cocone	In	Crad	S	Prune+Build	Sn	Total	Srad
Torus	348820	224s	101	23	70038	7s+44s	36	74s	23
Bunny	210646	100s	589	36	29242	4s+14s	28	33s	36
Armadillo	411777	216s	171	23	112593	9s+59s	31	93s	23
Buddha	512524	299s	261	19	143021	11s+79s	72	121s	19
Dragon	452679	256s	1177	22	80989	9s+41s	69	81s	23

Uniform	I	Cocone	In	Crad	S	Prune+Build	Sn	Total	Srad
Torus	57600	36s	5	74	53956	2s+25s	13	28s	74
Bunny	35286	20s	21	116	23503	1s+11s	23	13s	117
Armadillo	115897	60s	24	52	73706	3s+37s	41	45s	54
Buddha	142534	77s	62	42	112076	4s+62s	60	71s	43
Dragon	126359	66s	308	50	58439	3s+30s	84	39s	52

Table 1: The top and bottom tables show non-uniform and locally uniform samples respectively. `I`: input size; `Cocone`: Cocone's running time on the input; `In`: maximum no. of sample points within $1.5R_v$ from a vertex v, where R_v is the maximum circumradius among its incident triangles in the Cocone output; `Crad`: average circumradius in the Cocone output; `S`: subsample size; `Prune+Build`: time to extract the subsample + time to reconstruct from it using Cocone; `Sn`: maximum no. of subsample points within $1.5S_v$ from a vertex v, where S_v is the maximum circumradius among its incident triangles after running Cocone on the subsample; `Total`: prototype's running time; `Srad`: average circumradius in the prototype's output.

patches in Figures 1(a) and 1(d) are the resulting clusters of sample points. Our prototype is 51% to 68% faster than running Cocone alone. Extracting a subsample (i.e., `Prune`) takes about 1/10 of the total running time. The columns `In` and `Sn` show that the subsamples are significantly more locally uniform than the input samples. The columns `Crad` and `Srad` show that the average circumradii in Cocone's output and our prototype's output are similar.

We also experimented with some locally uniform samples. The results are shown in the bottom table in Table 1. Our prototype is about 8% faster than running Cocone alone for Buddha, and 22% to 41% faster for other models. Note that most of the time is spent on running Cocone to reconstruct the surfaces from the extracted subsample (i.e., `Build`).

6 Proof of Lemma 3.4

Let n denote $|P|$. A few technical results are needed before we give the proof of Lemma 3.4. First, Lemma 3.2(ii) implies that there are $O(n)$ octrees and there are $O(n)$ splittable cells in all octrees.

Lemma 6.1. *There are $O(n)$ octrees and there are $O(n)$ splittable cells in them.*

To bound the total size of the octrees, our plan is to charge each cell to some splittable cell. Consider the octree $\mathcal{T}_Q$ for some $Q \subseteq P$. The construction of $\mathcal{T}_Q$ can be divided into stages according to the applications of the splitting and balancing rules. Let $\mathcal{F}_0 = \tilde{\mathcal{F}}_0$ be the octree after the initialization when building $\mathcal{T}_Q$. For $i \geq 1$, let $\mathcal{F}_i$ be the octree after the ith round of applications of the splitting rule to the octree $\tilde{\mathcal{F}}_{i-1}$. So the leaf cells of $\mathcal{F}_i$ are non-splittable. For $i \geq 1$, let $\tilde{\mathcal{F}}_i$ be the octree after the ith round of applications of the balancing rule. It is the result of balancing $\mathcal{F}_i$. Lemma 6.2 below shows a gradation in side lengths as we move away from a leaf cell of $\tilde{\mathcal{F}}_i$.

For any integer $j \geq 0$ and any octree cell C, let $R_j(C)$ be the box with the same center as C and side length $(3 - 2^{1-j})\ell_C$. For example, $R_0(C) = C$ and $R_\infty(C)$ is the box with the same center as C and side length $3\ell_C$ (i.e., the union of C and its neighboring canonical boxes of side length ℓ_C). For $j \geq 1$, the annulus $R_j(C) \setminus R_{j-1}(C)$ has width $2^{-j}\ell_C$.

