Three-Dimensional Delaunay Refinement for Multi-Core Processors*

Andrey N. Chernikov
Department of Computer Science
College of William and Mary
Williamsburg, VA 23185, USA
ancher@cs.wm.edu

Nikos P. Chrisochoides
Department of Computer Science
College of William and Mary
Williamsburg, VA 23185, USA
nikos@cs.wm.edu

ABSTRACT

We develop the first ever fully functional three-dimensional guaranteed quality parallel graded Delaunay mesh generator. First, we prove a criterion and a sufficient condition of Delaunay-independence of Steiner points in three dimensions. Based on these results, we decompose the iteration space of the sequential Delaunay refinement algorithm by selecting independent subsets from the set of the candidate Steiner points without resorting to rollbacks. We use an octree which overlaps the mesh for a coarse-grained decomposition of the set of candidate Steiner points based on their location. We partition the worklist containing poor quality tetrahedra into independent lists associated with specific separated leaves of the octree. Finally, we describe an example parallel implementation using a publicly available state-of-the art sequential Delaunay library (`Tetgen`). This work provides a case study for the design of abstractions and parallel frameworks for the use of complex labor intensive sequential codes on multicore architectures.

Categories and Subject Descriptors

D.1.3 [**Software**]: Programming Techniques—*Concurrent Programming*; F.2.2 [**Theory of Computation**]: Analysis of Algorithms and Problem Complexity—*Nonnumerical Algorithms and Problems*; I.3.5 [**Computing Methodologies**]: Computer Graphics—*Computational Geometry and Object Modeling*

General Terms

Algorithms, Performance, Design, Theory

Keywords

Mesh generation, Delaunay triangulation, Multicore Architectures, Parallel Scientific Computing, COTS Software

*(Produces the permission block, and copyright information). For use with SIG-ALTERNATE.CLS. Supported by ACM.

ICS'08, June 7–12, 2008, Island of Kos, Aegean Sea, Greece.

1. INTRODUCTION

The possibilities to accelerate the existing sequential applications by manufacturing ever faster single-core processors have approached their physical limits [27]. Our past expectations of automatic performance gains in sequential software solely from purchasing newer computers do not hold any longer. With the development of multicore processors it has become imperative to write multithreaded parallel codes. However, for many highly complex codes (like mesh generation) used in diverse application areas, the sequential software is constantly under development to accommodate the functionality requirements from the wide ranges of applications and input geometries. In mesh generation the length of the research and development cycle for industrial strength codes often takes a hundred or more man-years [1]. Therefore, the rewriting of the sequential codes and their manual parallelization is extremely expensive. In addition, due to geometric dependencies, there is no known feasible approach to the automatic compile-time analysis and parallelization [16, 24].

Delaunay refinement is a popular technique for generating triangular and tetrahedral meshes for use in the finite element method and interpolation in various numeric computing areas. Among the reasons of its popularity is the amenability of the method to rigorous mathematical analysis, which allows to derive guarantees on the quality of the elements in terms of circumradius-to-shortest edge ratio, the gradation of the mesh, and the termination of the algorithm. The problem of parallel Delaunay *triangulation* of a specified point set has been solved by Blelloch et al. [3]. Delaunay *refinement* algorithms work by inserting additional (so-called Steiner) points into an existing mesh to improve the quality of the elements. In Delaunay mesh refinement, the computation depends on the input geometry and changes as the algorithm progresses. The basic operation is the insertion of a single point which leads to the removal of a poor quality tetrahedron and of several adjacent tetrahedra from the mesh and the insertion of several new tetrahedra. The new tetrahedra may or may not be of poor quality and, hence, may or may not require further point insertions. It is proven that the algorithm eventually terminates after having eliminated all poor quality tetrahedra, and in addition the termination does not depend on the order of processing of poor quality tetrahedra, even though the structure of the final meshes may vary [25, 11].

The parallelization of Delaunay mesh refinement codes can be achieved by inserting multiple points simultaneously. If the points are "far enough," then the sets of tetrahedra

influenced by their insertion are sufficiently separated, and the points can be inserted independently. However, if the points are "close," then their insertion needs to be serialized because of possible violations of the validity of the mesh or of the Delaunay property. One way to address this problem is to introduce runtime checks [21, 16, 30] which lead to the overheads due to locking [2] and to rollbacks [21], as well as to substantial re-structuring and often re-implementation of the sequential codes. Another approach is to decompose the initial geometry [18] which has been done for two dimensions but is not easily extendable to three dimensions. The third approach which we pursue here is to use a judicious way to choose the points for insertion, so that we can guarantee their independence and thus avoid runtime data dependencies and overheads.

In this paper we analyze the dependencies between the inserted points and propose a parallel mesh refinement technique which requires neither the runtime checks nor the geometry decomposition. Using a carefully constructed octree, we split the worklist of the candidate points up into smaller lists such that the available sequential codes can be used without modifications to process the sublists. In other words, we use the application-specific knowledge to resolve the dependencies between data elements and to open the field up for automatic scheduling and optimization techniques.

For the selection of Steiner point positions there have been a number of approaches [14, 25, 10, 17]. In [9] we unified the previous approaches and defined two types of selection disks, Type I and Type II, inside the circumscribed spheres of poor quality tetrahedra. With the use of Type I selection disks a Delaunay refinement algorithm is guaranteed to terminate and with the use of Type II selection disks it also always produces well graded meshes. As one would expect, the selection disk of Type II is always inside the selection disk of Type I.

In [8] we proposed a parallel two-dimensional Delaunay refinement algorithm which constructs uniform meshes, i.e., meshes with elements of approximately the same size. However, many applications require that the meshes be graded, i.e., have smaller elements in the regions of interest and larger elements everywhere else. In [7] we developed an algorithm for the construction of graded two-dimensional meshes.

In the current paper we present a three-dimensional parallel graded Delaunay refinement algorithm. We solve the following problems:

- We resolve the dependencies among the Steiner points and decompose the iteration space of mesh refinement by selecting subsets of candidate points that are guaranteed to be independent in three-dimensional geometries.
- Our algorithm is independent of the specific positions of Steiner points, as long as they are inserted inside the selection spheres of poor quality tetrahedra. Therefore, any problem-specific (optimization-based) approach for the selection of points can be accommodated.
- Our parallel algorithm uses a sequential Delaunay refinement algorithm with few modifications as a subroutine. Our changes in the code only touch the parameters of the subroutines and the decomposition of the worklist. We believe that through a joint effort with the developers of sequential mesh generators these parts of the code can be represented as higher level abstractions which will allow to make the development of sequential software transparent to the parallel frameworks for multi-core processors.

2. DELAUNAY MESHING BACKGROUND

The input to our algorithm is domain Ω described by a Planar Linear Complex (PLC) [25]. A PLC $\mathcal{X}$ consists of a set of vertices, a set of straight line segments, and a set of planar facets. Each element of $\mathcal{X}$ is considered *constrained* and must be preserved during the construction of the mesh, although it can be subdivided into smaller elements. The vertices of $\mathcal{X}$ must be a subset of the final set of vertices in the mesh.

Let the mesh $\mathcal{M}_\mathcal{X}$ for the given PLC $\mathcal{X}$ consist of a set $V = \{p_i\}$ of vertices and a set $T = \{\tau_i\}$ of tetrahedra which connect vertices from V. We will denote the triangle with vertices p_u, p_v, and p_w as $\triangle(p_u p_v p_w)$ and the tetrahedron with vertices p_k, p_l, p_m, and p_n as $\tau(p_k p_l p_m p_n)$. We will use the symbol $e(p_i p_j)$ to represent the edge of the mesh which connects points p_i and p_j, and the symbol $\mathcal{L}(p_i p_j)$ to represent the straight line segment connecting points which are not necessarily part of the mesh.

