Eurographics Symposium on Geometry Processing 2010
Olga Sorkine and Bruno Lévy
(Guest Editors)

Volume 29 (2010), Number 5

Localized Delaunay Refinement for Sampling and Meshing

Tamal K. Dey[1] Joshua A. Levine[2] Andrew Slatton[1]

[1] The Ohio State University, Columbus, OH, USA
[2] Scientific Computing and Imaging Institute, Salt Lake City, UT, USA

Abstract

The technique of Delaunay refinement has been recognized as a versatile tool to generate Delaunay meshes of a variety of geometries. Despite its usefulness, it suffers from one lacuna that limits its application. It does not scale well with the mesh size. As the sample point set grows, the Delaunay triangulation starts stressing the available memory space which ultimately stalls any effective progress. A natural solution to the problem is to maintain the point set in clusters and run the refinement on each individual cluster. However, this needs a careful point insertion strategy and a balanced coordination among the neighboring clusters to ensure consistency across individual meshes. We design an octtree based localized Delaunay refinement method for meshing surfaces in three dimensions which meets these goals. We prove that the algorithm terminates and provide guarantees about structural properties of the output mesh. Experimental results show that the method can avoid memory thrashing while computing large meshes and thus scales much better than the standard Delaunay refinement method.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Curve, surface, solid, and object representations

1. Introduction

The Delaunay refinement, a technique to iteratively sample locally furthest points from an input geometry, is a versatile tool for generating Delaunay meshes. Since its introduction by Chew [Che89] and Ruppert [Rup95] in 2D, the technique has been adopted to mesh a variety of three dimensional geometries such as polyhedra [She98], smooth surfaces [BO05, CDRR07] and volumes [ORY05], piecewise smooth surfaces [BO06, DLR05], and piecewise smooth complexes [CDR08]. The popularity of this technique is derived from its ability to render algorithms with provable guarantees as well as its ability to produce good quality meshes when combined with optimization techniques [ACSYD05, TWAD09, YLL*09].

Notwithstanding its use as a versatile mesh generation tool, the Delaunay refinement technique has been plagued with one shortcoming–it does not scale well with the mesh size. The state-of-the-art has improved with the current advances in Delaunay triangulation computations [ACR03, Dev02, ILSS06]. Still, we look for Delaunay refinement techniques whose scale is not limited by the particular Delaunay triangulation code being used. Some strategies to execute it faster [HMP06] and in parallel [NCC04] have been developed for special cases of polyhedra. However, the issue of scale for meshing surfaces with Delaunay refinement has not yet been dealt with. Since the method maintains a global Delaunay triangulation of the entire existing point set, it starts slowing down as the point set grows, and reaches a limping speed as the *available main memory* becomes insufficient to hold the entire Delaunay triangulation. A natural solution to this problem is to split the point set into clusters and operate on the Delaunay triangulations of the individual clusters. However, this solution incurs some fundamental difficulties. First of all, the termination of Delaunay refinement techniques depends on the property of Voronoi points being locally furthest from *all* existing points. If the entire point set is not used for building the Delaunay triangulation, this property cannot be guaranteed. Second, individual meshes computed for each cluster may not be consistent across clusters to provide a valid global mesh. These two problems are further complicated by the fact that the clusters need to be re-arranged as the point set grows.

In this paper we present an algorithm that adopts the Delaunay refinement of surfaces in three dimensions by local-

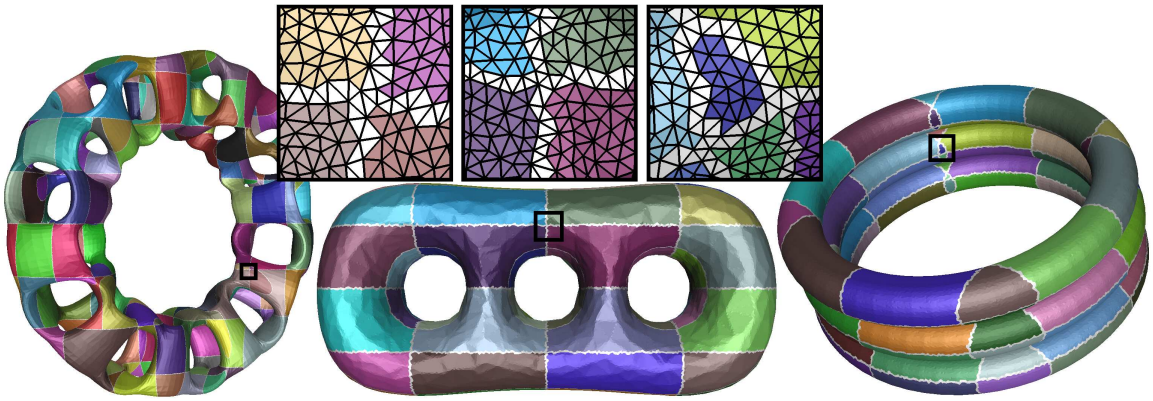

Figure 1: *Output meshes for (from left to right)* HOLEDRING, 3HOLES, *and* BRACELET. *Individual meshes in different nodes are colored differently. Highlights show triangles across node boundaries emphasizing the mesh consistency.*

izing it to octtree based clusters. The standard Delaunay refinement of surfaces [CDES01, BO05, CDRR07] maintains a restricted Delaunay triangulation comprised of triangles whose dual Voronoi edges intersect the surface. It repeatedly samples the surface with points where the Voronoi edges intersect the surface. When the algorithm terminates, it outputs a mesh which is homeomorphic (isotopic) to the input surface and is also geometrically close. In our case, we run the Delaunay refinement on a subset of points residing in an octtree node and some of its surrounding nodes. We modify the point insertion strategy and re-process some of the nodes to take care of both *termination* and *consistency*. Figure 1 shows some output meshes of our algorithm highlighting the consistency across node boundaries.

