Delaunay Based Shape Reconstruction from Large Data

Tamal K. Dey Joachim Giesen James Hudson

Ohio State University, Columbus, OH 43210, USA

Abstract

Surface reconstruction provides a powerful paradigm for modeling shapes from samples. For point cloud data with only geometric coordinates as input, Delaunay based surface reconstruction algorithms have been shown to be quite effective both in theory and practice. However, a major complaint against Delaunay based methods is that they are slow and cannot handle large data. We extend the COCONE algorithm to handle supersize data. This is the first reported Delaunay based surface reconstruction algorithm that can handle data containing more than a million sample points on a modest machine.

Keywords: Computational Geometry, Curves & Surfaces, Geometric Modeling, Mesh Generation, Polygonal Mesh Reduction, Polygonal Modeling, Shape Recognition.

1 Introduction

Shape modeling is an integral part of many visualization problems. *Surface reconstruction* which computes a piecewise linear interpolant through a set of discrete points that sample a surface provides a flexible and robust paradigm for this purpose. This linear approximation is good enough for most of the visualization needs. If higher order continuity is required, various subdivision [13, 28] and fairing schemes [25] can be used on the initial control mesh generated by surface reconstruction.

A number of algorithms have been proposed for the surface reconstruction problem in recent years. The results of [5, 6, 7, 10, 15, 18, 26, 27] provide the necessary foundation for the problem. Very recently, starting with the CRUST algorithm of [2], few other algorithms have been designed that provide theoretical guarantees with empirical support [1, 3, 4, 9, 11]. All these new algorithms are based on Voronoi diagrams and their dual Delaunay triangulations. If the input is equipped with additional information such as the estimation of surface normals, or matrix adjacency among the sample points from range scanners, efficient algorithms exploiting these information can be used such as the ones proposed in [8, 10, 17, 19]. However, in absence of any such extra information, Voronoi/Delaunay based methods seem to be very effective if not the most.

The major complaint against Voronoi/Delaunay based methods is that they are slow and face difficulty in handling large data with the current computing resources. Two factors worsen the matter. First, the complexity of the Delaunay triangulation may be quadratic in terms of the input size. Second, the computation of the Voronoi diagrams or Delaunay triangulations is susceptible to numerical errors. As a result, additional computations against possible numerical errors are absolutely necessary for robust computations. Although faster machines with larger memory can and will push the range of acceptable data sizes to greater limits, the demand for handling even higher range will always exist. So, we need an *algorithmic solution* which can supplement the gain made by improved hardware technology. In this paper we propose such an approach for a Delaunay based surface reconstruction. Our experiments show that the proposed algorithm can handle data in the range of million points with a moderately capable machine such as a PC with 733 MHz Pentium III processor and 512 MB memory. The gain in computing times are remarkable as exhibited in Table 1.

Algorithms for reconstructing surfaces from large data have been proposed in the past [8, 10, 19, 22]. These algorithms avoid three dimensional Delaunay triangulations and can do so by using special information such as surface normals that come with the input or by exploiting properties of the data like uniformity. However, the need for a Delaunay based reconstruction with large data still exists; mainly because such a method can be applied to a more general input that may not have any special information.

Remarkable achievement has been made in the MICHELANGELO project [10, 22] where an implicit surface is built using a distance function. Surface normals from several range scans are used to compute the distance function. Kobbelt and Botsch [19] use the hardware projection to compute a triangulation from each range scan and then stitch them together. This gives a very fast reconstruction for each range scan. It faces difficulty if the projection of a scan self-overlap. Also, consistent stitching of adjacent scans poses problems. Current methods for stitching are based on heuristics and often create artifacts in the surface. The ball pivoting algorithm of [8] was used in the Pietà project at IBM to handle millions of sample points. This algorithm builds the surface incrementally by rolling a ball over the sample points. It requires that the sample points be equally dense everywhere and each sample point have a surface normal. In [17] Gopi, Krishnan and Silva projected a point with its neighbors on a 2D plane and then computed the 2D Delaunay triangulation of the set before lifting it to 3D.

In this paper we extend a Delaunay based reconstruction algorithm called COCONE to handle large data. This algorithm turned out to be very effective within the range of 100K points, but data with close to a million points are almost impossible to handle. This serious restriction is imposed by the time and memory requirement for computing the Delaunay triangulation of the entire point set. In particular, disk swapping with random data access in a global Delaunay triangulation becomes a serious bottleneck. We solve this problem by giving up the computation of a global Delaunay triangulation of the entire set of sample points. Instead, we take a divide-and-conquer approach.

We partition the entire set of sample points into smaller clusters using an octree subdivision. Then, we apply our COCONE algorithm on each of these clusters separately. Of course, the question of matching the surface patches together arises. We do not adopt stitching which is the source of troubles in earlier approaches. Instead we exploit a specific property of COCONE which achieves the stitching automatically. Each cluster in an octree box is padded with

sample points from neighboring boxes that allow the necessary triangles for stitching to be computed consistently over all boxes. The results of our experiments are illustrated with a number of large data that are available over the Internet. To our knowledge this is the first surface reconstruction algorithm that can compute with commonplace resources the output surface as a subcomplex of the three dimensional Delaunay triangulation from data in the range of a million points.