Delaunay refinement in both two and three dimensions improves the circumradius-to-shortest edge ratio of elements. Unfortunately, in three dimensions tetrahedra with small dihedral angles called slivers can survive. Recently, an algorithm was developed for the construction of sliverless meshes by weighted Delaunay refinement [6]. However, it can offer only a very small angle guarantee made possible by the weight pumping method. Below we will focus only on the standard (unweighted) Delaunay refinement algorithm. Although slivers can impose a problem for some numerical methods, it has been shown in [20] that bounded circumradius-to-shortest edge ratio of mesh elements is sufficient to obtain optimal convergence rates for some other methods like the solution of Poisson equation using the control volume method.

Let us call the open disk corresponding to a triangle's circumscribed circle or to a tetrahedron's circumscribed sphere its *circumdisk*. We will use symbols $\bigcirc(t)$ and $r(t)$ to represent the circumdisk and the circumradius of t, respectively. A mesh is said to satisfy the *Delaunay property* if the circumdisk of every element does not contain any of the mesh vertices [13].

Traditional Delaunay mesh generation algorithms start with the construction of the initial mesh, which conforms to $\mathcal{X}$, and then refine this mesh until the element quality constraints are met. Here we focus on parallelizing the Delaunay refinement stage, which is the most memory- and computation-expensive [8]. The general idea of Delaunay refinement is to insert additional (Steiner) points inside the circumdisks of poor quality elements, which causes these elements to be destroyed, until they are gradually eliminated and replaced by better quality elements.

We will extensively use the notion of *cavity* [15] which is the set of elements in the mesh whose circumdisks include a given point p. We will denote $\mathcal{C}_\mathcal{M}(p)$ to be the cavity of p with respect to mesh $\mathcal{M}$ and $\partial\mathcal{C}_\mathcal{M}(p)$ to be the set of boundary triangles of the cavity, i.e., the triangles which belong to only one tetrahedron in $\mathcal{C}_\mathcal{M}(p)$. When $\mathcal{M}$ is clear from the context, we will omit the index. For our analysis, we will use the Bowyer-Watson (B-W) point insertion algorithm [4,

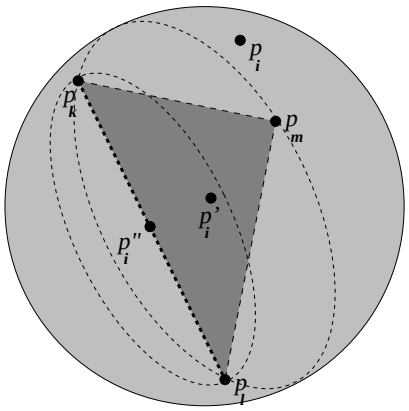

Figure 1: Encroachment in three dimensions.

29], which can be written shortly as follows:

$$\begin{array}{lcl} V^{n+1} & \leftarrow & V^n \cup \{p\}, \\ T^{n+1} & \leftarrow & T^n \setminus \mathcal{C}_{\mathcal{M}^n}(p) \cup \{(p\xi) \mid \xi \in \partial\mathcal{C}_{\mathcal{M}^n}(p)\}, \end{array} \quad (1)$$

where ξ is a three-dimensional triangle, while $\mathcal{M}^n = (V^n, T^n)$ and $\mathcal{M}^{n+1} = (V^{n+1}, T^{n+1})$ represent the mesh before and after the insertion of p, respectively. The set of newly created elements forms a *ball* [15] of point p, denoted $\mathcal{B}(p)$, which is the set of elements in the mesh that are incident upon p.

In order to avoid the creation of skinny elements close to the constrained segments and faces, sequential Delaunay algorithms observe special *encroachment* rules [25]. In particular, if a Steiner point p is considered for insertion but it lies within the open equatorial disk of a constrained subfacet f, p is not inserted but the circumcenter of f is inserted instead. Similarly, if p is inside the open diametral circle of a constrained subsegment s, then the midpoint of s is inserted instead. Consider the example in Figure 1. The new point p_i is inside the three-dimensional equatorial disk of a constrained face $\triangle(p_k p_l p_m)$. In this case, p_i is rejected and the algorithm attempts to insert the circumcenter p'_i of $\triangle(p_k p_l p_m)$. If p'_i does not encroach upon any constrained segments, it is inserted into the mesh. If, however, it encroaches upon a constrained segment, which is $e(p_k p_l)$ in our example, p'_i is also rejected and the midpoint p''_i of the constrained edge is inserted.

3. POINT DELAUNAY-INDEPENDENCE

In the context of mesh refinement by edge subdivision, Oliker and Biswas [22] introduced an independence condition via coloring. They say that "two triangles have different colors if they share an edge or a vertex." In the case of Delaunay refinement, one needs to consider more complex relations that involve cavities.

DEFINITION 1. *Points p_i and p_j are* Delaunay-independent *with respect to mesh $\mathcal{M}^n = (V^n, T^n)$ if their concurrent insertion yields the conformal Delaunay mesh $\mathcal{M}^{n+1} = (V^n \cup \{p_i, p_j\}, T^{n+1})$. Otherwise, p_i and p_j are* Delaunay-conflicting.

If a candidate Steiner point p_i encroaches upon a constrained face, let p'_i be the circumcenter of this face, and if p_i or p'_i encroach upon a constrained segment, let p''_i be the midpoint of this segment (similarly for p_j).

DEFINITION 2. *Points p_i and p_j are* strongly Delaunay-independent *with respect to mesh $\mathcal{M}^n$ iff any pair of points in $\{p_i, p'_i, p''_i\} \times \{p_j, p'_j, p''_j\}$ are Delaunay-independent with respect to $\mathcal{M}^n$.*

For practical purposes it is not enough to have the definitions of Delaunay-independent points; we need a means to verify whether two points are Delaunay-independent without actually inserting them into the mesh. Below we prove a necessary and sufficient condition (criterion).

3.1 Delaunay-Independence Criterion

THEOREM 1. *Points p_i and p_j are Delaunay-independent with respect to mesh $\mathcal{M}^n$ iff both* (2) *and* (3) *hold:*

$$\mathcal{C}_{\mathcal{M}^n}(p_i) \cap \mathcal{C}_{\mathcal{M}^n}(p_j) = \emptyset, \quad (2)$$

$$\forall \xi \in \partial\mathcal{C}_{\mathcal{M}^n}(p_i) \cap \partial\mathcal{C}_{\mathcal{M}^n}(p_j) \ : \ p_i \notin \bigcirc(\tau(p_j\xi)). \quad (3)$$

PROOF. First, $\mathcal{M}^{n+1} = (V^n \cup \{p_i, p_j\}, T^{n+1})$ is conformal iff (2) holds. Indeed, if (2) holds, then considering (1), the concurrent retriangulation of $\mathcal{C}_{\mathcal{M}^n}(p_i)$ and $\mathcal{C}_{\mathcal{M}^n}(p_j)$ will not yield overlapping triangles, and the mesh will be conformal. Conversely, if (2) does not hold, the newly created elements will intersect, and $\mathcal{M}^{n+1}$ will not be conformal.

Now, we will show that $\mathcal{M}^{n+1}$ is Delaunay iff (3) holds. The Delaunay Lemma [15] states that iff the empty circumdisk criterion holds for every pair of adjacent tetrahedra, then the tetrahedralization is globally Delaunay. Disregarding the symmetric cases, there are three types of pairs of adjacent tetrahedra τ_r and τ_s, where $\tau_r \in \mathcal{B}_{\mathcal{M}^{n+1}}(p_i)$, that will be affected:

(i) $\tau_s \in \mathcal{B}_{\mathcal{M}^{n+1}}(p_i)$,

(ii) $\tau_s \in T^{n+1} \setminus \mathcal{B}_{\mathcal{M}^{n+1}}(p_i) \setminus \mathcal{B}_{\mathcal{M}^{n+1}}(p_j)$, and

(iii) $\tau_s \in \mathcal{B}_{\mathcal{M}^{n+1}}(p_j)$.