The algorithm runs with two positive user parameters λ and κ. The parameter λ determines the granularity of the output mesh. The parameter κ determines the number of points a node holds before it gets split. In a sense, by λ, the user controls the scale at which the surface is sampled, and by κ she controls the granularity of the octtree subdivision. By decreasing λ, potentially one may produce a large mesh risking the memory to accommodate a large Delaunay triangulation, but the risk can be mitigated by choosing an appropriate κ. We argue later in section 5 that this appropriate value of κ does not depend on the model or the choice of λ. Therefore, it can be estimated a priori for a given computing platform and be used for all future meshing. We do not assume that the user has a knowledge of the input feature scale and thus λ is not tied to it. Therefore, one cannot expect that the topology of the input surface is captured completely unless λ is sufficiently small. We take the approach in [DL09] to overcome this difficulty. We prove that our algorithm terminates and outputs a manifold mesh irrespective of the value of λ. However, if λ is sufficiently small, the output mesh becomes topologically equivalent to the input. Figure 2 shows how the geometry and topology of NEPTUNE are progressively captured with decreasing λ.

1.1. Notations

The Voronoi/Delaunay tessellations and their restrictions on a surface are used in our algorithm and its analysis. Let S be a set of points in $\mathbb{R}^3$. We denote the Voronoi diagram and Delaunay triangulation of S as $\mathrm{Vor}\,S$ and $\mathrm{Del}\,S$ respectively. The Voronoi diagram $\mathrm{Vor}\,S$ is a cell complex whose cells are k-dimensional Voronoi faces for $k = 0,\ldots,3$. The 0-, 1-, 2-, and 3-dimensional Voronoi faces are called Voronoi *vertices, edges, facets,* and *cells* respectively. The Delaunay triangulation $\mathrm{Del}\,S$ is a simplicial complex dual to $\mathrm{Vor}\,S$ where a k-simplex $t \in \mathrm{Del}\,S$ is dual to a $(3-k)$-dimensional Voronoi face which we denote as V_t. The 0-, 1-, 2-, and 3-dimensional Delaunay simplices are Delaunay *vertices, edges, triangles*, and *tetrahedra* respectively.

In our case, the point set S will be a set of sample points from a surface M embedded in $\mathbb{R}^3$. We assume M to be closed, that is, compact and without boundary. We are interested in a subcomplex of $\mathrm{Del}\,S$ consisting of Delaunay simplices whose dual Voronoi faces intersect M. It is called the *restricted Delaunay triangulation* of S with respect to M and is denoted $\mathrm{Del}\,S|_M$. Formally,

$$\mathrm{Del}\,S|_M = \{\sigma \in \mathrm{Del}\,S : V_\sigma \cap M \neq \emptyset\}.$$

By definition a Delaunay simplex is *restricted* if its dual Voronoi face intersects the surface. For example, the restricted triangles are the Delaunay triangles whose dual Voronoi edges intersect the surface. The restricted triangles and their dual Voronoi edges play an important role in our algorithm since the Delaunay refinement is carried out on these restricted triangles.

It is possible that the dual Voronoi edge of a restricted triangle $t \in \mathrm{Del}\,S|_M$ intersects the surface M at multiple

points. Each of these points is a center of a ball that circumscribes t and does not contain any point of S inside. Following [BO05], we call such a ball a *surface Delaunay ball* of t. In our refinement we continuously insert the centers of some surface Delaunay balls, namely the largest one for a triangle.

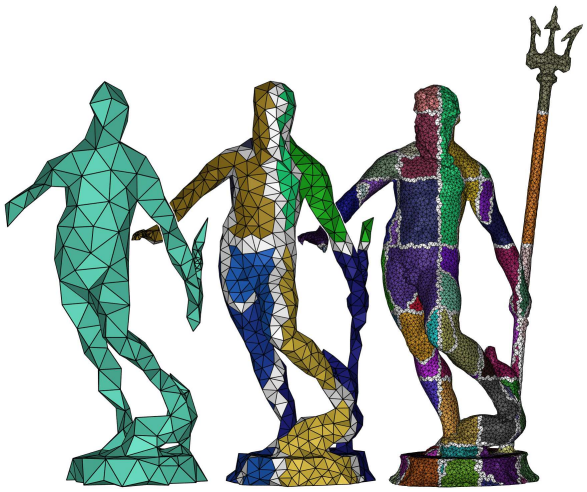

Figure 2: *Varying* λ *on* NEPTUNE*: Both geometry and topology get progressively captured with decreasing* λ *though all three meshes are guaranteed to be manifold.*

2. Algorithm

2.1. Overview

Delaunay refinement algorithms for surfaces [BO05, CDRR07] work on the following generic strategy. The algorithm maintains the restricted Delaunay triangulation $\mathrm{Del}\, S|_M$ for a set S of points sampled from the surface M. It checks certain conditions such as, if the restricted triangles around a point make a topological disk [CDRR07], or if the surface Delaunay balls of the triangles are small enough with respect to an estimate of the local feature size [BO05]. If these conditions are violated, a point is inserted into S and the appropriate restricted Delaunay triangulation is updated accordingly. The inserted point is usually chosen as the locally furthest point where a Voronoi edge meets the surface. One shows that this iterative insertion cannot go on forever and at the termination the restricted Delaunay triangulation tessellates the surface M with some desirable properties.

In our case, we do not process the entire current point set S as a single entity to avoid building the Delaunay triangulation of the entire set. Instead, we split S using an octtree data structure and process the set of points in each node individually. Since we do not access the entire sample S while processing a node, we cannot use the same point insertion strategy. For then, a locally furthest Voronoi point computed from $\mathrm{Vor}\, S'$ where $S' \subset S$ may be too close to a point in S. This would hinder maintaining a lower bound on inter-point distances, a crucial ingredient for guaranteeing termination. Also, there is another difficulty. Even if we guarantee that the output mesh for each node is a valid manifold mesh, it is not certain that these partial meshes over all nodes stitch consistently to provide a valid global mesh. This issue is not present in the refinement that uses a single global mesh since there is nothing to stitch.