2 Definitions and preliminaries

Our goal is to eliminate the global Delaunay/Voronoi diagram computation of the entire sample while reconstructing the surface with the COCONE algorithm. To this end we partition the data with an octree subdivision and adapt carefully all steps of the COCONE algorithm [3, 11] on each cluster. We need some definitions for further expositions. Let P denote the input point set sampling a smooth surface $S \subset \mathbb{R}^3$. Let V_P be the Voronoi diagram of P and V_p denote the Voronoi cell of a sample point $p \in P$. We use the notation $\angle(\mathbf{u}, \mathbf{v})$ to denote the acute angle between the lines supporting two vectors $\mathbf{u}$ and $\mathbf{v}$. The concept of *poles* was introduced in [2] to approximate the normals at the sample points.

Poles: The farthest Voronoi vertex p^+ in V_p is called the positive *pole* of p. 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.

An important observation made in [2] is that the pole vector $\mathbf{v}_p$ approximates the normal $\mathbf{n}_p$ to S at the sample p up to the orientation. This means the plane with $\mathbf{v}_p$ as normal at p approximates the tangent plane at p. The following definition of a *cocone* captures a neighborhood of the part of this tangent plane that lies inside the Voronoi cell of p.

Cocone: The set $C_p = \{y \in V_p : \angle((y-p), \mathbf{v}_p) \geq \frac{3\pi}{8}\}$ is called the cocone of p. In words, C_p is the complement of a double cone (clipped within V_p) centered at p with an opening angle $\frac{3\pi}{8}$ around the axis aligned with $\mathbf{v}_p$. See Figure 1 for an example of a cocone.

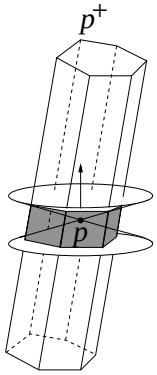


Figure 1: A Voronoi cell together with the normalized pole vector and the cocone (shaded).

Restricted Voronoi diagram/Delaunay triangulation play a key role in asserting the correctness of our algorithm though their explicit computation is impossible in the absence of S.

Restricted Voronoi diagram/Delaunay triangulation: The intersection of the Voronoi diagram V_P and the surface S is called the restricted Voronoi diagram of P on S. The set $V_{p,S} = V_p \cap S$ is called the restricted Voronoi cell of p. See Figure 2. Two sample points define a restricted Delaunay edge, three sample points define a restricted Delaunay triangle and four sample points define a restricted Delaunay tetrahedron if their restricted Voronoi cells have non empty common intersection. The complex built from restricted Delaunay simplices is called the restricted Delaunay triangulation.

Restricted Voronoi neighbor: A Voronoi neighbor q of a sample p on the restricted Voronoi diagram is called its *restricted Voronoi neighbor*. It means that $V_{p,S} \cap V_{q,S} \neq \emptyset$. In Figure 2, p and q are restricted Voronoi neighbor of each other.

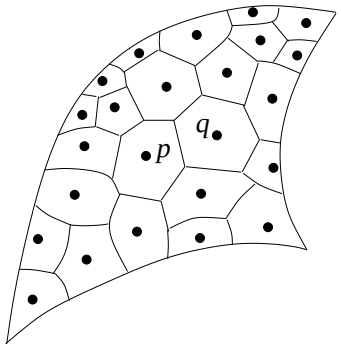


Figure 2: Intersection of V_P with the surface S; p and q are restricted Voronoi neighbors.

3 Local reconstruction

Our algorithm for large data localizes the COCONE algorithm by a divide-and-conquer approach. The COCONE as an improvement of CRUST relies on the fact that the sample P is sufficiently dense for the surface S. This dense sampling is defined with ε-sampling as introduced in [2]. The *medial axis* of S is defined as the closure of all points in $\mathbb{R}^3$ that have more than one closest point to S. The *local feature size* is a function $f : S \to \mathbb{R}$ where $f(p)$ is the least distance to the medial axis for any point $p \in S$. A nice property of $f(\cdot)$ is that it is 1-Lipschitz meaning $f(p) \leq f(q) + ||p - q||$ for any two points p and q on S. A sample P is an ε-sample of S if each point $x \in S$ has a sample point within $\varepsilon f(x)$ distance. Typically, dense samples have $\varepsilon < 0.4$ in practice.

Using the result of [16] Amenta and Bern show that for dense sample the restricted Delaunay triangulation of S is homeomorphic to S. That is, the set of triangles dual to the Voronoi edges intersecting the surface S form a surface homeomorphic to S. Furthermore, the restricted Voronoi cell $V_{p,S}$ is locally flat and lies close to the tangent plane at p. The cocone C_p approximates this tangent plane with its thickness accommodating the estimate of the normal $\mathbf{n}_p$ with the pole vector $\mathbf{v}_p$. Thus, the Voronoi edges that intersect the surface S must also intersect the cocone C_p as proved in [3]. The COCONE algorithm builds upon this observation.

The original COCONE algorithm [3] was designed for smooth surfaces without any boundary. It chooses all triangles incident to a sample point p whose dual Voronoi edges intersect the cocone C_p. But, this causes problems for surfaces that may have non-empty boundaries. A boundary detection step is necessary to handle surfaces with non-empty boundary. In [11] such an algorithm is proposed which detects the sample points that represent the boundaries in S. We call these points *boundary samples* and disallow them to choose any triangle in the COCONE algorithm. This gives the following four major steps of the COCONE algorithm.