The sequential Delaunay refinement algorithm guarantees that τ_r and τ_s will be locally Delaunay in the first two cases. In addition, condition (3) ensures that they will be locally Delaunay in the third case. Therefore, the mesh will be globally Delaunay. Conversely, if (3) does not hold, the tetrahedra $\tau(p_i\xi)$ and $\tau(p_j\xi)$ will not be locally Delaunay, and the mesh will not be globally Delaunay. □

COROLLARY 1. *From Theorem 1 it follows that if* (2) *holds and*

$$\partial\mathcal{C}_{\mathcal{M}^n}(p_i) \cap \partial\mathcal{C}_{\mathcal{M}^n}(p_j) = \emptyset, \quad (4)$$

then p_i and p_j are Delaunay-independent.

3.2 A Sufficient Condition

Let the *reflection of disk $\bigcirc(\triangle(p_k p_l p_m))$ about the edge $e(p_k p_l)$* be the disk $\bigcirc'_{e(p_k p_l)}(\triangle(p_k p_l p_m))$ that has the same radius, whose circle passes through points p_k and p_l, and whose center lies on the other side of edge $e(p_k p_l)$ from point p_m, see Figure 2.

In [8] we proved the following two-dimensional result:

LEMMA 1. *For any point p_i inside the region*

$$\bigcirc(\triangle(p_k p_l p_m)) \setminus \bigcirc'_{e(p_k p_l)}(\triangle(p_k p_l p_m)),$$

see Figure 2, the following inequality holds:

$$r(\triangle(p_k p_l p_i)) < r(\triangle(p_k p_l p_m)).$$

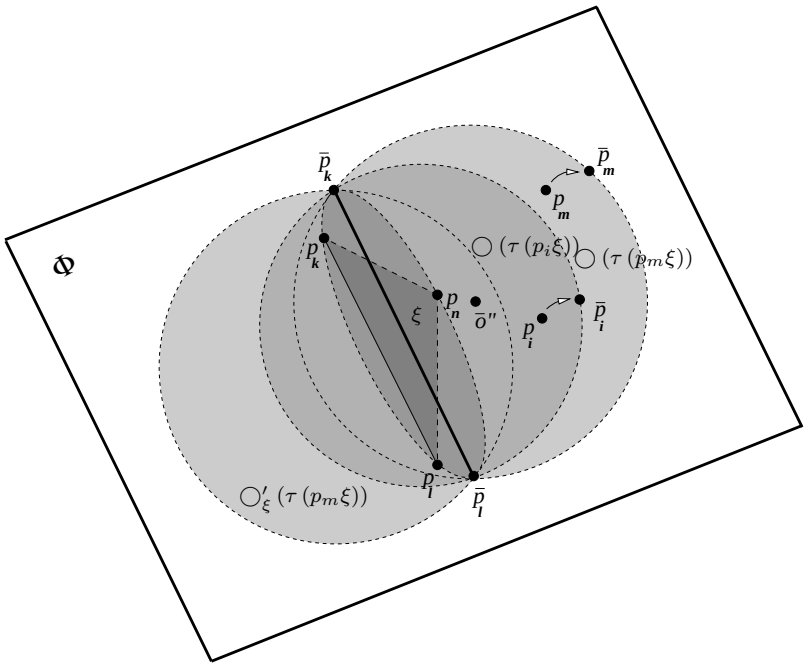

Figure 3: A three dimensional disk and its reflection cut by a plane.

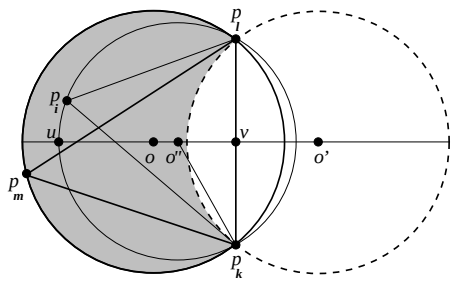

Figure 2: The solid circle corresponds to $\bigcirc(\triangle(p_k p_l p_m))$ with center in point o, and the dashed circle corresponds to $\bigcirc'_{e(p_k p_l)}(\triangle(p_k p_l p_m))$ with center in point o'. Point o'' is the center of the variable-radius disk $\bigcirc(\triangle(p_i p_k p_l))$, whose circle passes through p_k and p_l. We prove that for any point p_i inside the shaded region, $r(\triangle(p_k p_l p_i)) < r(\triangle(p_k p_l p_m))$.

Similar to the two-dimensional case, let the *reflection of disk* $\bigcirc(\tau(p_m p_k p_l p_n))$ *about the face* $\xi = \triangle(p_k p_l p_n)$ be the disk $\bigcirc'_\xi(\tau(p_m\xi))$ that has the same radius, whose sphere passes through the circle of $\bigcirc(\xi)$, and whose center lies on the other side of ξ from point p_m. The following lemma extends Lemma 1 to three dimensions.

LEMMA 2. *For any point p_i inside the region*

$$\bigcirc(\tau(p_m\xi)) \setminus \bigcirc'_\xi(\tau(p_m\xi)),$$

where $\xi = \triangle(p_k p_l p_n)$, the following inequality holds:

$$r(\tau(p_i\xi)) < r(\tau(p_m\xi)).$$

PROOF. We reduce the three-dimensional case to the two-dimensional case by the following construction (see Figure 3). Draw an arbitrary diameter $\mathcal{L}(\bar{p}_k\bar{p}_l)$ of disk $\bigcirc(\xi)$. Then let Φ be the plane which passes through $\mathcal{L}(\bar{p}_k\bar{p}_l)$ and is perpendicular to the plane containing ξ.

Let $\bigcirc(\bar{p}_k\bar{p}_l\bar{p}_i)$ be the intersection of Φ with $\bigcirc(\tau(p_i\xi))$ such that $\bar{p}_i \in \Phi$ is obtained by moving p_i along the surface of the sphere of $\bigcirc(\tau(p_i\xi))$ in the plane perpendicular to Φ.

Let $\bigcirc(\bar{p}_k\bar{p}_l\bar{p}_m)$ be the intersection of Φ with $\bigcirc(\tau(p_m\xi))$ such that $\bar{p}_m \in \Phi$ is obtained by moving p_m along the surface of the sphere of $\bigcirc(\tau(p_i\xi))$ in the plane perpendicular to Φ.

Also, let $\bar{o}$" be the center of the disk $\bigcirc(\tau(p_m\xi))$. Note that $\bar{o}$" $\in \Phi$ because $\mathcal{L}(\bar{p}_k\bar{p}_l)$ is a diameter of $\bigcirc(\xi)$.

By construction, since $\bigcirc(\bar{p}_i\bar{p}_k\bar{p}_l)$ is a two-dimensional equatorial disk of the three-dimensional disk $\bigcirc(\tau(p_i\xi))$, we have:

$$r(\tau(p_i\xi)) = r(\triangle(\bar{p}_i\bar{p}_k\bar{p}_l)), \tag{5}$$

and, since $\bigcirc(\bar{p}_m\bar{p}_k\bar{p}_l)$ is a two-dimensional equatorial disk of the three-dimensional disk $\bigcirc(\tau(p_m\xi))$, we have:

$$r(\tau(p_m\xi)) = r(\triangle(\bar{p}_m\bar{p}_k\bar{p}_l)). \tag{6}$$

Now we can see that the arrangement on the plane Φ is similar to the two-dimensional arrangement in Figure 2 with each point p in two dimensions corresponding to point $\bar{p}$ in the plane Φ. According to the two-dimensional result of Lemma 1,

$$r(\triangle(\bar{p}_i\bar{p}_k\bar{p}_l)) < r(\triangle(\bar{p}_m\bar{p}_k\bar{p}_l)). \tag{7}$$