We refine the local mesh for each node using similar conditions as in the original Delaunay refinement for surface meshing. It includes checking a topological disk condition as introduced in [CDRR07] and also a size condition of triangles as introduced in [BO05]. However, we do not necessarily insert the locally furthest Voronoi point. Instead, we may include some other point from the global set S into the partial set S' that we are currently dealing with. We show that such a strategy necessarily terminates while ensuring a manifold condition at each point. To take care of the global consistency of the local meshes we *reprocess* some of the nodes that are in the *vicinity* of the inserted points. When we start processing a node, we also augment its point set by including points from the nodes in its *vicinity*. It turns out that if these two vicinities are consistently chosen with respect to λ which also guides the size of the triangles, we obtain the mesh consistency.

It is important to notice that we do not keep the individual Delaunay triangulation associated with a node when it is not being processed. We only keep the sample points and the current mesh generated inside a node and rebuild the Delaunay triangulation when it is processed again. This is done to save the memory since otherwise individual Delaunay triangulations together may use memory space close to or more than the global Delaunay triangulation which we aim to avoid. Of course, this incurs some time overhead, but the gain in memory consumption results in avoiding memory thrashes which eventually trumps over this time overhead when the output mesh gets large. The balance between time overhead and economic use of memory is obtained by choosing the parameter κ appropriately which can be set a priori for a given platform.

2.2. Initialization

We assume that the input surface M is a compact 2-manifold without boundary. For the sake of theoretical proofs, we assume it to be smooth. However, one may also consider M to be "almost smooth" meaning that it is a piecewise smooth surface where input dihedral angles are close to π. Almost all ingredients that are used to prove theoretical guarantees for smooth surfaces remain valid for almost smooth surfaces [BO06, DLR05]. This is why the algorithms designed for smooth surfaces also work for almost smooth surfaces [BO06, DLR05].

We initialize the octtree with a bounding box of M. The

point set S is initialized with three very close points in M whose dual Voronoi edge intersects M. This can be done using a ray probing technique described in [BO05]. The assumption is that this Voronoi edge continues to intersect the surface throughout the algorithm meaning that the dual restricted triangle *persists* throughout meshing. Although theoretically the three points need to be close compared to local feature size, in practice, they can be chosen without estimating local feature sizes as explained in [BO05].

2.3. Node processing

A node in the octtree is subsequently split into smaller boxes generating other nodes. The points in the current sample S are divided by the boxes representing the leaf nodes in the octtree. We maintain the leaf nodes that need to be processed in a queue denoted as Q.

A node ν is processed with two actions, *split* or *refine*. A split of ν occurs when the number of points $S_\nu = S \cap \nu$ in it becomes larger than a threshold κ, that is, $|S_\nu| > \kappa$. In a split the node ν is divided into eight children which represent the division of ν into eight equal boxes. Each child acquires the subset of S_ν that belongs to it and gets enqueued in Q.

In a refine, we collect a subset of points $N_\nu \subseteq S$ that are outside ν but are within a distance of 2λ from its boundary. Here, λ is the scale parameter input by user. The choice of the factor 2 comes from our proof. Let $R_\nu = N_\nu \cup S_\nu$ denote the augmented set of points for ν. We compute a mesh comprised of the triangles in the restricted Delaunay triangulation $\mathrm{Del}\, R_\nu|_M$. Then, we refine this mesh as long as the triangles incident to each vertex in ν are large compared to the input parameter λ, or do not form a topological disk. During the refinement if the number of points in S_ν grows to κ, we invoke a split of ν. Also, during this process, other nodes can be enqueued as we explain below.

2.4. Localized refinement

Each point $p \in S$ in a node ν defines its *surface star* F_p as a subcomplex in the local Delaunay triangulation $\mathrm{Del}\, R_\nu$. It consists of all restricted triangles in $\mathrm{Del}\, R_\nu|_M$ incident to p and their sub-simplices. Formally,

$$F_p = \{\sigma \,|\, \sigma \in \mathrm{Del}\, R_\nu|_M \text{ is either a triangle incident to } p \text{ or a sub-simplex of such a triangle}\}.$$

Checking violations. There are two conditions that we check for the points in a node ν. If these conditions are violated we return a pair (p,q) where q is a candidate for insertion and p is a point in R_ν closest to q. These pairs are used later to decide which point is to be inserted into R_ν. Before checking the two conditions, we need to make sure that at least one Voronoi edge in $\mathrm{Vor}\, R_\nu$ intersects M. If not, we augment R_ν with the three nearby points that we inserted during initialization. In practice, this step is rarely executed and can be skipped.

Next, we check if there is any triangle t in $\mathrm{Del}\, R_\nu|_M$ incident to a point in $p \in S_\nu$ which has a surface Delaunay ball of radius more than λ. This can be determined by checking if the dual Voronoi edge V_t intersects M in a point p^* that is more than λ distance away from p. Such a triangle violates the condition that all triangles should be "small" compared to λ. In this case we return the pair (p, p^*). When all triangles have 'small' surface Delaunay balls, we check if F_p for each point $p \in S_\nu$ is a topological disk with each edge incident to p being adjacent to exactly two triangles in F_p. This means that we require the underlying space $|F_p|$ to be a topological disk with p being in the interior. If F_p is not a topological disk, we return the pair (p, p^*) where p^* is the center of the largest surface Delaunay ball among all such balls of triangles in F_p. Notice that in both violations p is the nearest point to p^* in R_ν since p^* belongs to the Voronoi cell of p in $\mathrm{Vor}\, R_\nu$. In both violations we may enqueue some nodes for reprocessing. If none of the violations occurs, we return a *null* pair.