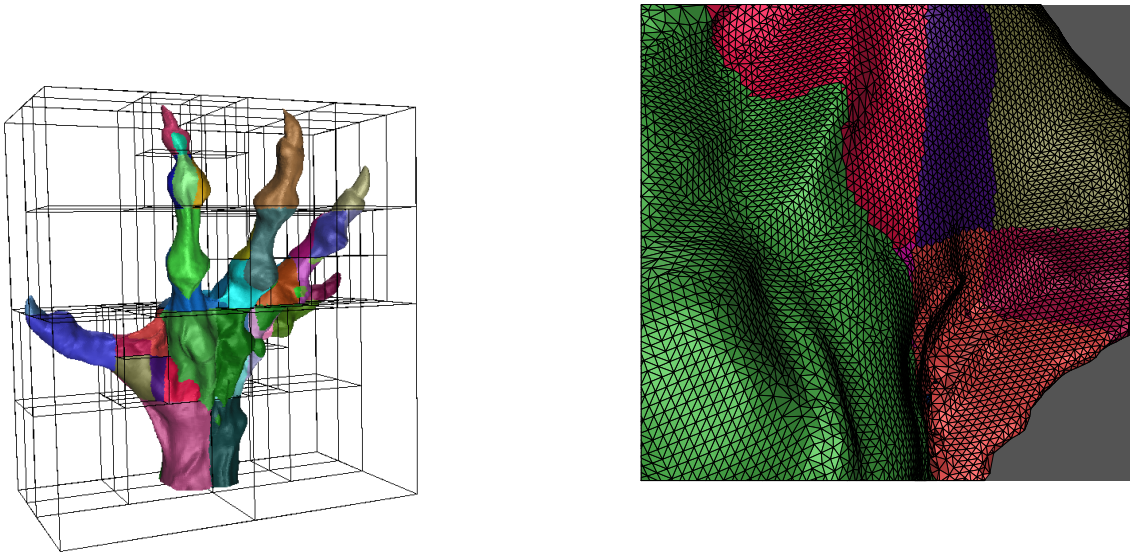


Figure 3: Reconstructed HAND with the octree boxes and a zoom showing the matching of adjacent surface patches. Surface patches from different boxes are colored differently.

COCONE(P)

(1) Compute the Voronoi diagram V_P.

(2) Determine the set of boundary samples.

(3) If p is not a boundary sample, compute the set of *candidate triangles*, T, that are dual to the Voronoi edges intersecting C_p.

(4) Extract a manifold from T using prune and walk as mentioned in [2] and detailed in [12].

In [3] it is shown that, step 4, the manifold extraction step, computes a manifold homeomorphic and geometrically close to S from the set of candidate triangles due to the following two properties.

Property 1 *Each candidate triangle t is small, i.e., the circumscribing circle of t has a radius $O(\varepsilon)f(p)$ where p is any vertex of t.*

Property 2 *All restricted Delaunay triangles are included in the set of candidate triangles.*

We argue that we can compute a set of candidate triangles satisfying Property 1 and 2 without computing the entire Voronoi diagram which is the most time and memory consuming step. In COCONE the candidate triangles for each sample point p span p and some of its Voronoi neighbors including the restricted ones. Our goal is to approximate these neighbors from a Voronoi diagram computed locally with fewer sample points. Let $\tilde{P} \subset P$ be such a subsample where $p \in \tilde{P}$. It turns out that we only need to ensure that the set of all restricted Voronoi neighbors of p, R_p, be present in $\tilde{P}$ in order to guarantee Property 1 and 2.

Now we face the problem of determining the set R_p, since S is unknown. As a remedy we overestimate R_p by taking a subsample that has a cluster of sample points around p. We use *octree subdivisions* and a *padding* to compute these clusters.

3.1 Octree subdivision

First, a root box is computed that contains all sample points. A generic step in this subdivision proceeds to split a box B into eight equal sub-boxes if B contains more sample points than a prescribed threshold Δ. Consequently, all boxes at the leaf level contain less than Δ sample points. In our experiments values of Δ in the range of few thousand sample points produce good results. See Figure 3 for an example where the octree boxes are shown along with the reconstructed surface.

3.2 Voronoi computation (step 1)

We can proceed with the Voronoi diagram of all sample points in a leaf box, say B, of the octree for approximating the surface patch going through the sample points in B. The trouble with this strategy is that some of the sample points that were non boundary for S may turn into boundary ones for the surface patch $S \cap B$. For sample points lying close to the boundary of $S \cap B$ it might happen that B does not contain all their restricted Voronoi neighbors and thus their cocones might not be computed correctly. We overcome this problem by padding more sample points around the boundary samples by taking the sample points from the adjacent boxes of B.

Padding

If we take all sample points from the adjacent boxes, the size of the input for local Voronoi diagram computation may be unnecessarily large and we could loose the advantage of computing the local Voronoi diagrams. As a remedy, we take a fraction of the sample points from the adjacent boxes of B while computing the local Voronoi diagram for its sample points. We subdivide each adjacent box B' of B further up to a level ℓ to produce boxes of size $\frac{1}{2^\ell}$th of B'. Let X denote the set of all such boxes bordering B that are produced as a result of this further subdivision. Denote the *extended box* $B \cup X$ as E_B. We take all the sample points $P_B = P \cap E_B$ from this extended box of B when we consider the sample points in B.

Our experiments show that ℓ in the range of $3-4$ produces good results.

3.3 Boundary samples (step 2)

Some sample points in P_B lie close to the boundary of the surface patch $S \cap E_B$. These boundary samples need to be detected as they may represent some of the real boundaries of S. We employ the boundary detection algorithm BOUNDARY of [11] to detect these boundary samples.

