Undersampling and Oversampling in Sample Based Shape Modeling

Tamal K. Dey Joachim Giesen Samrat Goswami James Hudson Rephael Wenger Wulue Zhao

Ohio State University Columbus, OH 43210

Abstract

Shape modeling is an integral part of many visualization problems. Recent advances in scanning technology and a number of surface reconstruction algorithms have opened up a new paradigm for modeling shapes from samples. Many of the problems currently faced in this modeling paradigm can be traced back to two anomalies in sampling, namely *undersampling* and *oversampling*. Boundaries, non-smoothness and small features create undersampling problems, whereas oversampling leads to too many triangles. We use Voronoi cell geometry as a unified guide to *detect* undersampling and oversampling. We apply these detections in surface reconstruction and model simplification. Guarantees of the algorithms can be proved. In this paper we show the success of the algorithms empirically on a number of interesting data sets.

Keywords: Computational Geometry, Surface Reconstruction, Geometric Modeling, Mesh Generation, Polygonal Mesh Reduction, Polygonal Modeling, Shape Recognition.

1 Introduction

Visualizations of shapes with their models are integral part of many scientific computations. *Surface reconstruction* which builds a piecewise linear approximation of a surface from its samples provides a powerful paradigm for modeling shapes. We call this paradigm *Sample Based Shape Modeling*, or SBSM in short. For many applications SBSM can provide an initial mesh for the model which can be processed further according to the need. For example, the traditional spline based surface modeling can benefit from SBSM in generating an initial control mesh and then smoothing it out with techniques such as surface fairing [29] or subdivision methods [33].

1.1 Background

A vast literature has built up on the problem of surface reconstruction in recent years. Various algorithms with different capabilities and guarantees have been proposed. A very early paper on the problem was by Boissonat [11] who proposed a 'sculpting' of the Delaunay triangulation for reconstruction. A more refined sculpting strategy was designed by Edelsbrunner and Mücke [16] in their α-shape algorithm. Bajaj, Bernardini and Xu [9] used α-shapes for reconstructing scalar fields and 3D CAD models. In [15] Edelsbrunner reported the design of a commercial software WRAP that eliminated the need for uniform samples in α-shapes. Hoppe et al. [24] reconstructed the surface using the zero level set of a distance function defined over the samples. Curless and Levoy [14] used a distance function to construct an implicit surface from multiple range scans. Turk and Levoy [31] devised an incremental algorithm that iteratively improves a reconstruction by erosion and zippering. A cluster based strategy was used by Heckel et al. for reconstructions in [22]. Gopi, Krishnan and Silva [21] projected sample points with their neighbors on a plane and lifted the local 2D Delaunay triangulations to reconstruct the surface. Bernardini et al. [10] proposed a ball pivoting algorithm that reconstructs the surface incrementally by rolling a ball over the sample points. Kobbelt and Botsch [25] used hardware projections to reconstruct surfaces from large data. Attali [8] introduced normalized meshes to reconstruct surfaces. Very recently Amenta, Bern and Kamvysselis [2] proposed a Voronoi based surface reconstruction called CRUST and proved its theoretical guarantees. This algorithm was later improved by the COCONE algorithm in [3] and the POWER CRUST in [4]. Boissonnat and Cazals [12] showed how *natural neighbors* can aid surface reconstructions. See the surveys by Mencl and Müller [26] for other surface reconstruction algorithms and by Uselton [32] for contour based surface reconstructions.

1.2 Undersampling and oversampling

Although a considerable success has been made by recent surface reconstruction algorithms, some of the major problems in SBSM that still remain can be traced back to two anomalies in sampling, namely *undersampling* and *oversampling*. In this paper we concentrate on detecting these two anomalies using Voronoi structures and show how they can aid surface reconstructions.

Undersampling happens when a surface has small features such as high curvatures that are sampled inadequately. It cannot be avoided when a surface is not smooth. In this case no finite sampling is dense enough for *sharp edges* or *corners*. Even the presence of boundaries in the surface can be

thought of as a consequence of undersampling. The sampling in this case is intentionally stopped to introduce the boundaries. In all these cases, the reconstructed surface often has undesirable holes and triangles and may even not be a manifold with boundary. Detection of undersampling can help mend these surfaces either by resampling manually, or by stitching the holes algorithmically. Other than this, it has direct application in reconstructing surfaces with boundaries and determining features such as discontinuities in scalar fields [20]. We also show that nonsmooth surface reconstruction can also benefit from this detection.

Not only undersampling poses challenge to SBSM, but also its counterpart, *oversampling*, causes difficulties, particularly in post-processing. A surface, sometimes, is sampled with unnecessarily high density. Surfaces reconstructed from an unnecessarily dense sample contain large numbers of triangles and thus become unwieldy for further processing such as graphic rendering or finite element analysis. A variety of algorithms have been proposed to *simplify* a piecewise linear surface [13, 17, 19, 23, 28, 30]. Most of these algorithms choose a subset of vertices, edges, or triangles for deletion so that the overall shape of the space delimited by the surface is maintained. The major concerns in such *simplification* strategies are to preserve topological and geometric features of the surface and avoid self intersections. Oversampling detection can directly benefit the simplification process by throwing away unnecessary sample points before reconstruction. Such sample decimations have been proposed earlier by [2] which we refine with new concepts and a guarantee in surface reconstruction. This alternative approach of sample decimation in place of model simplification has several advantages. First of all, one does not need to worry about preserving features or avoiding self intersections. If the decimation guarantees a sample density that is still appropriate for surface reconstruction, the reconstructed model retains all features without any self intersection. Furthermore, sample decimation at different levels allows an alternative approach to multiresolution meshing [18, 27].