Combining inequality (7) with equalities (5) and (6), we conclude the proof. □

LEMMA 3. *Let $\tau(p_m\xi) \in \mathcal{C}(p_i)$ and $\tau(p_r\xi) \notin \mathcal{C}(p_i)$, where $\xi = \triangle(p_k p_l p_n) \in \partial\mathcal{C}(p_i)$. Then*

$$r(\tau(p_i\xi)) < \max\{r(\tau(p_m\xi)), r(\tau(p_r\xi))\}.$$

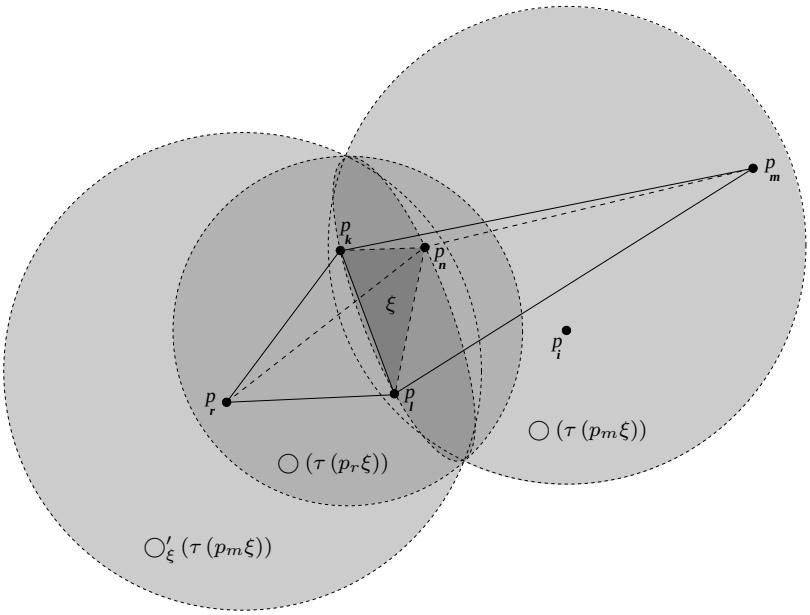

Figure 4: A face ξ at the boundary of the cavity of a Steiner point p_i, first case.

PROOF. The conditions $\tau(p_m\xi) \in \mathcal{C}(p_i)$ and $\tau(p_r\xi) \notin \mathcal{C}(p_i)$ imply that p_i lies inside the region $\Psi = \bigcirc(\tau(p_m\xi)) \setminus \bigcirc(\tau(p_r\xi))$. There are two cases:

(i) If $r(\tau(p_m\xi)) > r(\tau(p_r\xi))$, see Figure 4, then

$$\Psi \subset \left(\bigcirc(\tau(p_m\xi)) \setminus \bigcirc'_\xi(\tau(p_m\xi))\right),$$

and, according to Lemma 2, $r(\tau(p_i\xi)) < r(\tau(p_m\xi))$.

(ii) If $r(\tau(p_m\xi)) \leq r(\tau(p_r\xi))$, see Figure 5, then

$$\Psi \subset \left(\bigcirc'_\xi(\tau(p_r\xi)) \setminus \bigcirc(\tau(p_r\xi))\right)$$

and, by Lemma 2, $r(\tau(p_i\xi)) < r(\tau(p_r\xi))$.

□

THEOREM 2. *Points p_i and p_j are Delaunay-independent if there exists a subsegment s of segment $\mathcal{L}(p_ip_j)$ such that all tetrahedron circumdisks which intersect s have diameter less than or equal to the length of s, i.e.,*

$$\exists s \subseteq \mathcal{L}(p_ip_j) \;:\; \forall\tau \in T \;:\; s \cap \bigcirc(\tau) \neq \emptyset \implies 2r(\tau) \leq |s|, \tag{8}$$

where $|s|$ is the length of s.

PROOF. First, condition (8) implies that $\mathcal{C}(p_i) \cap \mathcal{C}(p_j) = \emptyset$. Indeed, if there had been a tetrahedron circumdisk which included both p_i and p_j, then the diameter of this circumdisk would be greater than the length of $\mathcal{L}(p_ip_j)$ which would contradict (8).

Now, there are two possibilities:

(i) If $\partial\mathcal{C}(p_i) \cap \partial\mathcal{C}(p_j) = \emptyset$, then, by Corollary 1, p_i and p_j are Delaunay-independent.

(ii) Otherwise, let $\partial\mathcal{C}(p_i) \cap \partial\mathcal{C}(p_j) \neq \emptyset$ and $\xi = \triangle(p_kp_lp_n)$ be an arbitrary face in $\partial\mathcal{C}(p_i) \cap \partial\mathcal{C}(p_j)$, such that $\tau(p_r\xi) \in \mathcal{C}(p_i)$ and $\tau(p_m\xi) \in \mathcal{C}(p_j)$. We are going to prove that $p_i \notin \bigcirc(\tau(p_j\xi))$ and, thus, p_i and p_j are Delaunay-independent by Theorem 1. The proof is by contradiction. Suppose condition (8) holds and $p_i \in \bigcirc(\tau(p_j\xi))$. There are two cases:

(ii-a) If $r(\tau(p_m\xi)) > r(\tau(p_r\xi))$, see Figure 6, then from Lemma 3 it follows that

$$r(\tau(p_j\xi)) < r(\tau(p_m\xi)). \tag{9}$$

In addition, the assumption that $p_i \in \bigcirc(\tau(p_j\xi))$ implies that

$$|\mathcal{L}(p_ip_j)| < 2r(\tau(p_j\xi)). \tag{10}$$

From (9) and (10) we conclude that the following inequality holds:

$$|\mathcal{L}(p_ip_j)| < 2r(\tau(p_m\xi)). \tag{11}$$

Due to (11) and the assumption that (8) holds as well as the fact that $|s| \leq |\mathcal{L}(p_ip_j)|$, we conclude that s cannot intersect $\bigcirc(\tau(p_m\xi))$. If p_t is the point of intersection of $\mathcal{L}(p_ip_j)$ with the boundary of $\bigcirc(\tau(p_m\xi))$, then s is restricted to be the subsegment of $\mathcal{L}(p_ip_t)$ and

$$|s| \leq |\mathcal{L}(p_ip_t)|. \tag{12}$$

From the assumptions that $p_i \in \bigcirc(\tau(p_j\xi))$ and $p_i \notin \bigcirc(\tau(p_m\xi))$, it follows that p_i has to lie in the region $\bigcirc(\tau(p_j\xi)) \setminus \bigcirc(\tau(p_m\xi))$ and the following two inequalities hold:

$$s \cap \bigcirc(\tau(p_r\xi)) \neq \emptyset \tag{13}$$

and

$$|\mathcal{L}(p_ip_t)| < 2r(\tau(p_r\xi)). \tag{14}$$

Inequalities (12), (13), and (14) together imply that the condition (8) does not hold and we have come to a contradiction.

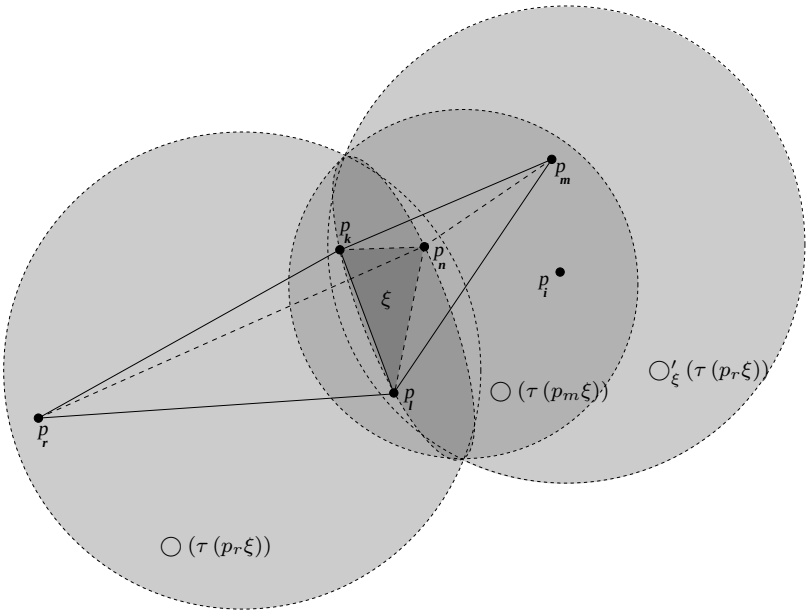

Figure 5: A face ξ at the boundary of the cavity of a Steiner point p_i, second case.