3.4 Candidate triangles (step 3)

Let p be any sample point contained in a leaf box B. We consider the sample points $P_B = P \cap E_B$ in the extended box E_B while computing the Voronoi diagram for the sample points in B. This ensures that p has its restricted Voronoi neighbors included in this larger set of samples. Accordingly, as argued later, the candidate triangles chosen by p satisfy Property 1 and 2 necessary for reconstruction.

3.5 Manifold extraction (step 4)

One possibility of executing the manifold extraction step is to adopt a global strategy. We collect all candidate triangles for each sample point over all boxes and then run a global prune and walk over this set of candidate triangles as in the original COCONE algorithm.

Combining all steps together we have the modified COCONE algorithm which we call SUPERCOCONE. All steps except the last one are obviously parallelizable.

SUPERCOCONE(P, Δ, ℓ)

(1) Compute an octree subdivision of the set P of sample points with parameter Δ. Also, compute the extended box E_B for each leaf box B using the parameter ℓ.

(2) For each leaf box B, perform steps 1,2 and 3 of COCONE on the extended set P_B. Retain a triangle in the candidate set T only if it has a vertex in B.

(3) Extract a manifold surface from T by the prune and walk method of [2, 3].

3.6 Theoretical justification

We argue that the candidate triangles computed by SUPERCOCONE satisfy Property 1 and 2. To distinguish between the Voronoi diagrams of P and the subsample P_B we use $\tilde{V}_p, \tilde{C}_p$ and $\tilde{\mathbf{v}}_p$ to denote the Voronoi cell, cocone and the pole vector of p respectively in the Voronoi diagram V_{P_B}.

First we observe that if the gap between a box B and its extension E_B is sufficiently large, then all restricted Voronoi neighbors of a point $p \in B$ are included in P_B. Actually, only a small extension suffices due to Lemma 1. The proof follows from the fact that, for each restricted Voronoi neighbor q of p, there must be a point $x \in V_{p,S}$ equidistant from p and q. This point has p as nearest sample and thus lies within $\varepsilon f(x)$ distance from p due to the ε-sampling condition. Now apply the Lipschitz property of $f(\cdot)$ to show $||p - q|| \leq 2\varepsilon f(x) \leq \frac{2\varepsilon}{1-\varepsilon} f(p)$.

Lemma 1 *Let q be a restricted Voronoi neighbor of a sample point $p \in P$. Then $||p - q|| \leq \frac{2\varepsilon}{1-\varepsilon} f(p)$.*

Assumption 1 *The gap between E_B and B is at least $\frac{2\varepsilon}{1-\varepsilon} f(p)$ long for any point $p \in B$.*

Since ε is small, padding B with some constant fraction of its neighboring boxes suffices to satisfy this assumption in practice. It follows that each point p in a box B has all its restricted Voronoi neighbors in the set P_B. This condition is sufficient to extend a proof of [2] for Lemma 2.

Lemma 2 *Let x be any point in $\tilde{V}_p$ so that $||x - p|| > \mu f(p)$. Then, the acute angle between the lines supporting the vectors $x - p$ and $\mathbf{n}_p$ is $\sin^{-1} \frac{\varepsilon}{\mu(1-\varepsilon)} + \sin^{-1} \frac{\varepsilon}{1-\varepsilon}$.*

Consider the ball touching S at p tangentially and with the center on the medial axis. This ball must be empty of other surface point. Therefore, its center must belong to V_p which is at least $f(p)$ distance away from p. Then, by definition the pole p^+ is at least $f(p)$ distance away from p. Applying this observation to Lemma 2 we obtain:

Lemma 3 *The acute angle between the lines supporting the pole vector $\tilde{\mathbf{v}}_p$ and the normal $\mathbf{n}_p$ is $2 \sin^{-1} \frac{\varepsilon}{1-\varepsilon}$.*

By definition the vector $x - p$ for any point x in the cocone $\tilde{C}_p$ makes an angle more than $\frac{3\pi}{8}$ with the pole vector $\tilde{\mathbf{v}}_p$. It follows from Lemma 3 that the acute angle between the supporting lines of $x - p$ and $\mathbf{n}_p$ is more than $\frac{3\pi}{8} - O(\varepsilon)$. We can use the contrapositive of Lemma 2 to assert that $||x - p|| = O(\varepsilon) f(p)$ as done in [3]. This implies that each candidate triangle t has a point x that centers a circumscribing sphere of t with radius $O(\varepsilon) f(p)$ where p is any of its vertex. Thus, we have Property 1 for all candidate triangles computed by SUPERCOCONE.

In order to prove Property 2 we can use the proof in [3] to assert that $V_{p,S}$ is completely contained in the cocone $\tilde{C}_p$.

Lemma 4 *For any point $p \in B$ the restricted Voronoi cell $V_{p,S}$ lies inside $\tilde{C}_p$.*

Lemma 4 implies that the Voronoi edges intersecting S also intersect $\tilde{C}_p$. It means that restricted Delaunay triangles incident to a point p are computed when we consider the box B containing p.

Therefore, as we stated earlier, the manifold extraction step computes a manifold surface N from the candidate triangles. Further, since circumcircle of any triangle is small, each point on the output surface is within a small distance of a sample point on S. We make these claims precise in the following theorem.