1.3 Our approach

We use the structure of Voronoi diagrams as the key ingredient for *undersampling* and *oversampling* detection. A result in [2] says that the Voronoi cells for a dense sample of a surface are long and thin along the direction of normals at the sample points. We use this structural information to detect samples in the vicinity of undersampled regions, which we call *boundary samples*. The information about boundary samples can be exploited in many surface reconstruction algorithms simply by disallowing triangles that are incident to the boundary samples.

The structure of the Voronoi cells also indicates oversampled regions. Voronoi cells in oversampled regions tend to be overly long and 'skinny'. Again, we measure the 'skinniness' of the Voronoi cells and delete sample points accordingly. The deletion stops when all Voronoi cells become sufficiently 'fat'. This strategy decimates the samples up to a level determined by an input parameter.

2 Voronoi cell geometry

The main tool we use to detect undersampling and oversampling is the geometry of the Voronoi cell structure. Our reasoning is based on a density condition called ε-sampling as introduced in [2]. This definition builds on the *medial axis* of a surface and the related term of *local feature size*. The medial axis of a surface $S \subset \mathbb{R}^3$ is defined as the locus of all points that have more than one closest point on S. The local feature size at a point $x \in S$, denoted as $f(x)$, is the least distance of x to the medial axis. Let $P \subset \mathbb{R}^3$ be a set of sample points on a surface S. We say P is an ε-sample of S if each point $x \in S$ has a sample within $\varepsilon f(x)$ distance. It has been observed that typically $\varepsilon < 0.4$ gives a dense enough sampling for surface reconstruction. For a dense sample each Voronoi cell is long and thin as shown in Figure 1. We introduce a measure of this structural property that is easily computable. We need the following definitions for further expositions.

Definitions

Let V_P and D_P denote the Voronoi diagram and Delaunay triangulation of the sample $P \subset \mathbb{R}^3$ respectively. For a sample $p \in P$, let V_p be its Voronoi cell. For convenience we use the notation $\angle(\mathbf{v}, \mathbf{w})$ to denote the *acute* angle between the lines supporting two vectors $\mathbf{v}$ and $\mathbf{w}$.

Poles: The farthest Voronoi vertex p^+ in V_p is called the positive *pole* of p. The negative pole of p is the farthest point $p^- \in V_p$ from p such that two vectors from p to p^+ and p^- make an angle more than $\frac{\pi}{2}$. We call $\mathbf{v_p} = p^+ - p$, the *pole vector* for p. If V_p is unbounded, p^+ is taken at infinity, and the direction of $\mathbf{v_p}$ is taken as the average of all directions given by unbounded Voronoi edges.

Cocone: The set $C_p(\theta, \mathbf{v}) = \{y \in V_p \ : \ \angle((y-p), \mathbf{v}) \geq \frac{\pi}{2} - \theta\}$ is called the cocone of p with axis $\mathbf{v}$ and angle θ. In words, $C_p(\theta, \mathbf{v})$ is the complement of a double cone (clipped within V_p) centered at p with an opening angle $\frac{\pi}{2} - \theta$ around the axis aligned with $\mathbf{v}$. Usually, we take $\mathbf{v} = \mathbf{v_p}$, the *pole* vector and $\theta < \frac{\pi}{8}$ small. See Figure 1 for an example of a cocone.

Cocone neighbor: A sample q is called a *cocone neighbor* of another sample p if V_q overlaps with the cocone of p. Thus, the set of cocone neighbors denoted N_p is given by $N_p = \{q \in P | V_q \cap C_p(\theta, \mathbf{v_p}) \neq \emptyset\}$.

Let d_p denote the maximum distance of any point from p that has p as its nearest sample on S. A point p lies in an oversampled region, if d_p is small compared to the local feature size $f(p)$. On the other hand, if this distance is too large compared to $f(p)$, the point p is in an undersampled region. However, we cannot perform these comparisons exactly since S is unknown. Instead, we use the approximations to d_p and $f(p)$.

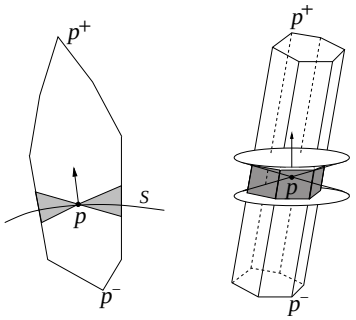

Figure 1: A Voronoi cell together with the normalized pole vector and the cocone (shaded) in two dimensions (left) and three dimensions (right).

Recent results in [4, 12] establish that poles approximate the medial axis in an asymptotic sense. Therefore, the *height* as defined below approximates $f(p)$.

Height: The *height* h_p is defined as the length $||p^- - p||$.