(ii-b) If $r(\tau(p_m\xi)) \leq r(\tau(p_r\xi))$, see Figure 7, then from Lemma 3 it follows that $r(\tau(p_j\xi)) < r(\tau(p_r\xi))$ and considering that

$$|s| \leq |\mathcal{L}(p_ip_j)| < 2r(\tau(p_j\xi)) < 2r(\tau(p_r\xi))$$

we conclude that s cannot intersect $\bigcirc(\tau(p_r\xi))$. This limits s to lie within the subsegment $\mathcal{L}(p_jp_t)$, where p_t is the point of intersection of $\mathcal{L}(p_ip_j)$ with the boundary of $\bigcirc(\tau(p_r\xi))$; therefore,

$$|s| \leq |\mathcal{L}(p_jp_t)|. \tag{15}$$

The subsegment $\mathcal{L}(p_jp_t)$ lies completely inside the region

$$\bigcirc(\tau(p_j\xi)) \setminus \bigcirc(\tau(p_r\xi))$$

which in turn is completely inside $\bigcirc(\tau(p_m\xi))$, hence the following two inequalities hold:

$$s \cap \bigcirc(\tau(p_m\xi)) \neq \emptyset \tag{16}$$

and

$$|\mathcal{L}(p_jp_t)| < 2r(\tau(p_m\xi)). \tag{17}$$

Inequalities (15), (16), and (17) together imply that the condition (8) does not hold and we have come to a contradiction.

□

4. OCTREE CONSTRUCTION

Theorem 2 provides a condition which allows to construct buffer zones between the regions of refinement. Indeed, if, for given two regions $R_i, R_j \subset \mathbb{R}^3$, any line segment $\mathcal{L}(p_ip_j)$ with $p_i \in R_i$ and $p_j \in R_j$ intersects some region $R_k \subset \mathbb{R}^3$ ($R_k \cap R_i = \emptyset$, $R_k \cap R_j = \emptyset$) such that $|\mathcal{L}(p_ip_j) \cap R_k| \geq w$ for some $w > 0$, and all tetrahedron circumdisks intersecting R_k have radius less than $w/2$, then condition (8) holds for p_i and p_j. Then all Steiner points in R_i can be inserted concurrently with all Steiner points in R_j. This is the idea behind the use of an octree in our algorithm. For each leaf, we keep the list of poor quality tetrahedra whose Steiner points fall inside the square of this leaf. When a leaf is scheduled for refinement (becomes active), only the corresponding Steiner points can be inserted. The carefully chosen leaves of the octree which surround the active leaf L_i work as the buffer zone. The buffer leaves cannot become active simultaneously with L_i, although the changes caused by the refinement of L_i can propagate to its buffer leaves but no further. Therefore, L_i can be refined concurrently with any other leaf in the octree which is not in the buffer zone of L_i.

Callahan and Kosaraju [5] developed a binary tree data structure for constructing well-separated pair decompositions of points, which was motivated by an application in n-body simulations. This data structure is based on a fair split tree of a point set which associates a leaf with each of the points. The construction of the octree which we describe below also uses a notion of separated regions. However, in the mesh generation context, the separation is based on the size and the shape of the tetrahedra in the underlying mesh.

De Cougny, Shephard, and Ozturan [12] use an underlying octree to aid in parallel three-dimensional mesh generation. After the generation of the octree and template meshing of the interior octants, their algorithm connects a given surface triangulation to the interior octants using face removal. The face removal procedure eliminates problems due to the small distance between the interior quadrants and boundary faces, by defining "an entity too close to the boundary triangulation" and "using the distance of about one-half the octant edge length as the minimum" [12]. We explore a somewhat similar question in the context of Delaunay refinement and derive precise distances that are necessary between the interiors and the boundaries of concurrently refined subdomains.

Löhner and Cebral [19] developed a parallel advancing front scheme. They use an octree to delimit the zones where

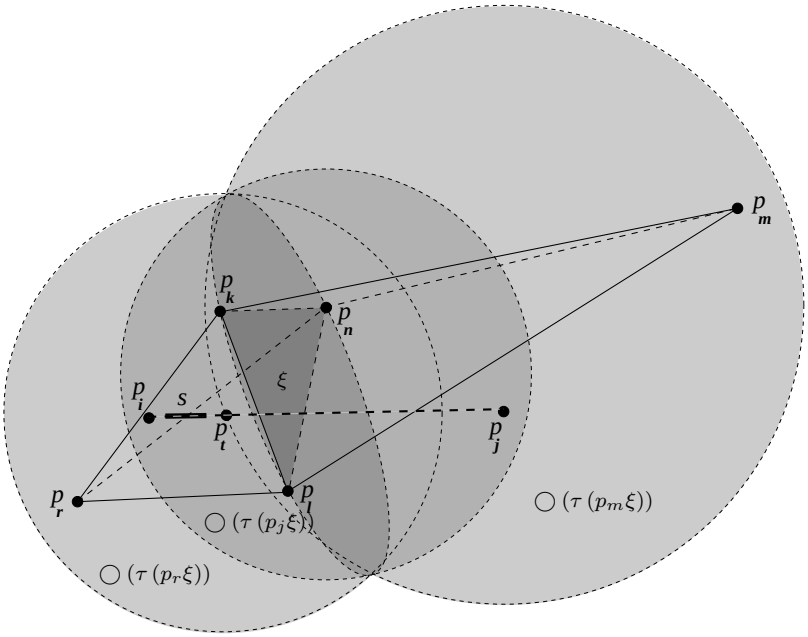

Figure 6: A face shared by the boundaries of cavities of two Steiner points, one point is inside the circumdisk created by the second point, first case.

elements can be introduced concurrently and set the edge length of the smallest octree box to be of the order of 20 to 50 times the specified element size. They implement a "shift and regrid" technique with the shift distance determined by $\min(0.5s_{min}, 2.0d_{min})$, where s_{min} is the minimum box size in which elements are to be generated, and d_{min} is the minimum element size in the active front. These distances are likely to work well in the case of advancing front meshing, when there is a clear distinction between triangulated and empty areas, however, Delaunay refinement, in addition to maintaining a mesh which at all times covers the entire domain, also requires that all triangle circumcenters be empty of mesh points.

Let $\Lambda_x = \{Left, Right\}$, $\Lambda_y = \{Top, Bottom\}$, and $\Lambda_z = \{Back, Front\}$ be the possible directions of face-adjacent leaves of an octree.

DEFINITION 3. *Let the* α-neighborhood $\mathcal{N}_\alpha(L)$ of an octree leaf L ($\alpha \in \Lambda_x \cup \Lambda_y \cup \Lambda_z$) *be the set of octree leaves that share a face with* L *and are located in the* α *direction of* L.

DEFINITION 4. *Let the set of leaves*

$$\begin{aligned}\mathrm{BUF}(L) = \; & \bigcup_{\alpha\in\Lambda_x} \mathcal{N}_\alpha(L) \cup \\ & \bigcup_{\beta\in\Lambda_y} \{\mathcal{N}_\beta(L') \mid L' \in \mathcal{N}_\alpha(L)\} \cup \\ & \bigcup_{\gamma\in\Lambda_z} \{\mathcal{N}_\gamma(L'') \mid L'' \in \{\mathcal{N}_\beta(L') \mid L' \in \mathcal{N}_\alpha(L)\}\}\end{aligned}$$

under the condition

$$\forall L' \in \mathrm{BUF}(L), \forall \tau \in T : \bigcirc(\tau) \cap L' \neq \emptyset \implies r(\tau) < \frac{1}{6}\ell(L'), \tag{18}$$

be called a buffer zone *of leaf* L *with respect to mesh* $\mathcal{M}$, *where* $\ell(L')$ *is the length of the edge of cube* L'.