Theorem 1 *Given a sample P from a smooth compact manifold without boundary in $\mathbb{R}^3$, SUPERCOCONE computes a triangulated manifold N with the properties:*

1. *N is homeomorphic to S.*
2. *Each point x on N has a point on S within $O(\varepsilon) f(x)$ distance.*

3.7 Implementation of manifold extraction

Although, theoretically, a global prune and walk to extract a manifold seems sound, we face a practical problem while dealing with large data. The walk phase of the manifold extraction step is sensitive to numerical errors. In particular, skinny flat tetrahedra in the Delaunay triangulation called *slivers* cause difficulty when carrying out a consistent walk. In the COCONE software [30] we avoided this problem by replacing the geometric decisions made on numerical computations by the combinatorial decisions made on the structure of the Delaunay triangulations. The details of this robustness issue are given in [12]. Unfortunately, this needs the Delaunay triangulation of the entire sample if we adopt a global strategy for manifold extraction.

In the implementation we cope with this problem by giving up the global manifold extraction. Instead, while considering the box

B we extract a manifold from the candidate triangles chosen by the sample points in P_B. Then we retain only those triangles that have at least a vertex in B. We argue that this achieves the result of global manifold extraction. To see this we need to understand the pruning and walk in more details.

The pruning in the manifold extraction step deletes any triangle iteratively that is incident to a sharp edge. An edge is *sharp* if all triangles incident to it are contained within an angle of $\frac{\pi}{2}$ around it. In particular, any edge with a single triangle incident to it is sharp. Subsequent to the pruning step no triangle with sharp edge exists and a depth first walk on the *outside* (or *inside*) of the remaining triangles outputs a manifold surface. The pruning step cannot delete all triangles in succession since the restricted Delaunay triangles which are in T cannot be incident to any sharp edge and their underlying space is a manifold homeomorphic to S.

Although it is possible to propagate a cascaded pruning far away from the triangle that initiated it, this propagation migrates only up to a few triangles away in practice. Thus, the padding for B in E_B helps to initiate all pruning that would have occurred globally at least for the triangles incident to a point in B. This means the pruning applied to the extended box E_B simulates the global pruning for all triangles with a vertex in B. Consequently the walk on the pruned set in E_B also simulates the global walk at least for the triangles incident to a point in B. Of course a consistent walk on the outside of the triangles over all boxes needs a globally consistent orientation of the triangles. We achieve this by passing the orientation information from a box to its adjacent boxes.

In our experiments, we observe that the surface patches computed in this manner in each box match with the adjacent surface patches computed in the adjacent boxes. See the zoomed hand in Figure 3 for an example. All other tested examples also confirm this claim, see Figure 4. Surface patches from different boxes are colored differently.

4 Experiments

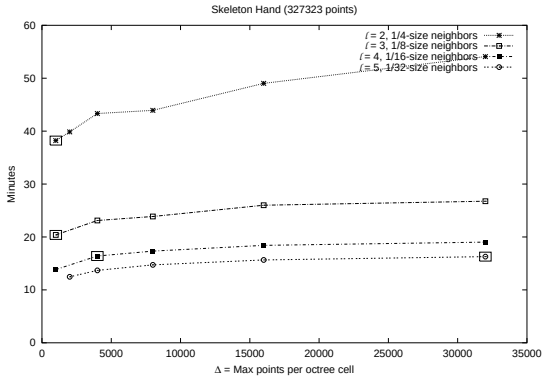
We experimented with several large models on a PC with 733 MHz Pentium III processor and 512 MB RAM. The code is written in C++ and makes use of the computational geometry algorithms library CGAL [29] for computing the Delaunay triangulations and other geometric predicates. CGAL provides template code that allows us to run the provided algorithms with different number types. We found that floating point arithmetic runs fast, but numerical errors may cause unreliable output. On the other hand, exact arithmetic computations are reliable but consume much more time. Filtered floating point arithmetic provided by CGAL turns out to be the right number type for our use. With filtered floating point arithmetic we get reliable results in roughly twice the time needed when using pure floating point arithmetic. We chose to accommodate this increase in time instead of compromising the output quality.

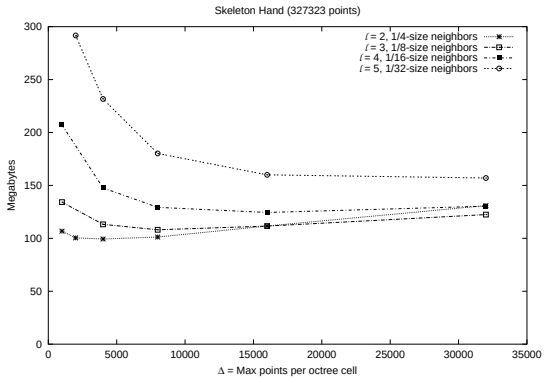
In order to determine a suitable level of subdivision in the octree, we plotted the graphs of computing time and memory usage vs. Δ. We also plotted a curve for different levels ℓ of subdivision of the neighbor boxes. We observed that $\Delta = 16,000$ and $\ell = 4$ provide good results for all data. The first node on each curve that corresponds to a valid reconstruction is marked with a box in time plots.

4.1 Examples

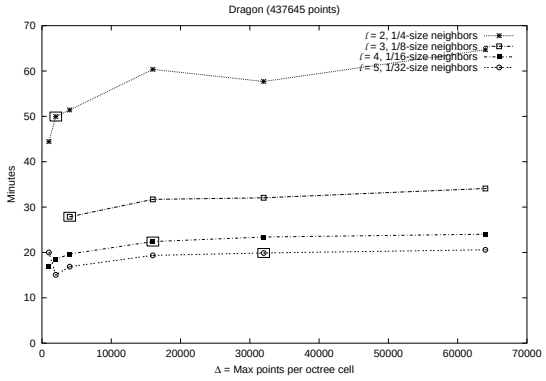
We experimented with six large data sets collected from [31] and [32] with sizes ranging from 300K points to 3.5 million points.