Algorithm 1 VIOLATION(M, ν, λ)

```
1: If there is no triangle in Del R_ν|_M, include three vertices
   of the persistent triangle into R_ν
2: Find a triangle pqr ∈ Del R_ν|_M so that p is in ν and V_pqr
   intersects M in a point p* where d(p, p*) > λ
3: if found then
4:     return the pair (p, p*)
5: else
6:     Find a p in ν so that F_p is not a disk
7:     if found then
8:         p* := argmax_{x∈M∩V_t | t∈F_p} {d(x, p)}
9:         return (p, p*)
10:    end if
11: end if
12: return null
```

Inserting points into R_ν. Let (p, p^*) denote the pair returned by VIOLATION. So, p^* is a possible candidate for insertion. Although p^* is locally furthest in R_ν, it may not be so in S. We check if the nearest point $s \in S$ to p^* is within $\lambda/8$ distance. If so and $s \neq p$, we throw away p^* and insert s into R_ν. Otherwise, we insert p^* into R_ν. Notice that, in the first case, if s is inserted, it cannot already be in R_ν because p is the closest point to p^* in R_ν and we explicitly check if $s \neq p$. Therefore, addition of s indeed updates R_ν. In the second case, p^* is a new point and it not only enlarges R_ν but also S. Figure 3 shows some points both inside and outside of a node for STRINGY that are inserted during its processing.

Reprocessing nodes. When we insert a new point $s \notin S$, we may enqueue certain nodes for reprocessing. This is a key

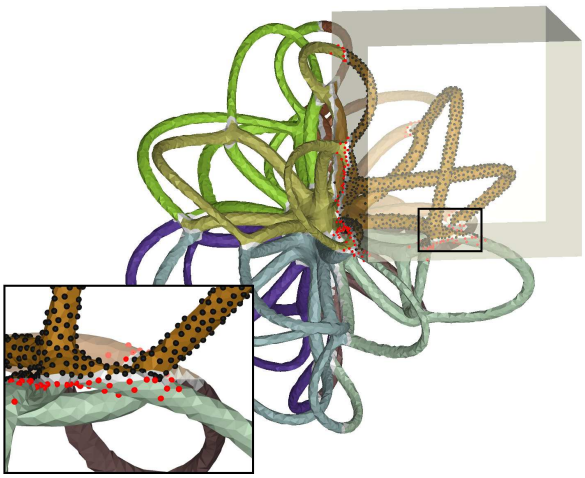

Figure 3: *Points inserted during processing a node on* STRINGY. *Black points are inserted within the node and red points are inserted outside the node.*

Algorithm 2 INSERTPOINT(ν, p, p^*, λ)

1: $s := \text{argmin}_{u \in S} d(p^*, u)$
2: **if** $d(p^*, s) \leq \lambda/8$ and $s \neq p$ **then**
3: return s else return p^*.
4: **end if**

feature of our algorithm which, as we argue later, ensures global consistency among meshes. We enqueue each node $\nu' \neq \nu$ so that the new point s lies within 2λ distance of ν'. Again, the factor 2 is derived from our proof. Figure 4 shows some inconsistencies that exist at some point of the algorithm and are ultimately removed by reprocessing nodes.

Algorithm 3 NODEENQUEUE(ν, s, λ)

1: Compute $W := \{\nu' \neq \nu | d(s, \nu') \leq 2\lambda\}$
2: **for** each $\nu' \in W$ **do**
3: enqueue(ν', Q)
4: **end for**

Output mesh. When a node ν is not being processed, we keep associated with it the subset $S_\nu = S \cap \nu$ of sample S and also the stars F_p with each point $p \in S_\nu$. When we finish processing all nodes, we output a complex formed by the union of all surface stars over all points in S, that is, $\cup_{p \in S} F_p$. We prove that $\cup_p F_p$ indeed forms a manifold mesh. The main algorithm LOCDEL is described in Algorithm 4.

3. Termination

First we observe that if LOCDEL inserts only finitely many points, it terminates. The algorithm either splits, refines, or

Algorithm 4 LOCDEL(M, κ, λ)

1: Initialize S as described;
2: Compute a bounding box of M and enqueue it to Q;
3: **while** Q is not empty **do**
4: ν :=dequeue(Q);
5: **while** (p, p^*) :=VIOLATION(M, ν, λ) is not null **do**
6: $s :=$ INSERTPOINT(ν, p, p^*, λ)
7: $R_\nu := R_\nu \cup \{s\}$;
8: **if** $s \notin S$ **then**
9: $S := S \cup \{s\}$;
10: NODEENQUEUE(ν, s, λ)
11: **end if**
12: **if** $|S_\nu| \geq \kappa$ **then**
13: Split ν and enqueue its eight children to Q
14: **end if**
15: **end while**
16: **end while**
17: Return S and $\cup_p F_p$.

enqueues a node. If there are finitely many insertions, splits and enqueue operations cannot be infinite since each such operation requires a new point to be added to S. A refinement of a node ν also cannot go forever since each refinement grows R_ν by a point and there are only finitely many of them by assumption.

Now we argue that, indeed, only finitely many points are inserted. If a point s is inserted in R_ν of a node ν, it is either an existing point, or a new point. Clearly, we need to argue only for the case when s is a new point. In this case s is the furthest intersection point of a Voronoi edge with M where the Voronoi edge is dual to a triangle in the surface star F_p for some point $p \in S_\nu$. If the nearest point to s in S is not p, its distance to all existing points in S is at least $\lambda/8$. This is ensured by step 2 of INSERTPOINT. When the nearest point to s is p, we cannot claim this lower bound on its distance to other points. Then, if s is inserted because of triangle size (step 2 in VIOLATION) it is at least λ distance away from p and all other points trivially, or if s is inserted because of disk condition (step 6 in VIOLATION), we appeal to the following result in [CDRR07] to claim a fixed positive lower bound.

Proposition 1 Let S be a sample of a smooth surface M. There exists a surface dependent constant $\varepsilon_M > 0$ so that for a point $p \in S$, if all intersection points of Voronoi edges in V_p with M lie within ε_M distance of p, then restricted triangles incident to p and their sub-simplices in $\text{Del}\, S|_M$ form a topological disk.