Lemma 6.2. *Let C be a leaf cell of $\tilde{\mathcal{F}}_i$. For any $j \geq 0$, if a cell in $\tilde{\mathcal{F}}_i$ intersects $R_j(C)$, its side length is at least $2^{-j}\ell_C$.*

Proof. Any neighboring leaf cell of C in $\tilde{\mathcal{F}}_i$ has side length at least $\frac{1}{2}\ell_C$, so their union together with C covers $R_1(C)$. Inductively, one can show that the union of all the leaf cells in $\tilde{\mathcal{F}}_i$ that have side lengths at least $2^{-j}\ell_C$ covers $R_j(C)$ for any j. The result also holds for non-leaf cells intersecting $R_j(C)$ as they can only be bigger. ◻

During the ith round of applications of the balancing rule, a split of a cell may trigger further splits of its neighboring cells. We show in the next lemma that a cell in $\tilde{\mathcal{F}}_i \setminus \mathcal{F}_i$ cannot be split by the balancing rule if the cells of $\mathcal{F}_i$ nearby are large.

Lemma 6.3. *Let C be a cell in $\tilde{\mathcal{F}}_i$. Assume that C is non-splittable or C is a cell in $\tilde{\mathcal{F}}_i \setminus \mathcal{F}_i$. If every splittable cell of $\mathcal{F}_i$ that intersects $R_j(C)$ has side length at least $2^{1-j}\ell_C$ for all $j \geq 0$, then C is a leaf cell of $\tilde{\mathcal{F}}_i$.*

Proof. We prove the lemma by induction in non-decreasing cell sizes in $\tilde{\mathcal{F}}_i$. The claim is clearly true when C is the smallest cell of $\tilde{\mathcal{F}}_i$ because it is a leaf cell. Suppose inductively that the result holds for all cells smaller than C in $\tilde{\mathcal{F}}_i$ that are non-splittable or cells in $\tilde{\mathcal{F}}_i \setminus \mathcal{F}_i$. Since C is non-splittable or a cell in $\tilde{\mathcal{F}}_i \setminus \mathcal{F}_i$, it can only be split by the balancing rule. We show that for all cell C' in $\tilde{\mathcal{F}}_i$ that is a neighboring of C and has side length $\frac{1}{2}\ell_C$, C' is a leaf cell of $\tilde{\mathcal{F}}_i$. Then C is also a leaf cell because it is not out of balance.

Take any neighboring cell C' of C in $\tilde{\mathcal{F}}_i$ that has side length $\ell_{C'} = \frac{1}{2}\ell_C$. Since C' intersects $R_1(C)$ and $\ell_{C'} < \ell_C$, by the assumption of the lemma, C' is non-splittable or a cell in $\tilde{\mathcal{F}}_i \setminus \mathcal{F}_i$. For any $j \geq 0$, $R_j(C') \subseteq R_{j+1}(C)$. So any splittable cell in $\mathcal{F}_i$ that intersects $R_j(C')$ also intersects $R_{j+1}(C)$. Therefore, by the assumption of the lemma, such cells have size at least $2^{1-(j+1)}\ell_C = 2^{1-j}\ell_{C'}$. Hence, C' satisfies the condition of the lemma, and so C' is a leaf cell by induction assumption. ◻

Next, we show that for every cell of $\mathcal{T}_Q$, if there is no sizable splittable cell near it, the subtree rooted at that cell has $2^{O(d)}$ levels. This result allows us to charge each non-splittable cell to some splittable cell so that a splittable cell only gets $O(1)$ charges.

Lemma 6.4. *Let $m = 3^d$. Let C be a cell of $\mathcal{T}_Q$ that is a proper descendant of a leaf cell of $\tilde{\mathcal{F}}_0$, the initial octree when we begin to build $\mathcal{T}_Q$. If no splittable cell in $\mathcal{T}_Q$ that intersects $R_\infty(C)$ has side length in the range $[2^{-m}\ell_C, 2\ell_C]$, then the subtree rooted at C has at most $m-1$ levels.*

Proof. By assumption, both C and its parent are non-splittable because they intersect $R_\infty(C)$ and have side lengths in the range $[2^{-m}\ell_C, 2\ell_C]$. Therefore, C must be created by the balancing rule. Assume that C is created during the construction of $\tilde{\mathcal{F}}_{i_0}$ for some $i_0 > 0$.