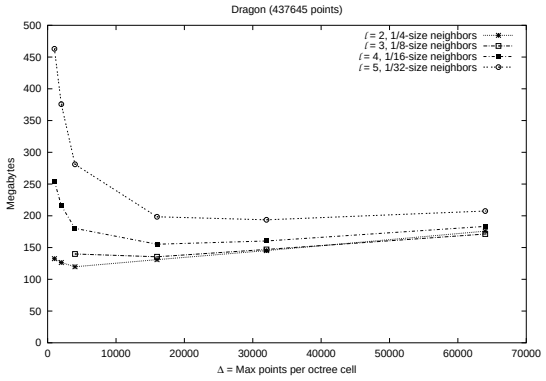
Data set SKELHAND: This data set has 327,323 points. It takes 100 minutes to reconstruct with a global Delaunay triangulation. SUPERCOCONE takes only 15 minutes with $\Delta = 16,000$ and $\ell = 4$. See the plots below and the output surface in Figure 4.



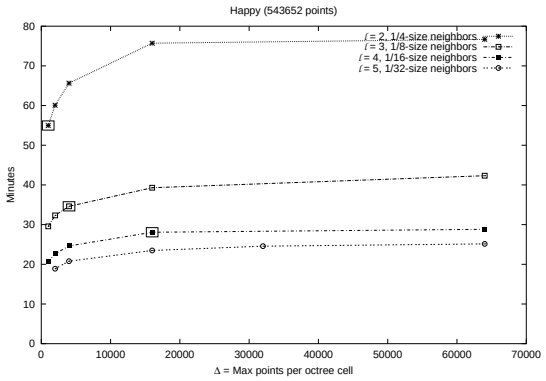


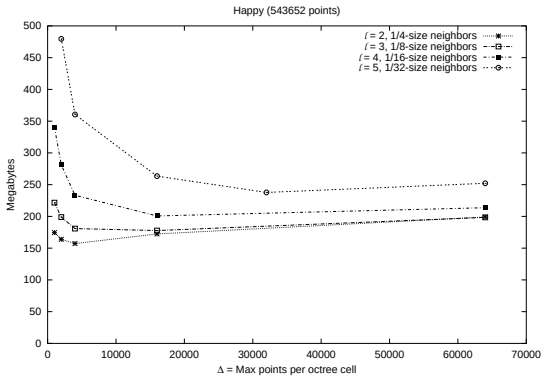
Data set DRAGON: This data set has 437,645 points. Surface reconstruction with a global Delaunay triangulation in COCONE takes 46 minutes. This time is considerably smaller than that of the data set SKELHAND though its size is larger. We suspect that the size of the Delaunay triangulation for SKELHAND data set is considerably larger than that for this data set. SUPERCOCONE takes 18 minutes with $\Delta = 16,000$ and $\ell = 4$. The graphs showing the time and memory requirements with different values of Δ and ℓ are shown in the plots below. The output surface is shown in Figure 4.



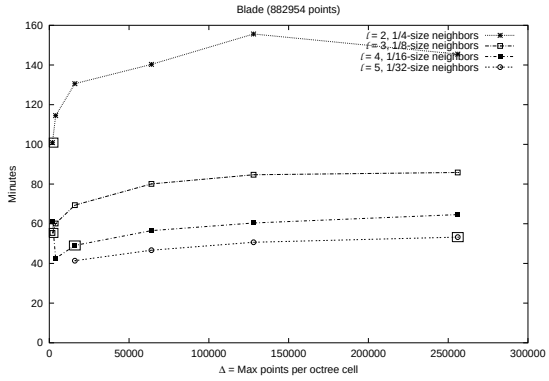


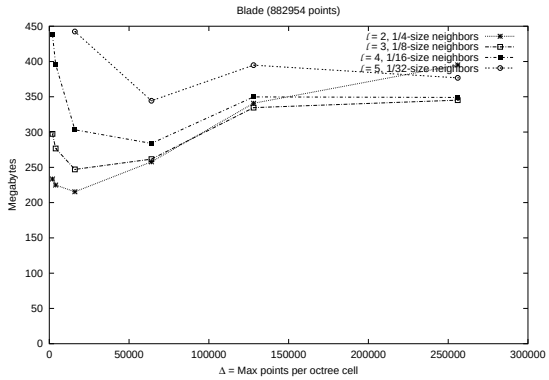
Data set HAPPY: This data set has 543,652 points. It takes 143 minutes with a global Delaunay triangulation computation, whereas SUPERCOCONE takes 28 minutes with $\Delta = 16,000$ and $\ell = 4$. See the figure below for the plots and Figure 4 for the output surface.