In order to approximate d_p we use the farthest point from p in $C_p(\theta, \mathbf{v}_p)$. To understand the rationale for this approximation, observe that $V_{p,S}$ lies close to the tangent plane at p if S is assumed to be locally flat, and the pole vector $\mathbf{v_p} = p^+ - p$ approximates the normal direction at p. Therefore, the cocone $C_p(\theta, \mathbf{v_p})$ coincides with a thin neighborhood around $V_{p,S}$ for a small θ. Thus, the *radius* of $C_p(\theta, \mathbf{v_p})$ as defined below approximates d_p.

Radius: The *radius* r_p is defined as the distance $\max\{||y - p|| : y \in C_p(\theta, \mathbf{v_p})\}$.

3 Undersampling

When a surface is sampled, parts of it may be sampled densely leaving patches that are undersampled. The *boundaries* between these well and under sampled patches demark the undersampled regions. We need to detect the samples that represent these boundaries. Formally, let $F \subset \mathbb{R}^3$ be a smooth compact surface without boundary. We focus on a maximal subset $S \subseteq F$ where each point $x \in S$ has a sample point $p \in P$ within $\varepsilon f(x)$ distance. Our goal is to detect the boundaries of S from the sample P. Notice that this formulation allows the boundary of S to represent all kinds of undersampling including the ones caused by nonsmoothness and true boundaries.

3.1 Boundaries

For any compact surface S we can distinguish interior points from the boundary points. An interior point has a neighborhood homeomorphic to the plane $\mathbb{R}^2$. A boundary point, on the other hand, has a neighborhood homeomorphic to the halfplane $\mathbb{H}^2_+ = \{(x, y) \in \mathbb{R}^2 : x \geq 0\}$.

In SBSM only a finite sample P, not the surface S, is given. Even though all samples in P may be interior points of S, the existence of a non-empty boundary has to be determined only from the sample points. We need a classification of interior and boundary *samples* that captures the intuitive difference between interior and boundary *points*. Recall that $V_{p,S} = V_p \cap S$ is the subset of S lying in V_p. The points in $V_{p,S}$ have p as nearest sample. In other words, the sample p is a discrete representative of the continuous patch $V_{p,S}$.

Definition 1 (Interior and boundary samples) *A sample $p \in P$ is called* interior *if $V_{p,S}$ does not contain a boundary point of S. Sample points that are not interior are called* boundary. *See Figure 2.*

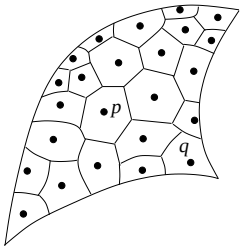

Figure 2: Intersection of V_P with the surface S, p is an interior sample, q is a boundary sample.

The above definition cannot be exploited algorithmically to detect boundary samples since S is unknown. Therefore, we aim for a different characterization that distinguishes interior samples from boundary ones.

Definition 2 (Flatness) *Given two parameters ρ and α, a sample point $p \in P$ is called flat if the following two conditions hold:*

(1) Ratio condition: $\frac{r_p}{h_p} \leq \rho$;

(2) Normal condition: $\forall q$ *with* $p \in N_q$, $\angle(\mathbf{v_p}, \mathbf{v_q}) \leq \alpha$.

The *ratio condition* captures that V_p is long and thin. Unfortunately, this single condition is not enough to differentiate the interior samples from the boundary ones since V_p of a boundary sample p may also be long and thin; see Figure 3 for an example in two dimensions. Hence we add the *normal condition* to define flat samples.

Obviously, flatness depends on the parameters ρ and α. For the proof that boundary samples cannot be flat, we use $\rho = 1.3\varepsilon$ and $\alpha = 0.14$ radians [5] for an ε-sampled surface. However, in practice much larger values work as our experiments show.

3.2 Theoretical guarantees

The algorithm for boundary sample detection is based on the two main theorems; one says that boundary samples cannot be flat, and the other says that a subset of interior samples called *deep interior samples* must be flat. An interior sample is *deep* if it does not have any boundary sample as its cocone neighbor. We formally prove these two theorems in [5].

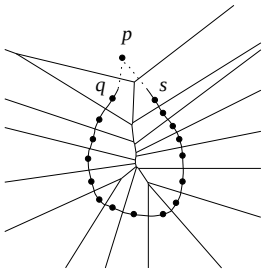

Figure 3: The large gaps between p and q and between p and s render them as boundary samples according to our definition (two dimensional case). Boundary samples q and s have long and thin Voronoi cells.

Theorem 1 *Boundary samples are not flat.*

Theorem 2 *Deep interior vertices are necessarily flat.*

Theorem 1 assumes some conditions about boundary samples that are stated formally in [5]. For most practical data these mild assumptions are quite valid as exhibited by our experimental success.

3.3 Boundary detection

The algorithm for boundary detection first computes the set of interior vertices, R, that are flat. It takes two parameters ρ and α to check the ratio and normal conditions respectively. In theory, we require $\rho = 1.3\varepsilon$ and $\alpha = 0.14$ for an ε-sampled surface. However, in our experiments we obtain good results for values as large as $\rho = 0.99$ and $\alpha = \frac{\pi}{6}$.