We first show that no splittable cell of $\mathcal{F}_{i_0}$ that intersects $R_\infty(C) \setminus C$ has side length strictly less than $2^{-m}\ell_C$. For the sake of contradiction, let D be a splittable cell in $\mathcal{F}_{i_0}$ that has side length less than $2^{-m}\ell_C$ and intersects $R_\infty(C) \setminus C$. Let C' and D' be the leaf cells in $\tilde{\mathcal{F}}_{i_0-1}$ that are ancestors of C and D, respectively. See Figure 9(a). The tree path from D' down to D is in $\mathcal{F}_{i_0}$, so the cells on the tree path are splittable. These cells are nested, and they contain D. So they all intersect $R_\infty(C)$. Since D' intersects $R_\infty(C) \subset R_1(C')$, C' and D' are identical or neighboring leaf cells in $\tilde{\mathcal{F}}_{i_0-1}$. So $\ell_{D'} \geq \frac{1}{2}\ell_{C'} \geq \ell_C$ as $\tilde{\mathcal{F}}_{i_0-1}$ is balanced. Therefore, there must exist a cell on the tree path from D' to D with side length in the range $[2^{-m}\ell_C, 2\ell_C]$, and

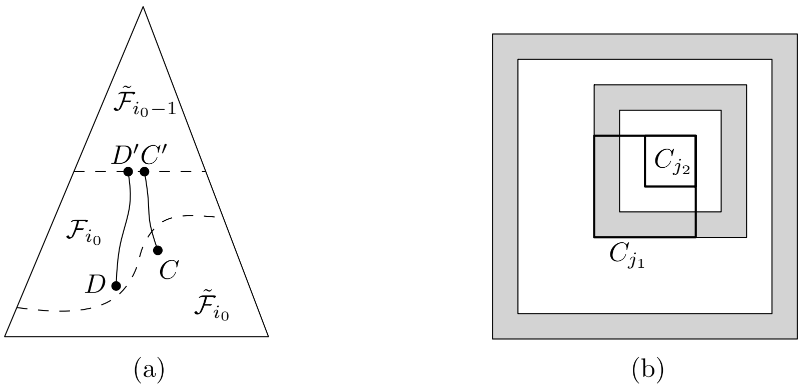


Figure 9: (a) D is in $\mathcal{F}_{i_0}$, C is in $\tilde{\mathcal{F}}_{i_0} \setminus \mathcal{F}_{i_0}$. D' and C' are leaf cells of $\tilde{\mathcal{F}}_{i_0-1}$ that are ancestors of D and C, respectively. (b) $C_{j_2} \subseteq C_{j_1}$. The annuli $R_\infty(C_{j_1}) \setminus R_{m-j_1}(C_{j_1})$ and $R_\infty(C_{j_2}) \setminus R_{m-j_2}(C_{j_2})$ are shaded in gray.

this cell must be splittable as it is an internal node of $\mathcal{F}_{i_0}$. But this is a contradiction to the assumption of the lemma.

We claim that C is a leaf cell in $\tilde{\mathcal{F}}_{i_0}$. We have shown in the previous paragraph that all splittable cells in $\mathcal{F}_{i_0}$ intersecting $R_\infty(C)$ have side lengths at least $2^{-m}\ell_C$. So the assumption of the lemma forces their side lengths to be greater than $2\ell_C$. Since C is non-splittable, Lemma 6.3 is applicable, establishing that C is a leaf cell of $\tilde{\mathcal{F}}_{i_0}$.