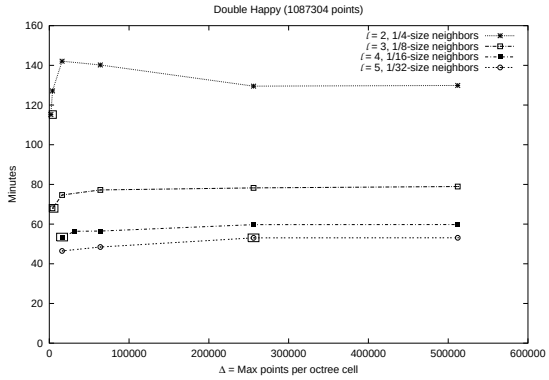


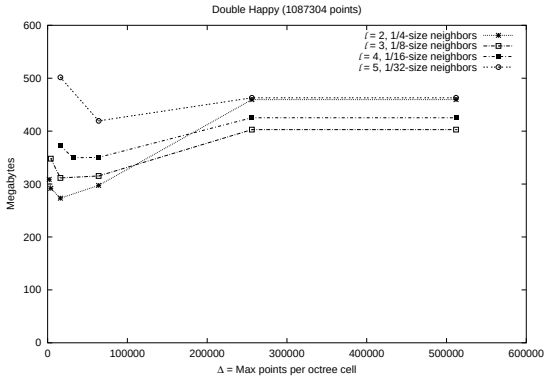
Data set BLADE: This data set has 882,954 points. The reconstruction takes 27 hours when we used a global Delaunay triangulation. The memory requirement shot up to more than 800MB which exceeded the memory capacity of the machine on which we ran our experiments. Remarkably, SUPERCOCONE takes only 50 minutes with $\Delta = 16,000$ and $\ell = 4$. See the figure below for the plots and Figure 4 for the output surface.



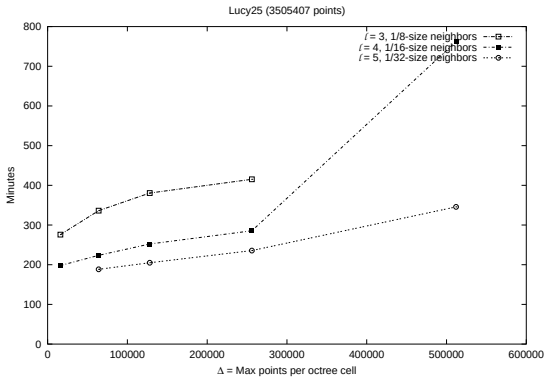


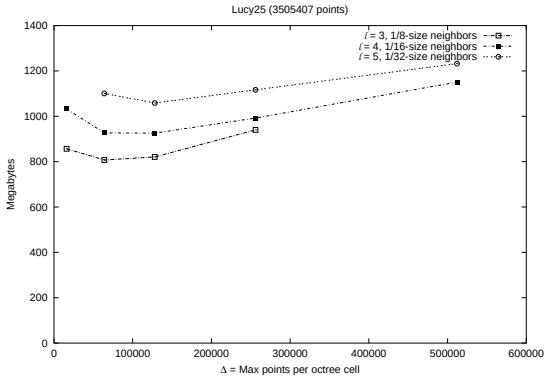
Data set DOUBLEHAPPY: In order to create a data set with roughly one million points, we took two copies of the HAPPY data set separated by a small distance. As a result this data set has 1,087,304 points. We could not finish the reconstruction within 10 days with a global Delaunay triangulation. SUPERCOCONE takes only 53 minutes. See the figure below for the plots and Figure 4 for the output surface.





Data set LUCY25: We took the LUCY data from the Stanford repository [31] which has approximately 14 million points. We pruned this data randomly to create a point set of 3.5 million points. This is because we restricted our experiments to a PC of 512MB memory which is not enough to hold the model of 14 million vertices. SUPERCOCONE finished the reconstruction on this data in 198 minutes. See the figure below for the plots and Figure 4 for the output surface.





4.2 Computing times

From the plots we observe that, as Δ goes down, the computing time decreases. The extended box may not provide enough padding to the sample points in B with decreasing Δ. As a result, the reconstructed surface may develop undesirable "holes". On the other hand as Δ increases, the computing time also increases. We marked the first node from left on each curve in the time vs. Δ plots with a small surrounding box which gives a valid surface without any "hole".

As expected, the computing times decrease as the subdivision level ℓ in neighbor cells increases. This is because less number of points are added in the extended box E_B. But, increased values of Δ can compensate for this deficiency. This is why the marked nodes appear later on the curves as we increase ℓ. Observing over a set of data, we conclude that $\Delta = 16,000$ and $\ell = 4$ produce a good reconstruction within acceptable computing time for most of the data that we tested on our PC. The running times for our examples using $\Delta = 16,000$ and $\ell = 4$ are summarized in Table 1.

object	number of points	Global Delaunay time (min.)	SUPER COCONE time (min.)
SKELHAND	327,323	100	15
DRAGON	437,645	46	18
HAPPY	543,652	143	28
BLADE	882,954	1615	50
DOUBLEHAPPY	1,087,304	unfin	53
LUCY25	3,505,407	unfin	198

Table 1: Time data.

Note that the BLADE took 27 hours with a global Delaunay triangulation whereas SUPERCOCONE finished the reconstruction in less than an hour. We could not finish DOUBLEHAPPY and LUCY25 with a global Delaunay triangulation even after 10 days whereas SUPERCOCONE finished them in 53 minutes and 198 minutes respectively.

4.3 Memory

Runtime memory consumption plays a vital role in handling large data. If a global Delaunay triangulation is used, the data is accessed randomly, and, with large amount of data, disk swaps dominate any fruitful computation crippling the progress heavily. Our divide-and-conquer approach localizes the memory access in octree boxes reducing the disk swaps substantially. That is why we could finish reconstruction from a sample with more than million points even with a 512MB memory machine.