We apply the above proposition to p whose surface star F_p is not a topological disk. By Proposition 1 the point s that we insert is at least ε_M away from p and hence from all other existing points since p is the nearest point to s in S. This completes the proof that all new points that are inserted maintain a fixed positive lower bound of $\min\{\lambda/8, \varepsilon_M\}$ on their

distances from all other existing points. A standard packing argument shows that S is finite.

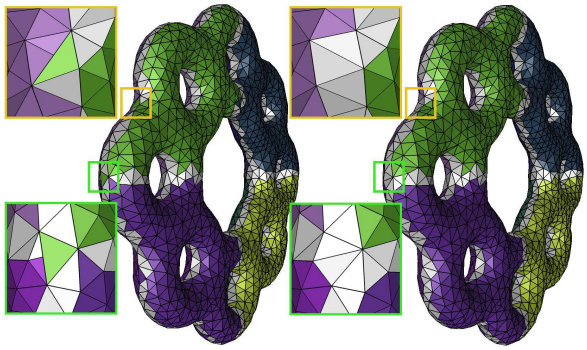

Figure 4: *Inconsistencies among meshes in different nodes (left) eventually get resolved (right) for* 9HANDLETORUS.

4. Guarantees

Our goal is to establish that LOCDEL produces a valid mesh all the time. In particular, we claim that the output is a 2-manifold for all positive values of λ. It is also geometrically close to the surface M relative to λ. However, if λ is sufficiently small, the output mesh captures the complete topology of M. Formally, the two guarantees are:

Theorem 1 The output mesh of LOCDEL(M,κ,λ) satisfies the following two properties:

(i) The underlying space of the output mesh is a 2-manifold without boundary and each point in the output is at a distance λ from M.
(ii) There exists a $\lambda^* > 0$ so that if $\lambda < \lambda^*$, the output mesh becomes isotopic to M with a Hausdorff distance $O(\lambda^2)$.

Proof Since LOCDEL terminates, we are guaranteed by VIOLATION that the surface star of each point in its local mesh is a topological disk. However, for (i) we require that all surface stars across all points globally fit together. This means that the following consistency condition should hold:

Consistency: In the output complex $\cup_p F_p$, a triangle abc is in F_a if and only if it appears in F_b and F_c.

We first show the following which leads to consistency.

Claim 1 Let S be the output sample and abc be any triangle in the output complex $\cup_p F_p$. The triangle abc belongs to the global restricted Delaunay triangulation $\mathrm{Del}\,S|_M$.

The above claim implies that each triangle in the output mesh is a restricted Delaunay triangle in the global restricted Delaunay triangulation $\mathrm{Del}\,S|_M$.
To prove the claim, assume without loss of generality that $abc \in F_c$. Consider the last time the node ν containing the point c is processed. The triangle abc belongs to $\mathrm{Del}\,R_\nu|_M$ when ν is finished. Also, its surface Delaunay balls have radius at most λ. Consider the global point set S when ν is finished. We claim that at this stage abc is also a restricted triangle in $\mathrm{Del}\,S|_M$. To see this assume to the contrary that $abc \notin \mathrm{Del}\,S|_M$. It follows that there is a point $s \in S$ which lies inside a surface Delaunay ball B of abc. Since B has a radius at most λ, the point s is within 2λ distance of c. Then, s is also in R_ν by definition since c is in ν. We reach a contradiction since then abc cannot be in $\mathrm{Del}\,R_\nu|_M$ as required. Therefore, the triangle abc is in the global restricted Delaunay triangulation when ν is finished.
Now we show that abc remains a restricted Delaunay triangle in $\mathrm{Del}\,S|_M$ afterward. Suppose not. Then, there is a new point $s \in S$ added after ν is finished which lies in the surface Delaunay ball B of abc. Then, s is within 2λ distance of c and hence within 2λ distance of ν. This would require that ν is enqueued again in NODEENQUEUE. We reach a contradiction since we already considered the last time ν is processed.

Claim 2 A triangle $abc \in \cup_p F_p$ satisfies the consistency condition.

To prove the claim by contradiction, assume without loss of generality that abc is in F_a but not in F_c. Consider the last time the node ν containing c is processed. At that point, if a is not present in R_ν, we must have the case that a is added into S after ν is finished. For otherwise a has to be in R_ν since $d(a,c) \leq 2\lambda$ as abc has a surface Delaunay ball of size at most λ. It is impossible that a is added after ν is finished since insertion of a should cause ν to be reprocessed as $d(a,c) \leq 2\lambda$. Therefore, a is in R_ν when ν is processed for the last time. The same argument applies to b as well. It follows that all vertices of abc are present when ν is processed for the last time. The triangle abc is a restricted Delaunay triangle in $\mathrm{Del}\,S|_M$ by claim 1 and hence also in $\mathrm{Del}\,R_\nu|_M$ as $R_\nu \subseteq S$. It follows that abc is in F_c by definition reaching a contradiction that F_c does not contain abc.
We complete the proof of (i) by observing that the star of each vertex in the output complex $\cup_p F_p$ is a topological disk with each edge incident to p being adjacent to exactly two triangles. Such a complex is a triangulation of a 2-manifold without boundary by a standard result in PL topology. The underlying space of $\cup_p F_p$ is also 2-manifold since it is embedded in $\mathbb{R}^3$ as a subcomplex of $\mathrm{Del}\,S$. The remaining claim in (i) that each point in the output is within λ distance of a point in M follows from the fact that each output restricted triangle is included in a surface Delaunay ball of radius at most λ.
Now we argue for the guarantee (ii). By (i), the output complex is a manifold without boundary where each triangle has a surface Delaunay ball of radius at most λ. If λ is sufficiently small, the results in [ACDL02] imply that such a mesh is homeomorphic to the surface M. Then the claims of isotopy and Hausdorff distance in (ii) follow from results in [BO05]. □