As C is non-splittable, it remains a leaf cell until it is split later by the balancing rule. Suppose that C is a leaf cell of $\mathcal{F}_{i_1}$ and C is split in the construction of $\tilde{\mathcal{F}}_{i_1}$ for some $i_1 > i_0$. Since C is non-splittable, Lemma 6.3 implies the following:

$$\text{There exists a splittable cell in } \mathcal{F}_{i_1} \text{ that intersects } R_\infty(C). \tag{6.1}$$

As $i_0 \leq i_1 - 1$, C is also a leaf cell of $\tilde{\mathcal{F}}_{i_1-1}$. Among all the leaf cells of $\tilde{\mathcal{F}}_{i_1-1}$ that intersect $R_\infty(C)$, none has side length $4\ell_C$ or more because such a leaf cell would be a neighbor of C, violating the balance. Therefore, all leaf cells of $\tilde{\mathcal{F}}_{i_1-1}$ that intersect $R_\infty(C)$ have side lengths at most $2\ell_C$. The assumption of the lemma further implies that all splittable leaf cells of $\tilde{\mathcal{F}}_{i_1-1}$ that intersect $R_\infty(C)$ have side lengths less than $2^{-m}\ell_C$. A cell of $\tilde{\mathcal{F}}_{i_1-1}$ that intersects $R_j(C)$ has side length at least $2^{-j}\ell_C$ for all $j \geq 0$ by Lemma 6.2. Therefore, the splittable leaf cells of $\tilde{\mathcal{F}}_{i_1-1}$ that intersect $R_\infty(C)$ do not intersect $R_m(C)$. The octree alignment ensures that:

$$\text{Every splittable cell of } \tilde{\mathcal{F}}_{i_1-1} \text{ that intersects } R_\infty(C) \text{ is contained in } R_\infty(C) \setminus R_m(C). \tag{6.2}$$

Any cell in $\mathcal{F}_{i_1} \setminus \tilde{\mathcal{F}}_{i_1-1}$ is a descendant of some splittable leaf cell of $\tilde{\mathcal{F}}_{i_1-1}$, so the cells in $\mathcal{F}_{i_1} \setminus \tilde{\mathcal{F}}_{i_1-1}$ that intersect $R_\infty(C)$ are also contained in $R_\infty(C) \setminus R_m(C)$. Combining this observation with (6.1) and (6.2) gives the following conclusion:

$$\text{There exists a splittable cell in } \mathcal{F}_{i_1} \text{ that lies in } R_\infty(C) \setminus R_m(C). \tag{6.3}$$

Let C_1 be a child of C. Thus, C_1 is a cell of $\tilde{\mathcal{F}}_{i_1} \setminus \mathcal{F}_{i_1}$. Since $R_\infty(C_1) \subset R_1(C) \subset R_m(C)$, no splittable cell in $\mathcal{F}_{i_1} \setminus \tilde{\mathcal{F}}_{i_1-1}$ intersects $R_\infty(C_1)$. Applying Lemma 6.3 to C_1 shows that C_1 is a leaf cell of $\tilde{\mathcal{F}}_{i_1}$. Now suppose that C_1 is split during the construction of $\tilde{\mathcal{F}}_{i_2}$ for some $i_2 > i_1$. We repeat the above argument to obtain a conclusion analogous to (6.3): some splittable cell in $\mathcal{F}_{i_2}$ lies in $R_\infty(C_1) \setminus R_{m-1}(C_1)$. Inductively, we obtain:

For any descendant C_k of C of side length $2^{-k}\ell_C$ for some $k \leq m-1$, C_k is not splittable, and if it is split by the balancing rule, some splittable cell in $\mathcal{F}_{i_{k+1}}$ lies in $R_\infty(C_k) \setminus R_{m-k}(C_k)$.

We are ready to establish the lemma. For the sake of contradiction, suppose that the subtree of C has at least $m = 3^d$ levels including C. Let $\mathcal{K}$ be the set of $3^d - 1$ disjoint canonical cubes in $R_\infty(C) \setminus C$ with side length ℓ_C. In the subtree of C, we can find two distinct cells C_{j_1} and C_{j_2} such that $0 \leq j_1 < j_2 \leq m-1$, C_{j_1} is an ancestor of C_{j_2}, and the splitting of C_{j_1} and C_{j_2} implies the existence of some splittable cells D_{j_1} in $\mathcal{F}_{i_{j_1}+1}$ and D_{j_2} in $\mathcal{F}_{i_{j_2}+1}$ that lie in the cube(s) in $\mathcal{K}$. Moreover, the pigeonhole principle implies that we can choose j_1 and j_2 such that D_{j_1} and D_{j_2} lie in the same cube in $\mathcal{K}$. By our result in the previous paragraph, $D_{j_1} \subseteq R_\infty(C_{j_1}) \setminus R_{m-j_1}(C_{j_1})$, and $D_{j_2} \subseteq R_\infty(C_{j_2}) \setminus R_{m-j_2}(C_{j_2})$. Refer to Figure 9(b). Since C_{j_2} is contained in C_{j_1}, the distance between the two annuli $R_\infty(C_{j_1}) \setminus R_{m-j_1}(C_{j_1})$ and $R_\infty(C_{j_2}) \setminus R_{m-j_2}(C_{j_2})$ is at least