The level of octree subdivision affects the memory usage of the algorithm. If Δ is too small, the number of octree cells increases tremendously. As a result, the memory usage increases very sharply towards the lower range of Δ. As Δ increases, the memory usage due to octree subdivision decreases, but the Delaunay triangulations start consuming more memory. In the extreme case where there is a single box, i.e. the input is not subdivided at all, the memory requirement may be larger than the available memory. Such a case happens for the data BLADE, DOUBLEHAPPY and LUCY25. The memory consumption for our examples using $\Delta = 16,000$ and $\ell = 4$ are summarized in Table 2. Notice that all memory data is obtained by a system tool that reports the virtual memory used by a process. This explains why we have virtual memory requirement more than 512MB for some data sets even with a 512MB memory machine.

object	number of points	Global Delaunay mem.(MB)	SUPER COCONE mem.(MB)
SKELHAND	327,323	356	127
DRAGON	437,645	475	158
HAPPY	543,652	595	205
BLADE	882,954	> 800	310
DOUBLEHAPPY	1,087,304	unfin	381
LUCY25	3,505,407	unfin	1010

Table 2: Memory data.

5 Conclusions

We present a surface reconstruction algorithm along with its implementation which for the first time demonstrates that surfaces can be reconstructed as a subcomplex of the Delaunay triangulation from unorganized point clouds that have more than a million points. The gain in time and memory consumption is substantial as our experiments show. Theoretical guarantees in line of the work in [2, 3] can be proved with the ε-sampling assumption.

A model produced from a large data can be too big to fit into main memory. In that case surface patches generated by SUPERCOCONE can be stored individually and memory efficient simplification [21, 24] can be used, or they can be stored in external memory and out-of-core simplification methods as in [20] can be used to simplify the model.

It is interesting to note that SUPERCOCONE can be made even faster by the use of parallel computation. Surface patches in boxes can be built in parallel since they do not need any global Delaunay triangulation or manifold extraction. We are hopeful that data in the range of a billion points can be handled with the parallelization. Currently, work is under progress in this direction.

References

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

[2] N. Amenta and M. Bern. Surface reconstruction by Voronoi filtering. *Discr. Comput. Geom.*, **22**, (1999), 481–504.

[3] N. Amenta, S. Choi, T. K. Dey and N. Leekha. A simple algorithm for homeomorphic surface reconstruction. *Proc. 16th. ACM 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] D. Attali. r-regular shape reconstruction from unorganized points. *Proc. 13th ACM Sympos. Comput. Geom.*, (1997), 248–253.

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

[7] F. Bernardini, C. Bajaj, J. Chen and D. Schikore. Automatic Reconstruction of 3D CAD Models from Digital Scans. *Int. J. on Comp. Geom. Appl.*, **9** (1999), 327–334.

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

[9] J. D. Boissonnat and F. Cazals. Smooth surface reconstruction via natural neighbor interpolation of distance functions. *Proc. 16th. ACM Sympos. Comput. Geom.*, (2000), 223–232.

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

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

[12] T. K. Dey, J. Giesen and W. Zhao. Robustness issues in surface reconstruction. *Proc. Intl. Conf. Comput. Sci.*, San Fancisco, Calif., May 28-30, (2001), `http://www.cis.ohio-state.edu/~tamaldey/papers`.

[13] M. Eck and H. Hoppe. Automatic reconstruction of *B*-spline surfaces of arbitrary topological type. *Proc. SIGGRAPH 96*, (1996), 325–334.

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

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

[16] H. Edelsbrunner and N. Shah. Triangulating topological spaces. *Proc. 10th ACM Sympos. Comput. Geom.*, (1994), 285–292.

[17] M. Gopi, S. Krishnan and C. Silva. Surface reconstruction based on low dimensional localized Delaunay triangulation, *Eurographics 2000*, (2000).

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

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

[20] P. Lindstrom. Out-of-core simplification of large polygonal models. *Proc. SIGGRAPH 00*, (2000), 259–262.

[21] P. Lindstrom and G. Turk. Fast and memory efficient polygonal simplification. *IEEE Visualization 98*, (1998), 279–286.

[22] M. Levoy, K. Pulli, B. Curless, S. Rusinkiewicz, D. Koller, L. Pereira, M. Ginzton, S. Anderson, J. Davis, J. Ginsberg, J. Shade and D. Fulk. The digital Michelangelo project: 3D scanning of large statues. *Proc. SIGGRAPH 00*, (2000), 131–144.

[23] C. Loop. Smooth spline surfaces over irregular meshes. *Proc. SIGGRAPH 94*, (1994), 303–310.

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

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

[26] S. P. Uselton. A survey of surface reconstruction techniques. *4th Ann. Conf. Natl. Comput. Graphics Assoc.*, June 1983.

[27] R. T. Whitaker. A level-set approach to 3D reconstruction from range data. *Intl. J. Comput. Vision*, **29**, (1998), 203–231.

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

[29] `http://www.cgal.org`

[30] `http://www.cis.ohio-state.edu/~tamaldey/cocone.html`

[31] `http://graphics.stanford.edu/data/3Dscanrep`

[32] `http://www.cc.gatech.edu/projects/large_models`

Figure 4: Reconstructed surfaces from SKELHAND (327,323 points), DRAGON (437,645 points), HAPPY (543,652 points), BLADE (882,954 points), DOUBLEHAPPY (1,087,304 points) and LUCY25 (3,505,407 points). Surface patches from different boxes are shown with different colors; see the color plate.