International Journal of Computational Geometry & Applications
Vol. 9, No. 6 (1999) 577–597

DATA GENERATION FOR GEOMETRIC ALGORITHMS ON NON-UNIFORM DISTRIBUTIONS

GARY L. MILLER
School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, USA

DAFNA TALMOR
CADSI, Suite 104, 3150 Almaden Expressway, San Jose, CA 95118, USA

SHANG-HUA TENG
Department of Computer Science, University of Illinois at Urbana Champaign, Urbana, IL 61801, USA
E-mail: steng@cs.uiuc.edu

Received 23 June 1996
Revised 23 November 1998
Communicated by N. Amenta and S. J. Fortune

ABSTRACT

We study the geometric properties of point sets that arise in the generation of bounded aspect-ratio meshes and present a constructive formulation to define distributions that allow arbitrary refinements. This formulation can be used to define distributions with one or more singularities, which do not occur in the uniform case but do occur in mesh generation for real-world applications. We give an efficient algorithm for the generation of a point set from these distributions.

This work is in part motivated by the following observation: The Poisson distribution, points placed uniformly and randomly in a fixed dimension, is one of the most commonly used classes of data sets in the experimental evaluation of geometric algorithms and their implementation. However, despite its importance and interest to computational geometry, the Poisson distribution fails to be good test data for triangulation algorithms and software for mesh generation. Consequently, many implemented sequential and parallel algorithms are tuned to work efficiently for the uniform distribution, but fail to be efficient for nonuniform distributions. Even though the focus of our work is on the generation of data for Delaunay-based mesh algorithms, we hope that it will motivate further theoretical investigations on the generation of data for other geometric algorithms and software.

Keywords: Delaunay triangulations, computational geometry, mesh generation, randomized algorithms, well-spaced points.

1. Introduction

The uniform distribution is one of the most interesting and important classes of

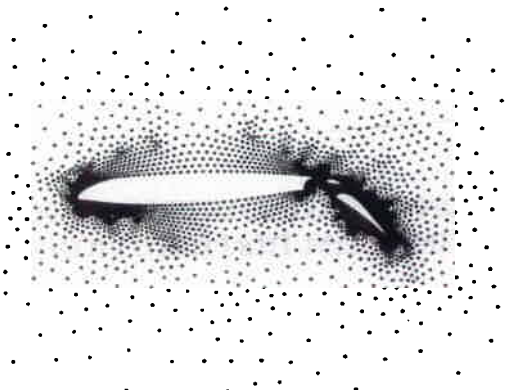

Fig. 1. A point set for a numerical discretization around an airfoil, based on a mesh generated by Barth and Jespersen

point distributions in computational geometry. Various geometric algorithms are designed to exploit the expected properties of this distribution. Examples include the divide-and-conquer convex-hull algorithm of Bentley and Shamos,[3] the nearest neighbors algorithm of Bentley, Weide, and Yao,[4] and the Voronoi and Delaunay diagrams algorithm of Dwyer[9] and its parallel extensions.[29,27]

Several important considerations lead researchers to choose the uniform distribution as experimental data to test and to evaluate geometric algorithms. The uniform distribution models some physical processes (such as arrival time) and therefore is natural for certain problems. Further, many geometric graphs, such as Delaunay diagrams and relative neighborhood graphs, when constructed over a point set from the uniform distribution, have expected linear size even in higher dimensions. The sparseness implies experiments can be performed efficiently. The uniform distribution can be used to test how well algorithms and software handle problematic issues such as data degeneracy. Last, but not least, the uniform point set can be easily generated.

However, the experimental data must be relevant to the objectives of the experiment. In general the goal of an experiment is to test certain theoretical conjectures and theorems, to check the correctness and complexity of an algorithm and its implementation, and/or to recognize patterns of computations in order to improve the efficiency of the algorithm. Given that software is usually targeted at certain classes of applications, it is crucial to generate data sets that best represent the range of problems in these applications.

Non-uniform distributions arise naturally in many real-world applications. Figure 1 gives an example of a non-uniform point distribution around an airfoil. A typical example is mesh generation in scientific computing. If we only use the uniform distribution to experimentally evaluate and improve Delaunay based mesh generation algorithms, we may mistakenly fine-tune our algorithms and software to work most efficiently for the uniform distribution; these fine-turned algorithms may fail on naturally occurring non-uniform distributions. Even when the underlying point density is regular, the uniform distribution might not reflect other properties of the data sets targeted by an algorithm. For example, Bern, Eppstein, and Yao[7]

have shown that the expected smallest angle of the Delaunay triangulation of a set of points placed uniformly and randomly in two dimensions is $\Theta(1/\sqrt{n})$. This implies, with high probability, that a uniform random point set (e.g., from the Poisson distribution) does not have a bounded aspect-ratio. Therefore, random point sets do not adequately represent bounded aspect-ratio meshes even for domains with a uniform density requirement.

The goal of this work is to give a theoretical foundation of distributions encountered in certain practical domains, to provide an understanding of point sets that approximate these distributions, and to develop an efficient algorithm for generating these data. In this paper, we focus on the problem of the design of relevant data for testing Delaunay based algorithms/software in mesh generation. We characterize the geometric properties of point sets that arise in bounded aspect-ratio meshes and present a constructive formulation to define distributions that allow arbitrary refinements, and include distributions with one or more singularities which do not occur in the uniform case but do occur in real-world applications. We show that a point set from such a distribution can be generated efficiently. We also give a formulation that allows users to design their own classes of distributions. Even though our work focuses on the generation of experimental data for Delaunay-based mesh algorithms and software, we hope that it will motivate further theoretical investigations on the generation of experimental data for other geometric algorithms and software.