λ	κ	#tri	mem(MB)	time(h:m)
0.0029	2M	2.33M	2221	7:30
0.0029	1M	2.33M	2099	1:45
0.0029	500K	2.34M	466	2:20
0.0029	250K	2.33M	421	2:30
0.0029	100K	2.33M	453	3:00
0.0029	50K	2.33M	375	3:40
0.0029	25K	2.33M	327	3:20
0.0029	10K	2.33M	414	4:40
0.0027	2M	2.69M	2625	23:55
0.0027	1M	2.69M	2101	2:20
0.0027	500K	2.70M	512	2:45
0.0027	250K	2.69M	466	3:00
0.0027	100K	2.69M	486	3:55
0.0027	50K	2.69M	406	4:25
0.0027	25K	2.69M	440	3:55
0.0027	10K	2.69M	451	5:05
0.002	2M	Abort*	Abort*	Abort*
0.002	1M	4.91M	2098	6:30
0.002	500K	4.91M	861	7:15
0.002	250K	4.90M	1006	8:50
0.002	100K	4.91M	760	12:00
0.002	50K	4.91M	671	12:20
0.002	25K	4.91M	886	11:35
0.002	10K	4.91M	900	12:30

Table 1: *Effects of varying* κ *on* 3HOLES *(M: million, K:thousand).*

5. Experiments and results

We have implemented LOCDEL using the Delaunay triangulation of CGAL 3.2 [cga]. A number of experiments were conducted on a PC with 2.0GB of 667MHz RAM, 1.5GB swap space, and a 2.8GHz processor running with Ubuntu 9.04. The parameter λ is chosen as a factor of the largest dimension of the bounding box of M. In all tables we show λ as this factor.

First we discuss how we can tune the parameter κ and then comment on the scalability of LOCDEL. Observe that the single node cases coincide with the standard Delaunay refinements. In some of these cases the experiment is aborted by the operating system due to insufficient memory. These are indicated as Abort* in the tables. Figures 1-7 show the results on different models.

5.1. Tuning κ

Clearly, the number of nodes depends on κ. Smaller values of κ produce larger number of nodes. Consequently, the overhead for processing nodes increases. On the other hand large κ increases the number of points per node requiring more memory space. As a result when κ reaches a certain value, the memory starts thrashing. This suggests that there should be a value of κ for which LOCDEL performs optimally. Of course, to find a value near this optimal point, one has to run the code multiple times for multiple values of κ which effectively wipes out the advantage of an optimal performance.

Since κ regulates the memory usage, its optimal value should mostly depend on the specific platform on which the code is executed. This includes the memory capacity of the machine and its management by the operating system and the particular Delaunay triangulation code that is being employed. In other words, the optimal value should depend little on the particular model being meshed or the particular λ being used. In that case, for a fixed platform, we should be able to hone in on a value of κ near the optimal by running LOCDEL on some initial model with multiple values of κ and then use that value for any other model later.

Our experiments confirm the above hypothesis. Table 1 shows the test results on the model 3HOLES. For three different values of λ, we find that $\kappa = 1$ million provides the best result among eight other values. It means that among all tested values, $\kappa = 1$ million generates the largest Delaunay triangulation that still fits into main memory avoiding continuous memory swaps. Therefore, we determine that 1 million points per node is close to the optimal value for the platform on which we ran the code. Notice that, different memory capacity or different Delaunay code would probably determine a different value of κ, but, once determined, it can be kept fixed for all models irrespective of the mesh size.

Table 2 shows the performance of LOCDEL on a number of models. All output meshes are large containing around 2.5 million triangles. We observe that $\kappa = 1$ million indeed provides the best result among different κ values that we tested confirming our hypothesis. Also, notice that the qualitative property of the output mesh remains almost the same for different values of κ. Figure 6 exemplifies this aspect.

5.2. Scaling

Our experimental results show that LOCDEL scales much better than the standard Delaunay refinement which corresponds to the single node case.

Single vs. multi-nodes. In our tested examples in Table 2, we get the single node case when $\kappa = 2$ million. We observe that we obtain almost 6-10 times speed-up in the computation time on this particular platform when we use the tested optimal value of 1 million for κ compared to the single node case. Table 3 shows some cases where the program with $\kappa = 2$ million had to be aborted since it does not terminate whereas it outputs a mesh in couple of hours when $\kappa = 1$ million.

Varying λ. We have already indicated that λ regulates the the size of the output mesh. Figure 2 shows how the geom-

model	λ	κ	#leaf nodes	#tri	mem (MB)	time (hr:min)
9HANDLETORUS	0.008	2M	1	2.58M	2680	7:25
	0.008	1M	8	2.59M	2176	2:55
	0.008	25K	232	2.59M	477	5:20
OCTAHANDLES	0.001	2M	1	2.43M	2642	28:15
	0.001	1M	8	2.43M	2314	2:25
	0.001	25K	176	2.43M	426	3:50
BRACELET3	0.0033	2M	1	2.73M	2758	22:30
	0.0033	1M	8	2.73M	2170	2:30
	0.0033	25K	288	2.73M	352	4:50
HOLEDRING	0.0015	2M	1	2.66M	2805	18:45
	0.0015	1M	8	2.67M	2253	2:30
	0.0015	25K	232	2.67M	479	4:50
3HOLES	0.0027	2M	1	2.69M	2625	23:55
	0.0027	1M	8	2.69M	2101	2:20
	0.0027	25K	176	2.69M	440	3:55
HOMER	0.0022	2M	1	2.45M	2558	23:00
	0.0022	1M	8	2.46M	2222	2:30
	0.0022	25K	204	2.46M	450	4:10
NEPTUNE	0.0015	2M	1	2.30M	2667	14:20
	0.0015	1M	8	2.30M	2484	3:10
	0.0015	25K	197	2.30M	476	4:50
LION	0.0022	2M	1	2.38M	2769	26:05
	0.0022	1M	8	2.38M	2546	2:20
	0.0022	25K	218	2.38M	765	4:35
DAVID	0.0029	2M	1	2.33M	2479	13:00
	0.0029	1M	8	2.33M	2284	2:30
	0.0029	25K	218	2.33M	472	5:20
BUDDHA	0.0015	2M	Abort*	Abort*	Abort*	Abort*
	0.0015	1M	8	3.29M	2284	3:30
	0.0015	25K	281	3.29M	602	6:55

Table 2: *Time and memory usage for different models for single- and multi-node mesh generation with* LOCDEL.