DEFINITION 5. *Let two regions* $R_i \subset \mathbb{R}^3$ *and* $R_j \subset \mathbb{R}^3$ *be called* Delaunay-separated *with respect to mesh* $\mathcal{M}$ *iff any two arbitrary points* $p_i \in R_i$ *and* $p_j \in R_j$ *are strongly Delaunay-independent.*

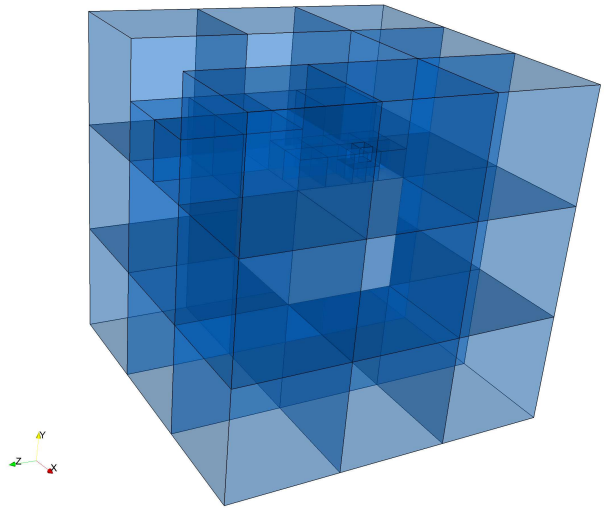

Figure 8: The buffer zone of an octree leaf L. The leaf L is the solid white box in the center. The transparent boxes around it is BUF (L).

THEOREM 3. *If* L_i *and* L_j *are octree leaves,* $i \neq j$, *and* $L_j \notin \mathrm{BUF}(L_i)$, *then* L_i *and* L_j *are Delaunay-separated.*

PROOF. (Sketch) We start by proving that, for an arbitrary pair of points $p_i \in L_i$ and $p_j \in L_j \notin \mathrm{BUF}(L_i)$, p_i and p_j are Delaunay-independent. To do that, we enumerate all possible configurations of the leaves in BUF (L_i), up to symmetry, and prove that in all cases there exists a sequence of

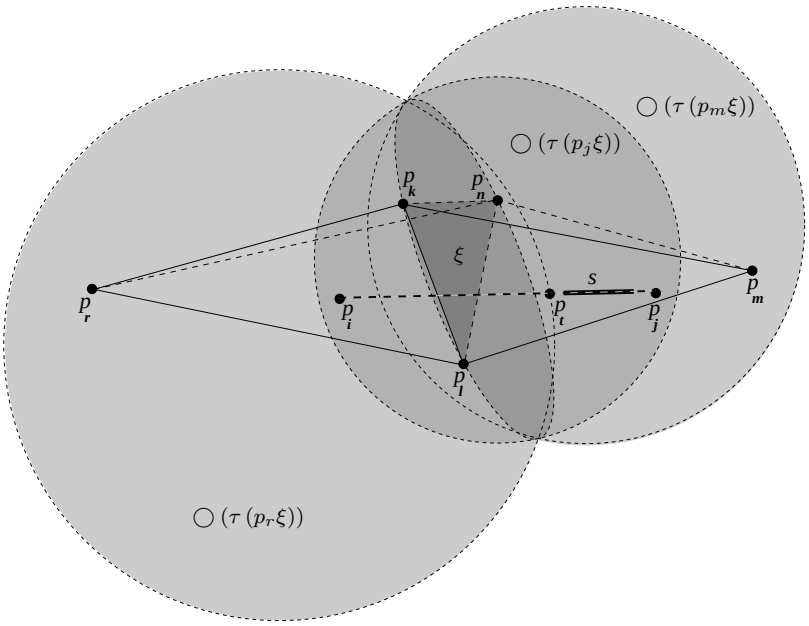

Figure 7: A face shared by the boundaries of cavities of two Steiner points, one point is inside the circumdisk created by the second point, second case.

leaves $L_{k_1}, \ldots, L_{k_b} \in \text{BUF}(L_i)$ and $w > 0$ such that

$$\left| \mathcal{L}(p_i p_j) \cap \bigcup_{m=1}^{b} L_{k_m} \right| \geq w,$$

while

$$\forall \tau \in T^n \ : \ \bigcirc(\tau) \cap \bigcup_{m=1}^{b} L_{k_m} \neq \emptyset \implies r(\tau) < \frac{1}{6} w,$$

and, thus, the condition of Theorem 2 is satisfied.

Then we extend the proof to show that any pair of points from $\{p_i, p_i', p_i''\} \times \{p_j, p_j', p_j''\}$ are Delaunay-independent since the encroachment can only lead to the insertion of a point at distance at most $2w$ from the original point. Therefore, even with double encroachment by both points, the condition of Theorem 2 can still be satisfied, and p_i and p_j are strongly Delaunay-independent; hence, L_i and L_j are Delaunay-separated. □

4.1 Implementation and Evaluation

For the implementation of our parallel Delaunay refinement algorithm we were able to utilize the serial Delaunay refinement code realized in `Tetgen` [26]. We consider this as a major accomplishment towards the goal of separating the parallel and the sequential design issues in parallel mesh generation. Indeed, `Tetgen` consists of about 33 thousand lines of highly optimized C++ code which took its author (Hang Si) several years to write. In addition, the implementation is based on a large number of theoretical and algorithmic results which were published during the last several decades and keep being introduced. Therefore, it is imperative that the development of the sequential part of the software be separated from the parallel part. The idea of separating the components of complex software systems for better maintainability is also used in the design

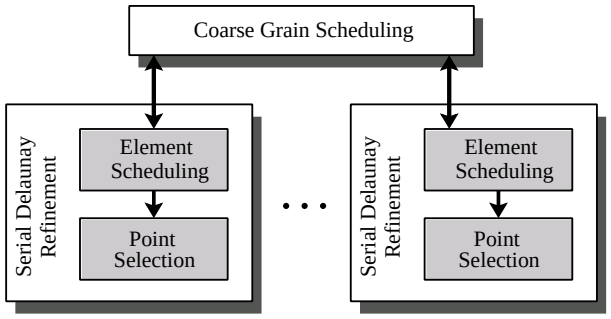

Figure 9: The diagram of the design of our parallel Delaunay refinement software.

of web-services [28]. Tightly coupled parallel mesh generation [21] has been shown to perform poorly on the grid due to intensive communication. The method presented here is partially coupled, and although the communication between the subproblems does not lead to high overheads on multicore systems, its performance remains to be evaluated in the distributed (web-service based) environments.

Figure 9 presents a high level diagram of our software design. The blocks marked "Serial Delaunay Refinement" represent P instances of sequential Delaunay refinement code which is `Tetgen` in our implementation, but could be another code as well, e.g., `Pyramid`, `Tetmesh`, or `Gridgen`. The "Element Scheduling" boxes represent the management of poor quality element queues by the sequential code. In our implementation, we split the `Tetgen` worklist to create a separate queue for each of the leaves of the octree. We schedule only one leaf at a time for refinement by a single thread, so that each thread pops from and pushes into a separate poor element queue. The "Point Selection" box is the abstraction for choosing a particular strategy for inserting

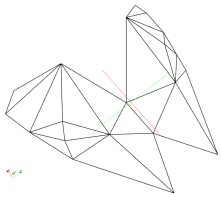

Figure 10: A wireframe model of a flying bat.