ISFLAT ($p \in P$, α, ρ)
1 compute the radius r_p and the height h_p.
2 **if** $\frac{r_p}{h_p} \leq \rho$
3 **if** $\forall q$ with $p \in N_q$, $\angle(\mathbf{v_p}, \mathbf{v_q}) \leq \alpha$
4 **return** `true`
5 **endif**
6 **endif**
7 **return** `false`

Theorem 1 guarantees that R cannot include any boundary sample. Assuming that P has a deep interior sample for each component of S, R cannot be empty by Theorem 2. Thus, ISFLAT can be thought of as a procedure to capture really good interior samples. However, not all interior samples are flat, mainly because the normal condition is violated. So we start walking from R, expanding it iteratively, each time including a new interior sample. A generic iteration proceeds as follows. Let p be any cocone neighbor of a sample $q \in R$ so that $p \notin R$ and p satisfies the ratio condition. If $\mathbf{v_p}$ and $\mathbf{v_q}$ make a small angle up to orientation, i.e., if $\angle(\mathbf{v_p}, \mathbf{v_q}) \leq \alpha$, we include p in R. If no such sample can be found, the iteration stops at which point all interior samples are included in R. The rest of the samples, i.e. $P \setminus R$ must be boundary ones. See Figure 4.

BOUNDARY (P, α, ρ)
1 $R := \emptyset$
2 **for** all $p \in P$
3 **if** ISFLAT(p, α, ρ) $R := R \cup p$
4 **endfor**
5 **while** $\exists p \notin R$ and $\exists q \in R$ with $p \in N_q$,
and $r_p \leq \rho h_p$ and $\angle(\mathbf{v_p}, \mathbf{v_q}) \leq \alpha$
6 $R := R \cup p$
7 **endwhile**
8 **return** $P \setminus R$

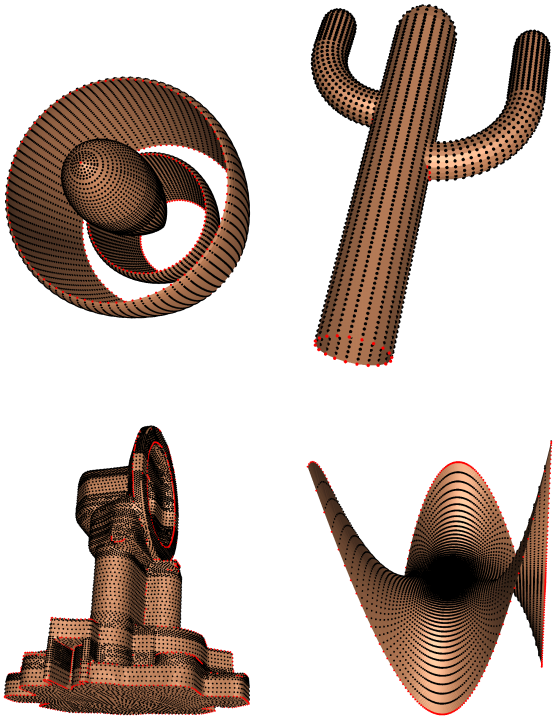

Figure 4: Boundary samples (colored red) detected by BOUNDARY on the data sets ENGINE, CACTUS, OILPUMP and MONKEYSADDLE. This picture needs to be seen in color.

3.4 Reconstruction

As we pointed out earlier almost all surface reconstruction algorithms can benefit from detecting undersampling, especially if the undersampling is due to true boundaries. For our experiments we decided to use the COCONE algorithm of [3]. The COCONE algorithm uses single Voronoi diagram computation as opposed to two in CRUST.

The COCONE algorithm proceeds as follows. Each sample chooses a set of triangles from the Delaunay triangulation of the sample P whose dual Voronoi edges are intersected by the cocones defined at the sample. The set of all such chosen

triangles over all samples is called the *candidate set*. Each of these triangles is shown to lie close to the original surface S and has normal oriented nearly in the same direction as those at its three sample vertices [3]. A subsequent manifold extraction step extracts a manifold surface out of this set of candidate triangles. This manifold is homeomorphic and geometrically close to S [3].

This algorithm works nicely with well sampled smooth surfaces that have no boundary. However, it is our experience that the algorithm computes many undesirable triangles near non-smooth regions and boundaries. The appearance of these undesirable triangles can be attributed to the fact that the boundary samples do not have reliable normals and as a result they choose 'garbage' triangles in the candidate set. At this point we make use of the algorithm BOUNDARY by just disallowing boundary samples to choose triangles. In most cases neighboring interior samples choose correct triangles for boundary samples. However, small holes may be left in the surface where boundary samples come close to each other, a case that happens near non smooth and high curvature regions. We stitch these holes in a later step. See Figure 5.

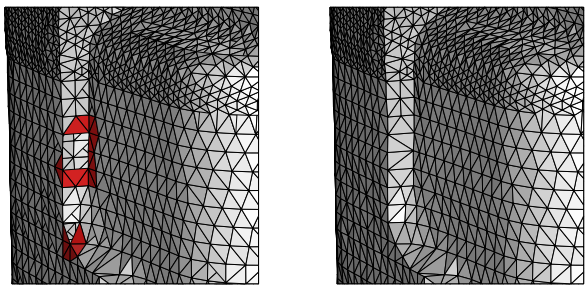

Figure 5: A zoom on the OILPUMP: sharp edges are mostly reconstructed with *candidate triangles* (left), final reconstruction (right). Dark shaded triangles border holes.