Model	λ	κ		#tri		mem(MB)		time	
BUDDHA	0.0015	2M	1M	Abort*	3.29M	Abort*	2284	Abort*	3:30
3HOLES	0.0025	2M	1M	Abort*	3.14M	Abort*	2104	Abort*	3:05
3HOLES	0.002	2M	1M	Abort*	4.91M	Abort*	2098	Abort*	6:30
BIMBA	0.001	2M	1M	Abort*	5.48M	Abort*	2351	Abort*	7:50
LUCY	0.0014	2M	1M	Abort*	6.05M	Abort*	2243	Abort*	10:15
9HANDLETORUS	0.005m	2M	1M	Abort*	6.64M	Abort*	2179	Abort*	20:00

Table 3: *Cases with $\kappa = 2M$ have to be aborted whereas the cases with $\kappa = 1M$ run in few hours.*

etry and topology of an input surface are progressively captured by decreasing λ. In Figure 5 we plot the time versus the output mesh size. Clearly, it shows that LOCDEL with the tuned value of $\kappa = 1$ million scales much better than the standard Delaunay refinement which closely corresponds to the case when $\kappa = 2$ million.

6. Conclusions

In this paper we showed how one can adapt the Delaunay refinement technique for surface meshing with only local Delaunay triangulations. By tuning a parameter κ in the algorithm, one can produce large meshes of a model that would not be possible otherwise with a single global Delaunay triangulation. The parameter κ is mostly platform dependent including the memory availability and the particular Delaunay code being employed. Once it is estimated with some initial experiments, one can use the same κ for all future

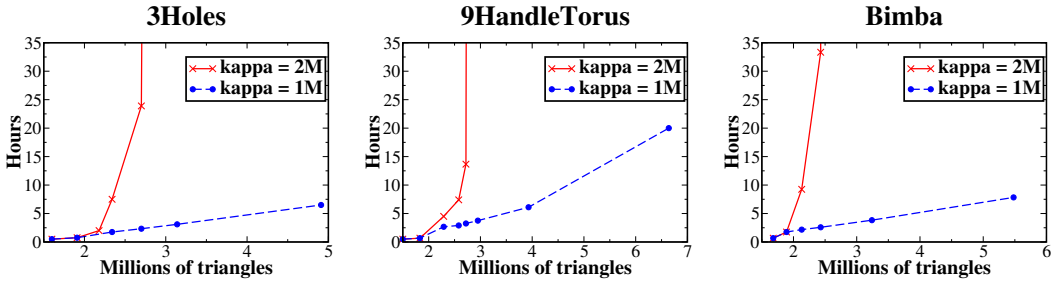

Figure 5: *Time Vs. mesh size plot for some models.*

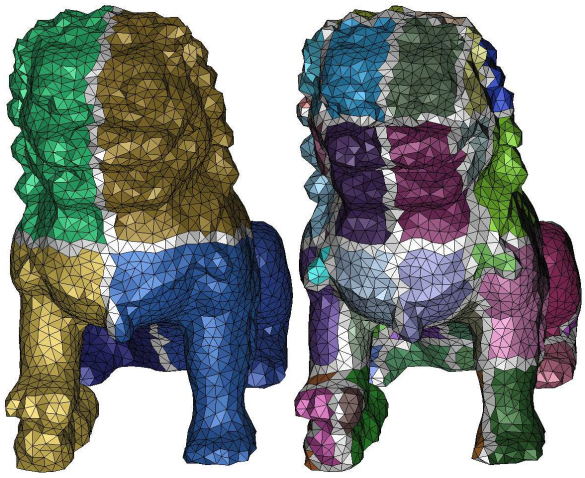

Figure 6: *Varying* κ *on* LION *does not change the mesh qualitatively.*

experiments on the same platform thereby eliminating the need for tuning κ. The other parameter λ provides the user the flexibility to choose the scale at which the surface is to be meshed. The output is guaranteed to be a manifold and captures the input topology if the supplied scale is sufficiently small. It is not hard to include the check for aspect ratio to produce a guaranteed quality mesh in our framework. We avoided including it in the description of LOCDEL just to keep its essential aspects in focus. Also, it is not difficult to observe that the technique is not tied to octtrees and can be applied to other hierarchical space decompositions as well.

We can take advantage of the cluster structure of the sample points at termination for downstream applications. For example, one may transmit the augmented point set R_v node by node and the receiver may compute the individual meshes from each cluster without worrying about consistency. A mesh processing application can treat clusters in parallel again without worrying about the mesh consistency.

This work brings forth some open questions. We used λ to produce an output mesh which is almost uniform. Is it possible to compute a mesh which is adaptive to the surface features using localized Delaunay refinement? This will require an estimation of the feature size which itself would be a nontrivial task under a localized framework. However, it may be possible to let the algorithm choose adaptively the mesh density to meet the topological disk criterion.

What about extending our framework to volume meshing? Although most of the ideas generalize to tetrahedra refinements, it is not clear at the moment how to fit everything together.

Acknowledgments.

Most of the models used in this paper are taken from the AIM@SHAPE repository. We thank Kuiyu Li for assisting with generating some of the pictures. We acknowledge CGAL consortium for making the Delaunay triangulation code available for experiments. The work on this research is partially supported by NSF grants CCF-0830467 and CCF-0915996.