Table 2: The total time (in seconds) spent by the three-dimensional PGDR code on refining the mesh of the bat model: $\bar{r} = 0.25$ if $(-19.89 < x < 6.50) \wedge (-5.65 < y < 8.05) \wedge (-5.61 < z < 5.61)$; and $\bar{r} = 0.5$ otherwise; 5.8 million tetrahedra.

Number of available compute threads			
1	2	3	4
235.2	142.1	120.0	111.2

points inside the circumdisks of a poor quality elements. As we have shown in [9], sequential Delaunay refinement algorithms have the flexibility to choose Steiner points from the regions inside the circumdisk of a poor quality tetrahedron which we call the selection disks. The box marked "Coarse Grained Scheduling" represents the construction of the octree and the scheduling of leaves for the refinement. The leaves with larger volumes have higher refinement priorities than the leaves with smaller volumes, and the leaves of the same size are processed in the first-in-first-out order. This strategy is designed to achieve maximal concurrency as early as possible in the progress of the algorithm without introducing large overheads. The development of more efficient scheduling algorithms is the topic of our future research.

Figure 10 shows a wireframe model of a flying bat used in the simulation and visualization of air flow around bat wings [23]. The modeling is performed by constructing a large box containing the bat, see Figure 11. A tetrahedral mesh is constructed in the interior of the box, such that the face ahead of the bat is considered the inflow, the face behind is the outflow, and the other faces are paired for the use of periodic boundary conditions. The mesh is used as input to the spectral/hp element solver `Nektar` which solves the incompressible Navier-Stokes equations in arbitrary Lagrangian-Eulerian formulation. The most interesting physical phenomena like high vorticity happen in the area directly adjacent to the bat and in the trail just behind it. That is why these areas require a more refined mesh to capture their details. We defined a second box of parameterizable size inside the large box to specify the area of fine refinement, see Figure 11.

We ran our experiments on the SciClone Cluster at the College of William and Mary and used one of its "Vortex" nodes which is a quad-cpu Sun Fire V440 server with 1.28 GHz clock and 8 GB of main memory. Our implementation first constructs an octree with leaf size reflecting the local values of the grading function. Then a tetrahedral mesh is constructed and sequentially refined until inequality (18) holds. Finally, the mesh is refined in parallel until all elements satisfy the requested shape and size.

Table 1 shows the breakdown of the total time spent by the PGDR code on the refinement of the unit cube for different maximum octree depths d. For the experiments with the unit cube we specified a uniform grading function; therefore, all leaves in the resulting octree have the same depth, and the number of leaves is 8^{d-1}. These data reflect the tradeoff between the available concurrency and the sequential overheads in our approach: in order to increase the concurrency (number of octree leaves) we need to increase the sequential preprocessing time. However, as we did in our two-dimensional implementation [7], the construction of the tree and the initial mesh refinement can also be parallelized; we are going to address this issue for three dimensions in the future work. Table 2 summarizes the running times of our experiments for the bat model with the maximum octree depth equal to 5. Since the mesh is non-uniform, some leaves have larger size than the smallest possible leaf (which corresponds to the maximum depth).

The scalability of our approach is influenced not only by the sequential overheads, but also by the smoothness of the user-specified grading function. For example, if the grading function has a sharp peak at one point of the domain, then the computation time associated with the corresponding leaf may dominate the entire running time and, as a result, the speedup from using multiple cores may be poor to none. Therefore, the analysis needs to be conducted separately for different classes of grading functions. For constant grading functions, for example, one can use a number of optimizations and simplifying assumptions which allow to improve the performance.

Our previous experiments with a two-dimensional implementation and constant grading functions, which result in uniform meshes, indicate that one can achieve almost linear speedup in practice on a cluster of up to 121 nodes [8]. Assuming linear mesh refinement time with respect to the final number of triangles, as well as no sequential scheduling overheads (since refinement can follow a regular pattern), we can estimate the theoretical asymptotic speedup in terms of the number P of Processing Elements (PEs) and the size N in triangles of the final mesh. From the arrangement of the PEs in a regular grid, such that each PE is responsible for the refinement of a specific area (square) of the domain, each PE communicates with a constant number of PEs that refine the adjacent squares. The total communication time in this case is $\mathcal{O}\left(\sqrt{N/P}\right)$. Since the number of squares is $\mathcal{O}(P)$, and the number of triangles required to satisfy the relation between the upper bound on triangle circumradius and the side length of the square is constant, the sequential preprocessing required to reach concurrency P takes $\mathcal{O}(P)$ time. Assuming the preprocessing can also be parallelized, the time to reach concurrency k is $\mathcal{O}\left(1 + \sum_{i=1}^{k-1} 1/i\right)$, and, therefore, the time to reach concurrency P can be reduced to $\mathcal{O}(\log P)$. Thus, the total parallel running time is $\mathcal{O}\left(\log P + (N - \log P)/P + \sqrt{N/P}\right)$. For practical problems $N \gg P$, and, therefore, the speedup of the algorithm is linear with respect to P. The constants, however, may be non-negligible and will be studied experimentally in our future work.

5. SUMMARY

We presented the the first to our knowledge three-dimensional parallel Delaunay mesh generator which, compared to the

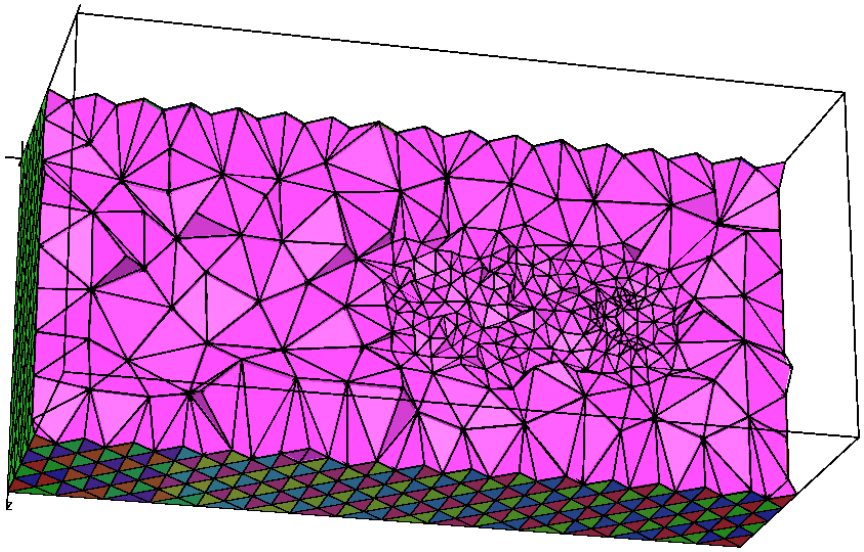

Figure 11: A nonuniform mesh of the bat inside a box.

Table 1: The breakdown of the time (in seconds) spent by the PGDR code on the refinement of the unit cube for different maximum octree depths, $\bar{r} = 0.01$, 6.4 million tetrahedra.