COCONE (P, α, ρ, θ)
1 compute the Voronoi diagram V_P of P.
2 B :=BOUNDARY(P, α, ρ)
3 $T := \emptyset$
4 **for** all $p \in P \setminus B$
5 $T := T \cup$ CANDIDATETRIANGLES(p)
6 **endfor**
7 extract a manifold M from T.
8 stitch small holes in M.
9 **return** M

3.4.1 Manifold extraction with boundaries

The algorithm CANDIDATETRIANGLES(p) returns the set of triangles dual to the Voronoi edges intersecting the cocone $C_p(\theta, \mathbf{v_p})$. The manifold extraction step first *prunes* the triangles with bare edges iteratively. Such prunings are necessary for a subsequent *walk* over the remaining candidate triangles as described in [2]. This poses an inherent difficulty for surfaces with boundary. Reconstruction of such a surface must allow triangles with bare edges at the true boundary. In this case the manifold extraction step deletes the entire surface in a cascaded manner. Similar problem arises for otherwise undersampled surfaces. However, if we disallow the removal of triangles incident to boundary samples, this problem does not arise. Thus, our boundary detection algorithm also makes the manifold extraction feasible for undersampled surfaces, especially surfaces with boundaries.

3.4.2 Stitching

After we extract M from candidate triangles, holes may remain in M. These holes may be due to a true boundary, or may reflect undersampling near non-smooth or high curvature regions. First, we separate out the true boundary by considering the size of the triangles which have three vertices on a hole. If these triangles are too big compared to the triangles incident on the edges of the hole in M, we designate that hole as a true boundary. Several measures are possible to determine the size of a triangle. We consider the circumradius of the triangles for this purpose.

The holes that are not designated as true boundary are stitched. These holes are usually small. We observe from our experiments that a set of Delaunay triangles exist that can fill up these small holes. This observation is validated by the results in [1]. In that paper it is shown that an enumeration of all possible combinations of triangles with the linear programming technique does find a set of Delaunay triangles that can fill up small holes. We avoid the computation with linear programming. Instead, for a small hole we include all triangles that have three vertices in that hole. A subsequent prune and walk step as in manifold extraction recovers only the triangles needed to patch the hole from this set. See Figure 6.

3.5 Experiments

Computation of cocones at each sample point needs the cocone angle θ. In [3], it is proved that $\theta = \frac{\pi}{8}$ is a feasible choice. In practice, we also observed that $\theta = \frac{\pi}{8}$ gives good results. All our experiments are conducted with this value of θ. The other two parameters ρ and α for boundary detection should have values $\rho \leq 1.3\varepsilon$ and $\alpha \leq 0.14$ radians in theory [5]. We observed that values as large as $\rho = 0.99$ and $\alpha = \frac{\pi}{6}$ work well in practice. All of our experiments are carried out with these fixed values.

We implemented COCONE in C++ using the computational geometry algorithms library CGAL [35]. The software is available from [34]. Robust Delaunay triangulation/Voronoi diagram computations in the presence of degeneracies and finite precision arithmetic are absolutely necessary for valid output. To this end we used the filtered floating point arithmetic of CGAL. This simulates exact arithmetic

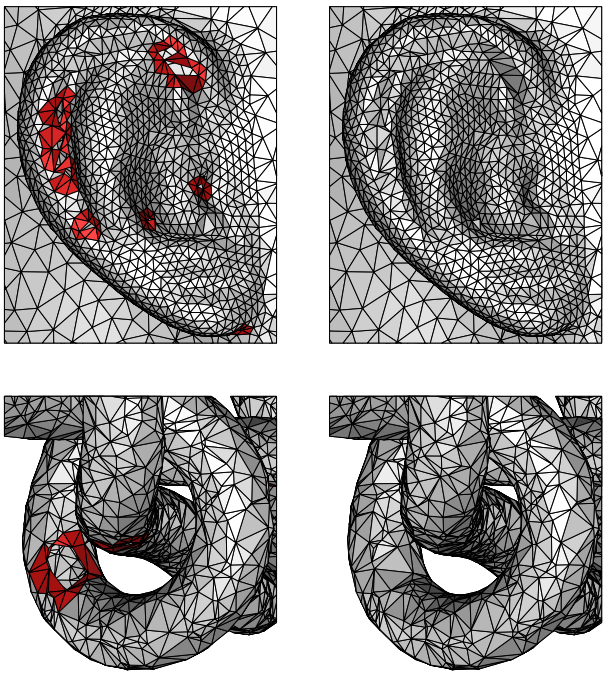

Figure 6: A zoom on the ear of the MANNEQUIN and a zoom on the KNOT: before (left) and after stitching (right).

only on a demand basis. Thus, it provides the advantage of exact arithmetic with a small increase in running time by a factor of two. The running times we report here are due to experiments performed on a PC with 933 Mhz Pentium III processor and 512 MB main memory.