$$\begin{aligned}
& (3/2 - 2^{-(m-j_1)})\ell_{C_{j_1}} - (2^{-1}\ell_{C_{j_1}} + \ell_{C_{j_2}}) \\
\geq\ & (2^{-1} - 2^{-(m-j_1)})2^{-j_1}\ell_C \quad (\because 2^{-j_1}\ell_C = \ell_{C_{j_1}} \geq 2\ell_{C_{j_2}}.) \\
=\ & (2^{-1-j_1} - 2^{-m})\ell_C \\
\geq\ & 2^{-m}\ell_C. \quad (\because j_1 \leq m-2.)
\end{aligned}$$

Let D be the lowest common ancestor of D_{j_1} and D_{j_2} in $\mathcal{T}_Q$. Since D must cross the gap between the two annuli $R_\infty(C_{j_1}) \setminus R_{m-j_1}(C_{j_1})$ and $R_\infty(C_{j_2}) \setminus R_{m-j_2}(C_{j_2})$, the side length of D is at least $2^{-m}\ell_C$. Since D_{j_1} and D_{j_2} lie in the same cube in $\mathcal{K}$, D is not bigger than a cube in $\mathcal{K}$ by the octree alignment. Therefore, the side length of D lies in the range $[2^{-m}\ell_C, \ell_C]$, and D intersects $R_\infty(C)$. Both D_{j_1} and D_{j_2} contain some sample points as they are splittable. The sample points in $D_{j_1} \cup D_{j_2}$ must be distributed into two or more children of D as D is their lowest common ancestor. It follows that D is splittable. But this is a contradiction because the lemma assumes that no splittable cell in $\mathcal{T}_Q$ that intersects $R_\infty(C)$ has side length in the range $[2^{-m}\ell_C, 2\ell_C]$. $\square$

Proof of Lemma 3.4. Consider $\mathcal{T}_Q$ for some $Q \subseteq P$. Let C be a cell in $\mathcal{T}_Q$ that is not in the initial octree. If the subtree rooted at C has more than 3^d levels, by Lemma 6.4, there is a splittable cell with side length $\Theta(\ell_C)$ at distance $\Theta(\ell_C)$, so we can charge C to it. Each splittable cell is charged $O(1)$ times, which implies a total charge of $O(n_Q + 1)$, where n_Q is the number of the splittable cells in $\mathcal{T}_Q$. The cells of $\mathcal{T}_Q$ not covered by the $O(n_Q + 1)$ charge are at $O(1)$ levels below the root. The total size of all octrees is thus $O(n)$ by Lemma 6.1.

We use a trick in [10] to get the desired running time. First, sort P d times by each coordinate component in $O(n \log n)$ time, and store the d sorted lists at the root of $\mathcal{T}_P$. Testing whether a cell C in $\mathcal{T}_P$ is splittable can be done in $O(1)$ time by maintaining the minimum and maximum of each coordinate component of the points in C. To split C, we divide C into two equal halves A_1 and A_2 by a hyperplane perpendicular to the first axis. We split the list sorted by the first coordinate component in $\min\{|P \cap A_1|, |P \cap A_2|\}$ time by moving from both ends of the list towards the middle. Suppose that $|P \cap A_1| \leq |P \cap A_2|$. Remove the points of $P \cap A_1$ from the d sorted lists, and create d sorted lists for A_1. This takes $O(|P \cap A_1|)$ time. After the removal of $P \cap A_1$, the modified sorted lists for C are exactly the sorted lists for A_2, so we only need to store reference pointers to them at A_2. Divide A_1 into 2^{d-1} cells directly in $O(|P \cap A_1|)$ time. For A_2, we repeat the above to split it by hyperplanes in the other $d-1$ directions. In the end, there is exactly one child C' of C with reference pointers to the final sorted lists left at C. The total time spent is $O(|P \cap (C \setminus C')|)$. We charge each point in $C \setminus C'$ $O(1)$ amount of work. Since every child of C other than C' has at most $|P \cap C|/2$ points, a point can be charged only $O(\log n)$ times. The construction time of $\mathcal{T}_P$ is thus $O(|\mathcal{T}_P| + n \log n)$.