References

[ACDL02] AMENTA N., CHOI S., DEY T. K., LEEKHA N.: A simple algorithm for homeomorphic surface reconstruction. *International Journal of Computational Geometry and Applications 12* (2002), 125–141. 6

[ACR03] AMENTA N., CHOI S., ROTE G.: Incremental constructions con BRIO. In *Proceedings of the 19th Annual Symposium on Computational Geometry* (2003), pp. 211–219. 1

[ACSYD05] ALLIEZ P., COHEN-STEINER D., YVINEC M., DESBRUN M.: Variational tetrahedral meshing. *ACM Transactions on Graphics 24*, 3 (July 2005), 617–625. 1

[BO05] BOISSONNAT J.-D., OUDOT S.: Provably good surface sampling and meshing of surfaces. *Graphical Models 67* (2005), 405–451. 1, 2, 3, 4, 6

[BO06] BOISSONNAT J.-D., OUDOT S.: Provably good sampling and meshing of Lipschitz surfaces. In *Proceedings of the 22nd Annual Symposium on Computational Geometry* (2006), pp. 337–346. 1, 3

[CDES01] CHENG H.-L., DEY T. K., EDELSBRUNNER H., SULLIVAN J.: Dynamic skin triangulation. *Discrete and Computational Geometry 25* (2001), 525–568. 2

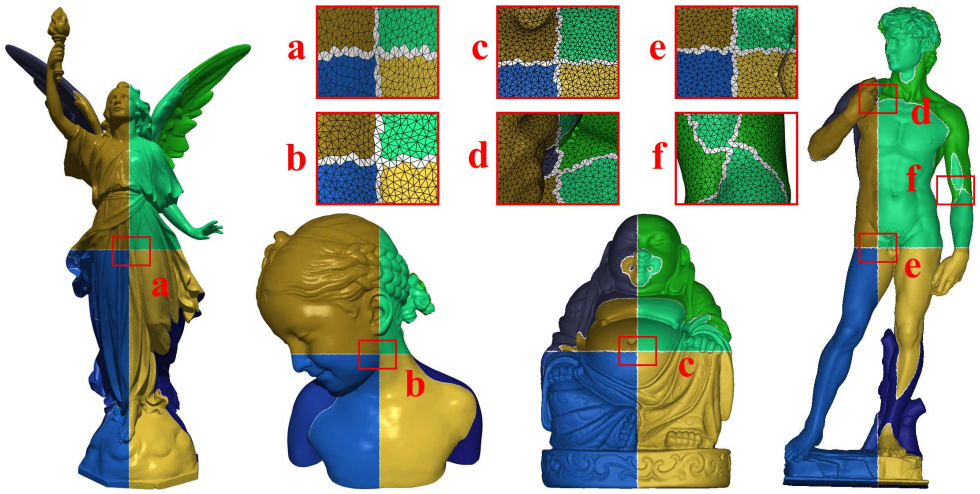

Figure 7: *Large models with consistent meshes.* LUCY, BIMBA, BUDDHA, *and* DAVID *with 6.05M, 5.48M, 3.29M, and 2.33M triangles respectively.*

[CDR08] CHENG S.-W., DEY T. K., RAMOS E. A.: Delaunay refinement for piecewise smooth complexes. *Discrete and Computational Geometry* (2008). 1

[CDRR07] CHENG S.-W., DEY T., RAMOS E., RAY T.: Sampling and meshing a surface with guaranteed topology and geometry. *SIAM Journal on Computing 37* (2007), 1199–1227. 1, 2, 3, 5

[cga] http://www.cgal.org. 7

[Che89] CHEW L. P.: *Guaranteed-quality triangular meshes*. Tech. Rep. Report TR-98-983, Department of Computer Science, Cornell University, Ithaca, New York, 1989. 1

[Dev02] DEVILLERS O.: The Delaunay hierarchy. *Internat. J. Found. Comput. Sci. 13* (2002), 163–180. 1

[DL09] DEY T. K., LEVINE J. A.: Delaunay meshing of piecewise smooth complexes without expensive predicates. *Algorithms 2* (2009), 1327–1349. 2

[DLR05] DEY T. K., LI G., RAY T.: Polygonal surface remeshing with Delaunay refinement. In *Proceedings of the 14th International Meshing Roundtable* (2005), pp. 343–361. 1, 3

[HMP06] HUDSON B., MILLER G., PHILLIPS T.: Sparse voronoi refinement. In *Proceedings of the 15th International Meshing Roundtable* (2006), pp. 339–356. 1

[ILSS06] ISENBURG M., LIU Y., SHEWCHUK J., SNOEYINK J.: Streaming computation of Delaunay triangulations. *ACM Trans. Graphics 25*, 3 (2006), 1049–1056. 1

[NCC04] NAVE D., CHRISOCHOIDES N., CHEW L.: Guaranteed-quality parallel Delaunay refinement for restricted polyhedral domains. *Computational Geometry: Theory and Applications 28* (2004), 191–215. 1

[ORY05] OUDOT S., RINEAU L., YVINEC M.: Meshing volumes bounded by smooth surfaces. In *Proceedings of the 14th International Meshing Roundtable* (2005), pp. 203–219. 1

[Rup95] RUPPERT J.: A Delaunay refinement algorithm for quality 2-dimensional mesh generation. *Journal of Algorithms 18* (1995), 548–585. 1

[She98] SHEWCHUK J. R.: Tetrahedral mesh generation by Delaunay refinement. In *Proceedings of the 14th Annual Symposium on Computational Geometry* (1998), pp. 86–95. 1

[TWAD09] TOURNOIS J., WORMSTER C., ALLIEZ P., DESBRUN M.: Interleaving Delaunay refinement and optimization for practical isotroic tetrahedron mesh generation. *ACM Trans. Graphics 28* (2009). 1

[YLL*09] YAN D.-M., LÈVY B., LIU Y., SUN F., WANG W.: Isotropic remeshing with fast and exact computation of restricted voronoi diagram. *Computer Graphics Forum 28* (2009), 1445–1454. 1