Our modified COCONE algorithm has produced a 'perfect' manifold surface for all data mentioned in Table 1. See Figures 7 and 8. For ENGINE, CACTUS, FOOT, MANNEQUIN and MONKEYSADDLE it has produced a manifold with no hole except at the true boundaries. All these data have undersampling due to either high curvatures or nonsmoothness. For KNOT, 3HOLES and OILPUMP it has produced watertight models though they have undersampling due to missing samples in KNOT and 3HOLES and nonsmoothness in OILPUMP. We verified this claim with the Euler equation relating the number of vertices, edges, triangles and true boundaries (if any).

4 Oversampling

We have already mentioned that the Voronoi structure also provides information about oversampling. The samples responsible for oversampling have overly skinny Voronoi cells. Again, we measure the skinniness by computing the ratio $\frac{r_p}{h_p}$ for each sample p. If this ratio is smaller than a threshold ρ, we delete p. This is the main scheme in our simple decimation algorithm.

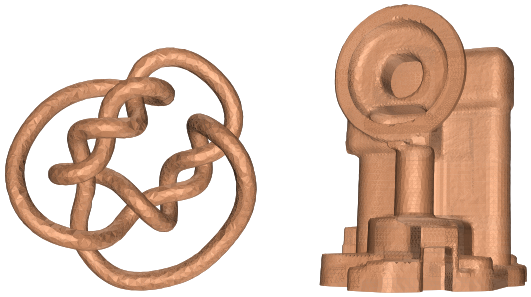

Figure 7: Reconstruction of the data sets KNOT and OILPUMP.

object	number of points	number of triangles	running time(sec.)
CACTUS	3280	6538	50
3HOLES	4000	8008	38
KNOT	10000	20000	96
MONKEYSADDLE	10000	19599	367
ENGINE	11360	22356	229
MANNEQUIN	12772	25427	91
FOOT	20021	40002	149
OILPUMP	30931	61858	235

Table 1: Experimental data.

4.1 Decimation

Recall that h_p estimates the distance of p from the medial axis. As we delete samples, the Voronoi diagram V_P changes. We do not recompute h_p. The reason is that poles approximate the medial axis better with a denser sample than with a sparser one. This also means that the cocone axis $\mathbf{v}$ remains the same for a point p throughout the decimation. However, $C_p(\theta, \mathbf{v})$ and hence its radius r_p change as Voronoi diagram changes. Our simple decimation algorithm keeps track of these changes. We use $\theta = \frac{\pi}{8}$ as before.

DECIMATE(P, ρ)
1 **for** each $p \in P$ compute h_p and r_p
2 $Q := P$
3 **while** $\exists p \in Q$ so that $\frac{r_p}{h_p} < \rho$
4 delete p from Q.
5 update V_Q.
6 **if** V_q is changed for any $q \in Q$ update r_q
7 **endwhile**
8 **return** Q

In [7] it is proved that, for an appropriate ρ, DECIMATE produces a subset $Q \subset P$ so that Q is a dense sample for reconstruction. Additionally, for each remaining sample $p \in Q$ there is a point, say $x \in S$, so that x has p as its nearest sample and the distance $||x-p||$ is more than a constant factor of $\rho f(x)$. It means that p is necessary for x to maintain a sample