Similar analysis applies to the recursive calls. Our construction of the initial octree guarantees that every leaf of the initial octree for a cluster X in $\mathcal{T}_Q$ contains sample points from one core in X. So we can keep reference pointers at such a leaf to the sorted lists by every coordinate stored at an appropriate leaf cell in $\mathcal{T}_Q$. We have shown that $\sum_Q |\mathcal{T}_Q| = O(n)$. The total time is thus $O\big(n \log n + \sum_Q |\mathcal{T}_Q|\big) = O(n \log n)$. $\square$

7 Conclusion and discussion

We propose a simple and fast surface reconstruction algorithm that runs in $O(n \log n)$ time, which is optimal in the pointer machine model. The only existing $O(n \log n)$-time algorithm is due to Funke and Ramos [24], but no experimental result has been offered. We follow the first two high-level steps in the algorithm of Funke and Ramos, in which the first step is to extract a locally uniform subsample. Several sophisticated data structures were employed in [24] for the subsample extraction step in order to achieve the $O(n \log n)$-time bound. In contrast, our algorithm builds a variant of the standard octree, and then obtains a locally uniform subsample by traversing and trimming the octrees. It is much simpler and more efficient in both theory and practice. We built a prototype of our surface reconstruction algorithm that extracts a locally uniform sample from the input sample, runs Cocone on it, and then add back the remaining sample points via edge flips. Experiments shows that it is faster than running Cocone alone for both non-uniform and uniform inputs. For non-uniform inputs, it is 51% to 68% faster.

After extracting a locally uniform subsample, one can obtain a reconstruction efficiently in ways different from what we described. We believe that a good reconstruction quality can also be obtained by invoking other surface reconstruction algorithms in the literature. Dey, Funke, and Ramos [19] showed that the Cocone algorithm can be modified to run in $O(n \log n)$ time under a stronger notion of locally uniform sampling. It is possible that some other reconstruction algorithms can also be modified to run in $O(n \log n)$ time under locally uniform sampling.

It is an interesting question to study whether our method can also be used for fast reconstruction of surfaces with boundaries. The presence of boundaries means the presence of holes. However, if the current scale being used is too small, the space among sample points may also appear as "holes". Currently, our procedure TRIM makes use of the detection of such "holes" to decide whether the current scale is appropriate. Therefore, in order to detect real holes in the surface, one may need to mark the sample points near the boundaries. These marked sample points can then inform the procedure TRIM that such holes are real. It will require more research work in order to identify sample points near the boundaries.

We believe that our methods can also be applied to extract a locally uniform subsample from a dense point cloud in $\mathbb{R}^d$ for $d \geq 4$, although the details have to be worked out. A key task is the tangent space estimation in our procedure TRIM. For surfaces in $\mathbb{R}^3$, the normal of a triangle that connects three appropriately chosen sample points is used in TRIM. For manifolds in higher dimensions, a different method is needed to estimate tangent spaces. Local PCA is a promising approach, and it has been proved to be provably good under a stronger notion of locally uniform sampling [16].

References

[1] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C.T. Silva. Point set surfaces. In *Proceedings of the IEEE Conference on Visualization*, pages 21–28, 2001.

[2] N. Amenta and M. Bern. Surface reconstruction by Voronoi filtering. *Discrete and Computational Geometry*, 22:481–504, 1999.

[3] N. Amenta, S. Choi, T.K. Dey, and N. Leekha. A simple algorithm for homeomorphic surface reconstruction. *International Journal on Computational Geometry and Applications*, 12:125–141, 2002.