Maximum octree depth	Time measured	Number of available compute threads			
		1	2	3	4
2	Total	245.77	same	same	same
	Octree construction (sequential)	0.00	same	same	same
	Initial refinement (sequential)	0.11	same	same	same
	Parallel refinement (maximum)	245.62	same	same	same
3	Total	246.52	same	same	same
	Octree construction (sequential)	0.02	same	same	same
	Initial refinement (sequential)	0.69	same	same	same
	Parallel refinement (maximum)	245.78	same	same	same
4	Total	250.20	145.62	126.67	115.76
	Octree construction (sequential)	0.26	same	same	same
	Initial refinement (sequential)	5.24	same	same	same
	Parallel refinement (maximum)	244.61	140.05	121.10	110.19
5	Total	262.36	156.39	121.52	104.08
	Octree construction (sequential)	2.74	same	same	same
	Initial refinement (sequential)	42.98	same	same	same
	Parallel refinement (maximum)	216.30	110.34	75.51	58.08
6	Total	761.38	575.26	514.01	483.50
	Octree construction (sequential)	24.74	same	same	same
	Initial refinement (sequential)	359.84	same	same	same
	Parallel refinement (maximum)	374.69	190.03	128.59	98.30

previous approaches, benefits from all of the following properties: (1) offers quality guarantees on the shape of the elements in terms of circumradius-to-shortest edge ratio (that are directly inherited from the underlying sequential algorithms and software); (2) offers asymptotic proofs of element good grading (which are also inherited from the sequential counterparts); (3) makes it possible to use a user-defined grading function to control element sizes; (4) makes it possible to use custom point placement strategies; (5) replaces the solution of a difficult domain decomposition problem with an easier data distribution approach without relying on the speculative execution model [21]; (6) leverages sequential algorithms and software for Delaunay mesh refinement; (7) offers more than 50% performance improvement over the state-of-the-art sequential software, even on workstations with just a few hardware cores.

6. ACKNOWLEDGMENTS

We thank George Karniadakis for the bat model. This work was performed using computational facilities at the College of William and Mary which were enabled by grants from Sun Microsystems, the National Science Foundation, and Virginia's Commonwealth Technology Research Fund. This work was supported (in part) by the NSF grant CCS-0750901 and by the John Simon Guggenheim Foundation. We thank the anonymous referees for their comments.

7. REFERENCES

[1] http://www.distene.com/en/build/index.html. Accessed on Jan. 13, 2008.

[2] C. D. Antonopoulos, X. Ding, A. N. Chernikov, F. Blagojevic, D. S. Nikolopoulos, and N. P. Chrisochoides. Multigrain parallel Delaunay mesh generation: Challenges and opportunities for multithreaded architectures. In *Proceedings of the 19th Annual International Conference on Supercomputing*, pages 367–376, Cambridge, MA, 2005. ACM Press.

[3] G. E. Blelloch, J. Hardwick, G. L. Miller, and D. Talmor. Design and implementation of a practical parallel Delaunay algorithm. *Algorithmica*, 24:243–269, 1999.

[4] A. Bowyer. Computing Dirichlet tesselations. *Computer Journal*, 24:162–166, 1981.

[5] P. B. Callahan and S. R. Kosaraju. A decomposition of multidimensional point sets with applications to k-nearest-neighbors and n-body potential fields. *Journal of the ACM*, 42(1):67–90, 1995.

[6] S.-W. Cheng, T. K. Dey, and T. Ray. Weighted Delaunay refinement for polyhedra with small angles. In *Proceedings of the 14th International Meshing Roundtable*, pages 325–342, San Diego, CA, Sept. 2005. Springer.

[7] A. N. Chernikov and N. P. Chrisochoides. Generalized Delaunay mesh refinement: From scalar to parallel. In *Proceedings of the 15th International Meshing Roundtable*, pages 563–580, Birmingham, AL, Sept. 2006. Springer.

[8] A. N. Chernikov and N. P. Chrisochoides. Parallel guaranteed quality Delaunay uniform mesh refinement. *SIAM Journal on Scientific Computing*, 28:1907–1926, 2006.

[9] A. N. Chernikov and N. P. Chrisochoides. Three-dimensional semi-generalized point placement method for Delaunay mesh refinement. In *Proceedings of the 16th International Meshing Roundtable*, pages 25–44, Seattle, WA, Oct. 2007. Springer.

[10] L. P. Chew. Guaranteed quality mesh generation for curved surfaces. In *Proceedings of the 9th ACM Symposium on Computational Geometry*, pages 274–280, San Diego, CA, 1993.

[11] L. P. Chew. Guaranteed-quality Delaunay meshing in 3D. In *Proceedings of the 13th ACM Symposium on Computational Geometry*, pages 391–393, Nice, France, 1997.

[12] H. L. de Cougny, M. S. Shephard, and C. Ozturan. Parallel three-dimensional mesh generation. *Computing Systems in Engineering*, 5:311–323, 1994.

[13] B. N. Delaunay. Sur la sphere vide. *Izvestia Akademia Nauk SSSR, VII Seria, Otdelenie Mataematicheskii i Estestvennyka Nauk*, 7:793–800, 1934.

[14] W. H. Frey. Selective refinement: A new strategy for automatic node placement in graded triangular meshes. *International Journal for Numerical Methods in Engineering*, 24(11):2183–2200, 1987.

[15] P.-L. George and H. Borouchaki. *Delaunay Triangulation and Meshing. Application to Finite Elements*. HERMES, 1998.

[16] M. Kulkarni, K. Pingali, B. Walter, G. Ramanarayanan, K. Bala, and L. P. Chew. Optimistic parallelism requires abstractions. *SIGPLAN Not.*, 42(6):211–222, 2007.

[17] X.-Y. Li. Generating well-shaped d-dimensional Delaunay meshes. *Theoretical Computer Science*, 296(1):145–165, 2003.

[18] L. Linardakis and N. Chrisochoides. Algorithm 870: A static geometric medial axis domain decomposition in 2D Euclidean space. *ACM Transactions on Mathematical Software*, 34(1):1–28, 2008.

[19] R. Löhner and J. R. Cebral. Parallel advancing front grid generation. In *Proceedings of the 8th International Meshing Roundtable*, pages 67–74, South Lake Tahoe, CA, 1999.

[20] G. L. Miller, D. Talmor, S.-H. Teng, and N. Walkington. A Delaunay based numerical method for three dimensions: Generation, formulation, and partition. In *Proceedings of the 27th Annual ACM Symposium on Theory of Computing*, pages 683–692, Las Vegas, NV, May 1995.

[21] D. Nave, N. Chrisochoides, and L. P. Chew. Guaranteed–quality parallel Delaunay refinement for restricted polyhedral domains. In *Proceedings of the 18th ACM Symposium on Computational Geometry*, pages 135–144, Barcelona, Spain, 2002.

[22] L. Oliker and R. Biswas. Parallelization of a dynamic unstructured application using three leading paradigms. In *Supercomputing '99: Proceedings of the 1999 ACM/IEEE conference on Supercomputing (CD-ROM)*, page 39, New York, NY, USA, 1999. ACM Press.

[23] I. Pivkin, E. Hueso, R. Weinstein, D. Laidlaw, S. Swartz, and G. Karniadakis. Simulation and visualization of air flow around bat wings during flight. In *Proceedings of the International Conference on Computational Science*, pages 689–694, Atlanta, GA, 2005.

[24] K. Psarris and K. Kyriakopoulos. The impact of data dependence analysis on compilation and program parallelization. In *Proceedings of the 17th annual international conference on Supercomputing*, pages 205–214, New York, NY, USA, 2003. ACM.

[25] J. R. Shewchuk. Tetrahedral mesh generation by Delaunay refinement. In *Proceedings of the 14th ACM Symposium on Computational Geometry*, pages 86–95, Minneapolis, MN, 1998.

[26] H. Si. Tetgen version 1.4.1. http://tetgen.berlios.de/. Accessed on Aug. 3, 2006.

[27] H. Sutter. The free lunch is over: A fundamental turn toward concurrency in software. *Dr. Dobb's Journal*, 30(3), Mar. 2005.

[28] R. A. van Engelen and K. Gallivan. The gSOAP toolkit for web services and peer-to-peer computing networks. In *Proceedings of the 2nd IEEE International Symposium on Cluster Computing and the Grid*, pages 128–135, Berlin, Germany, May 2002.

[29] D. F. Watson. Computing the n-dimensional Delaunay tesselation with application to Voronoi polytopes. *Computer Journal*, 24:167–172, 1981.

[30] C. Wen and K. Yelick. Compiling sequential programs for speculative parallelism. In *Proceedings of the International Conference on Parallel and Distributed Systems*, Taiwan, Dec. 1993.