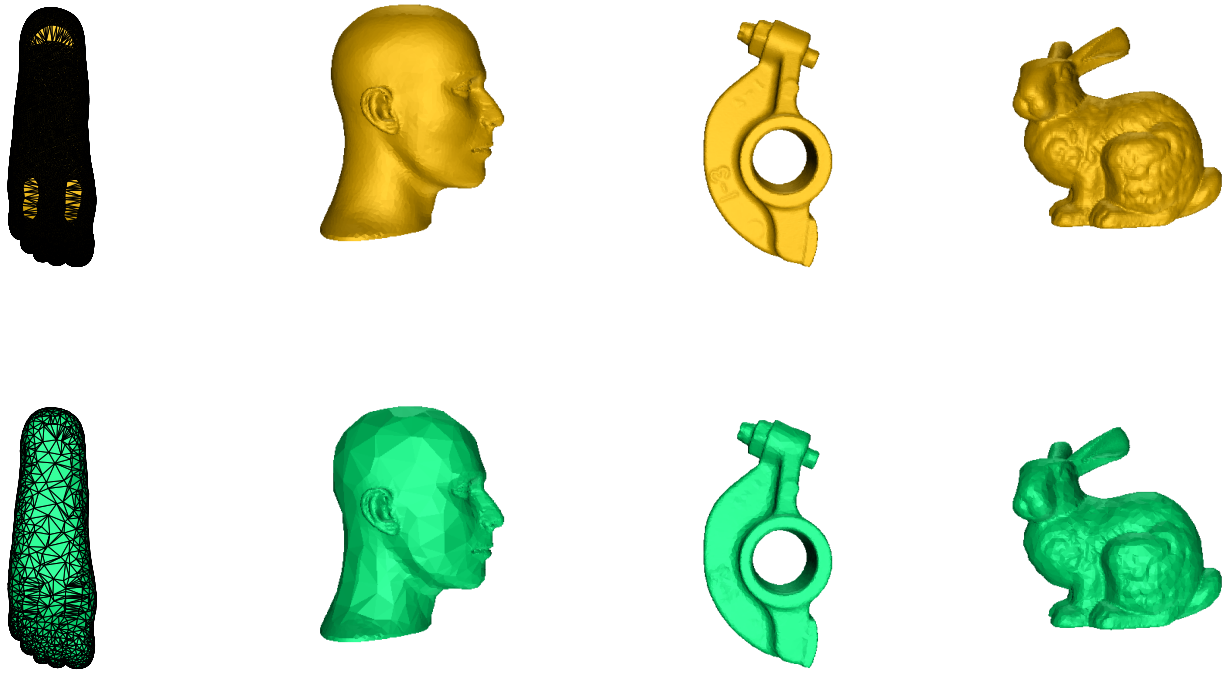

Figure 8: Data sets FOOT, MANNEQUIN, ROCKER and BUNNY. The upper row shows reconstructions without any sample decimation. The lower row shows reconstruction after DECIMATE with $\rho = 0.3$.

density determined by the user parameter ρ. Although in theory we need an extremely small value of ρ, we obtain good results for ρ in the range of $0.25 - 0.4$.

4.2 Experiments

We implemented DECIMATE in C++ using the CGAL library [35]. Figure 8 shows the results on the four data sets FOOT, MANNEQUIN, ROCKER and BUNNY. We achieve almost 65% reduction in data points when $\rho = 0.3$ is used for these experiments; see Table 2. Increasing ρ to 0.4 achieves almost 80% reduction as shown in the table. Different levels of decimation with three different values of ρ are shown in Figure 9.

Data	#points before	#points after $\rho = 0.3$ $\rho = 0.4$	Running time (sec.)
MANNEQUIN	12772	5616 4408	97 108
FOOT	20021	3359 2247	277 268
BUNNY	35947	11171 7747	421 460
ROCKER	40177	12685 8025	1391 1567

Table 2: Results of DECIMATE with $\rho = 0.3$ and $\rho = 0.4$.

5 Conclusions

A major complaint against Delaunay based surface reconstruction algorithms is that they are slow. COCONE can handle data size in the range of 100K sample points with current computing resources. In order to make COCONE useful for large data sets, we investigated if a divide-and-conquer strategy can help in this regard. A remarkable success has been achieved recently with this approach. We partition the data using an octree and then apply the COCONE algorithm described in this paper on each cluster. The surface patches from different cells are stitched automatically by manifold extraction. This result shows that COCONE can be extended to handle data even with more than a million points in less than an hour using modest computing resource such as PCs. The details appear in a companion paper [6].

Noise is the other challenge that needs further investigation. The noise created in the form of outliers is easy to detect and COCONE can easily isolate them. However, local noise introduced by small perturbations in the coordinates of the sample points causes difficulty with the current state of the art. Can we take advantage of the Voronoi cell geometry for

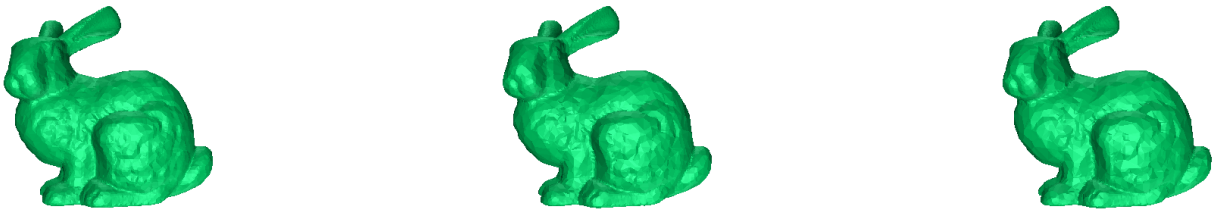

Figure 9: The data set BUNNY (original 35947 points). Decimations with $\rho = 0.25$, 13792 points (left), $\rho = 0.35$, 9251 points (middle) and $\rho = 0.4$, 7747 points (right).

this problem? We plan to investigate this line of research.

Acknowledgement: This research was partially supported by NSF under grant CCR-9988216.

References

[1] U. Adamy, J. Giesen, M. John. New Techniques for Topologically Correct Surface Reconstruction. *Proc. Visualization 2000*, (2000), 373–380.

[2] N. Amenta, M. Bern and M. Kamvysselis. A new Voronoi-based surface reconstruction algorithm. *Proc. SIGGRAPH 98*, (1998), 415-421.

[3] N. Amenta, S. Choi, T. K. Dey and N. Leekha. A simple algorithm for homeomorphic surface reconstruction. *Proc. 16th. Ann. Sympos. Comput. Geom.*, (2000), 213–222.

[4] N. Amenta, S. Choi and R. K. Kolluri. The power crust, unions of balls, and the medial axis transform. *Manuscript*, 2000.

[5] T. K. Dey and J. Giesen. Detecting undersampling in surface reconstruction. *Proc. 17th Ann. Sympos. Comput. Geom.*, (2001), 257–263.

[6] T. K. Dey, J. Giesen and J. Hudson. A Delaunay based shape reconstruction from large data. *Proc. IEEE Sympos. in Parallel and Large Data Visualization and Graphics*, (2001), to appear.