[4] N. Amenta, S. Choi, and N. Kolluri. The power crust, unions of balls, and the medial axis transform. *Computational Geometry: Theory and Applications*, 19:127–153, 2001.

[5] N. Amenta and T.K. Dey. Normal variation with adaptive feature size. A note as an erratum to Lemma 2 in [18].

[6] D. Attali, J.-D. Boissonnat, and A. Lieutier. Complexity of the Delaunay triangulation of points on surfaces: the smooth case. In *Proceedings of the 19th Annual Symposium on Computational Geometry*, pages 201–210, 2003.

[7] J.-D. Boissonnat and F. Cazals. Smooth surface reconstruction via natural neighbor interpolation of distance functions. *Computational Geometry: Theory and Applications*, 22:185–203, 2002.

[8] J.-D. Boissonnat and A. Ghosh. Manifold reconstruction using tangential delaunay complexes. *Discrete and Computational Geometry*, 51:221–267, 2014.

[9] J.-D. Boissonnat, L.J. Guibas, and S.Y. Oudot. Manifold reconstruction in arbitrary dimensions using witness complexes. *Discrete and Computational Geometry*, 42:37–70, 2009.

[10] P.B. Callahan and S.R. Kosaraju. A decomposition of multidimensional point sets with applications to k-nearest-neighbors and n-body potential fields. *Journal of ACM*, 42:67–90, 1995.

[11] T.M. Chan, J. Snoeyink, and C.-K. Yap. Primal dividing and dual pruning: output-sensitive construction of four-dimensional polytopes and three-dimensional Voronoi diagrams. *Discrete and Computational Geometry*, 18:433–454, 1997.

[12] B. Chazelle. An optimal convex hull algorithm in any fixed dimension. *Discrete and Computational Geometry*, 10:377–409, 1993.

[13] S.-W. Cheng and M-K. Chiu. Implicit manifold reconstruction. In *Proceedings of the 25th ACM-SIAM Symposium on Discrete Algorithms*, pages 161–173, 2014.

[14] S.-W. Cheng, T.K. Dey, and E.A. Ramos. Manifold reconstruction from point samples. In *Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1018–1027, 2005.

[15] S.-W. Cheng and J. Jin. Edge flips in surface meshes. *Discrete and Computational Geometry*, 54:110–151, 2015.

[16] S.-W. Cheng, Y. Wang, and Z. Wu. Provable dimension detection using principal component analysis. *International Journal of Computational Geometry and Applications*, 18:414–440, 2008.

[17] B. Curless and M. Levoy. A volumetric method for building complex models from range images. In *SIGGRAPH*, pages 306–312, 1996.

[18] T.K. Dey. *Curve and surface reconstruction: Algorithms with mathematical analysis.* Cambridge University Press, New York, 2006.

[19] T.K. Dey, S. Funke, and E.A. Ramos. Surface reconstruction in almost linear time under locally uniform sampling. In *European Workshop on Computational Geometry.* Berlin, 2001.

[20] D. Dumitriu, S. Funke, M. Kutz, and N. MilosavljevIć. On the locality of extracting a 2-manifold in $\mathbb{R}^3$. In *Proceedings of the 11th Scandinavian Workshop on Algorithm Theory*, pages 270–281, 2008.

[21] D. Dumitriu, S. Funke, M. Kutz, and N. MilosavljevIć. How much geometry it takes to reconstruct a 2-manifold in $\mathbb{R}^3$. *ACM Journal of Experimental Algorithmics*, 14:2.2:1–2.2:17, 2009.

[22] J. Erickson. Nice point sets can have nasty Delaunay triangulations. *Discrete Computational Geometry*, 30:109–132, 2003.

[23] H. Federer. Curvature measures. *Transactions of the American Mathematical Society*, 93:418–491, 1959.

[24] S. Funke and E.A. Ramos. Smooth-surface reconstruction in near-linear time. In *Proceedings of the 13th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 781–790, 2002.

[25] J. Giesen and U. Wagner. Shape dimension and intrinsic metric from samples of manifolds. *Discrete and Computational Geometry*, 32:245–267, 2004.

[26] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Surface reconstruction from unorganized points. In *SIGGRAPH*, pages 71–78, 1992.