[7] T. K. Dey, J. Giesen and J. Hudson. Decimating samples for mesh simplification. *Proc. 13th Canadian Conf. Comput. Geom.*, (2001), to appear.

[8] D. Attali. *r*-regular shape reconstruction from unorganized points. *Proc. 13th Ann. Sympos. Comput. Geom.*, (1997), 248–253.

[9] C. Bajaj, F. Bernardini and G. Xu. Automatic reconstruction of surfaces and scalar fields from 3D scans. *SIGGRAPH 95*, (1995), 109–118.

[10] F. Bernardini, J. Mittelman, H. Rushmeir, C. Silva and G. Taubin. The ball-pivoting algorithm for surface reconstruction. *IEEE Trans. Vis. Comput. Graphics*, **5**, no. 4, 349–359.

[11] J. D. Boissonnat. Geometric structures for three dimensional shape representation, ACM Transact. on Graphics 3(4), (1984) 266–286.

[12] J. D. Boissonat and F. Cazals. Natural coordinates of points on a surface. *Intl. J. Comput. Geom. Appl.*, to appear.

[13] J. Cohen, A. Varshney, D. Manocha, G. Turk, H. Weber, P. Agarwal, F. Brooks and W. Wright. Simplification envelopes. *SIGGRAPH 96*, (1996), 119–128.

[14] B. Curless and M. Levoy. A volumetric method for building complex models from range images. *SIGGRAPH 96*, (1996), 303-312.

[15] H. Edelsbrunner. Shape reconstruction with Delaunay complex. *LNCS 1380, LATIN'98: Theoretical Informatics*, (1998), 119–132.

[16] H. Edelsbrunner and E. P. Mücke. Three-dimensional alpha shapes. *ACM Trans. Graphics*, **13**, (1994), 43–72.

[17] J. El-Sana and A. Varshney. Controlled simplification of genus for polygonal models. *Proc. Visualization 97*, (1997), 403– 412.

[18] L. de Floriani, P. Magillo and E. Puppo. Multiresolution representation and reconstruction of triangulated surfaces. *In* Advances in Visual Form Analysis, C. Arcelli, L. Cordella, G. Sanniti di Baja (Editors), World Scientific, Singapore, (1997), 140–149.

[19] M. Garland and P. S. Heckbert. Surface simplification using quadric error metrics. *Proc. SIGGRAPH 97*, (1997), 209–216.

[20] T. Gutzmer and A. Iske. Detection of discontinuities in scattered data approximation. *Numerical Algorithms*, **16**, (1997), 155–170.

[21] M. Gopi, S. Krishnan and C. Silva. Surface reconstruction based on lower dimensional localized Delaunay triangulation. *Eurographics 2000*.

[22] B. Heckel, A. E. Uva, B. Hamann and K. I. Joy. Surface reconstruction using adaptive clustering methods. *In* G. Brunnett, H. Bieri and G. Farin, eds., Geometric Modeling, Supplement to the Journal Computing, to appear.

[23] H. Hoppe. Progressive meshes. *Proc. SIGGRAPH 96*, (1996), 99-108.

[24] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald and W. Stuetzle. Surface reconstruction from unorganized points. *Proc. SIGGRAPH 92*, (1992), 71-78.

[25] L. Kobbelt and M. Botsch. An interactive approach to point cloud triangulation. *Proc. Eurographics*, 2000.

[26] R. Mencl and H. Müller. Interpolation and approximation of surfaces from three-dimensional scattered data points. *State of the Art Reports, Eurographics 98*, (1998), 51–67.

[27] J. Rossignac and P. Borrel. Multi-resolution 3D approximations for rendering. *Modeling in Comput. Graphics*, (1993), 455–465.

[28] W. Schröder, J. Zarge and W. Lorensen. Decimation of triangle meshes. *Proc. SIGGRAPH 92*, (1992), 65–70.

[29] G. Taubin. A signal processing approach to fair surface design. *Proc. SIGGRAPH 95*, (1995), 351–358.

[30] G. Turk. Re-tiling polygonal surfaces. *Siggraph 92*, (1992), 55–64.

[31] G. Turk and M. Levoy. Zippered polygon meshes from range images. *Proc. SIGGRAPH 94*, (1994), 311-318.

[32] S. P. Uselton. A survey of surface reconstruction techniques. *4th Ann. Conf. Natl. Comput. Graphics Assoc.*, June 1983. triangulated surfaces. *Proc. SIGGRAPH 94*, (1994), 247–256.

[33] D. Zorin and P. Schröder. Subdivision for modeling and animation. *SIGGRAPH 99 Course Notes*.

[34] http://www.cis.ohio-state.edu/∼tamaldey/cocone.html

[35] http://www.cgal.org

Figure 10: Upper row shows reconstructed surfaces with boundary samples (colored red) detected by BOUNDARY on the data sets ENGINE, CACTUS, OILPUMP and MONKEYSADDLE. The middle row shows the reconstruction of the data sets FOOT, MANNEQUIN, ROCKER and BUNNY. The bottom row shows the reconstruction after decimating these data sets with $\rho = 0.3$.