Another goal for our work is to provide computational geometers an algorithm that efficiently generates test data for output sensitive Delaunay triangulation algorithms and provide numerical analysts an automatic procedure for the generation of a large class of unstructured sparse matrices and finite-element meshes.

The rest of the paper is as follows: in Section 2, we review Lipschitz conditions and introduce well-spaced points for Delaunay-based mesh generation. In Section 3, we show how to superimpose simple local spacing functions to model multiple singularities and maintain the Lipschitz conditions. In Section 4, we present an efficient algorithm for the generation of well-spaced point sets. We will also present the main components of our well-spaced point set generator. In Section 5, we analyze the spacing function of the Plummer distribution, an important point distribution in Astrophysics. In Section 6 we conclude this paper and discuss the future directions motivated by this work.

2. A Geometric Characterization of Well-Spaced Point Sets

In this section, we examine the properties of point sets associated with bounded aspect-ratio meshes. The use of well-spaced point sets is implicitly at the heart of many numerical methods such as the finite-element and finite-volume methods.[18,26] Well-spaced point sets also have potential applications in smooth particle hydrodynamics (SPH). We can describe the discretization process associated with mesh generation as a two-phase process. In the first phase, we generate a set of points that satisfies both geometric and numerical conditions imposed on the physical domain. In the second phase we construct a robust and bounded aspect-ratio mesh

over this point set, e.g., the Delaunay triangulation of the point set. Mesh generation algorithms often merge the two phases, and generate the point set implicitly as part of the mesh generation phase. Keeping the two phases distinct enables us to concentrate on the properties of the point sets.

The input description of a physical domain has two components: a geometric model of the domain and a numerical condition within the domain. The geometric model provides the boundary of the domain either in the form of a continuous model or of a discretized boundary model. The continuous model, such as constructive solid geometry (CSG),[23,31] defines a domain in terms of union, intersection, and complement of basic geometric objects such as boxes, spheres, or other simple objects. The discretized boundary model defines the boundary of the domain in a piecewise fashion using low degree polygonal elements. The discretized boundary model is often used in mesh generation. A description of the domain in the continuous model can be transformed into a discretized boundary description by replacing basic geometric objects with their local discretizations. The geometric model of the domain defines a local spacing function restricting the final discretization of the domain. The numerical condition within the domain is typically obtained from an initial numerical simulation on a preliminary set of points. It defines an additional local spacing function restricting the final point set.

2.1. The Numerical Spacing Function

The numerical spacing function, typically denoted by $h(x)$, defines a scalar at each point x. It is usually defined at a point x by the largest eigenvalues of the Hessian matrix of the solution u to the governing partial differential equations (PDEs).[2,26,18] Locally, u behaves like a quadratic function

$$u(x + dx) = \frac{1}{2}(xHx^T) + x\nabla u(x) + u(x),$$

where H is the *Hessian matrix* of u, the matrix of second partial derivatives. The spacing of mesh points, $h(x)$, required by the accuracy of the discretization at a point x, should depend on the reciprocal of the square root of the largest eigenvalues of H at x.[26]

When solving a PDE numerically, we estimate the eigenvalues of the Hessian matrix at a certain set of points in the domain based on the numerical approximation of the previous iteration.[2,26] We then expand the spacing requirement induced by the Hessian at these points over the entire domain.

2.2. The Geometric Spacing Function

For the finite element formulation, Strang and Fix[26] have given an error bound that depends on the aspect-ratio of the mesh elements. Babuška and Aziz[2] have shown that the largest angle of the mesh elements should be bounded away from π. Recently, it has been shown in Ref. [18] that a weaker aspect-ratio condition in conjunction with the finite volume method on Delaunay triangulations also achieves an optimal error bound.

All three results above require that the local spacing of the mesh points changes slowly spatially. The magnitude of change in the point spacing reflects the aspect-ratio, or its variants, achieved by the mesh. The geometry of the boundary of a domain implicitly defines a local feature function extending from the boundary to the entire domain, and possessing a bounded derivative almost everywhere . This "geometric local feature function" is much better understood thanks to the work of Bern-Eppstein-Gilbert,[5] Mitchell-Vavasis,[19] Ruppert,[24] Mitchell,[20] and Miller *et al.*[18]

Given a set of input features F, e.g., a collection of points or simple geometric objects, one definition of the geometric local feature function at point x in the domain, $\text{lfs}_F(x)$, introduced by Ruppert,[24] is the maximum distance from x to the closest two non-adjacent features (either corner-vertices or boundary faces) of the domain. The geometric local feature size of a domain Ω then is given by $\text{lfs}_\Omega(x)$.

The lfs function, as shown by Ruppert, is ***Lipschitz*** with a constant 1.

Definition 1 (Lipschitz) *A function f is Lipschitz with constant α if for any two points x, y in the domain, $|f(x) - f(y)| \leq \alpha||x - y||$.*

The spacing requirement of the final mesh then must satisfy both geometric and numerical features.

2.3. A Geometric Characterization

The combination of the numerical and geometric requirements of a continuous problem defines a ***local spacing function*** over a domain. We denote this function by ls(). This spacing function must satisfy both geometric and numerical features, i.e., $\text{ls}(x) \leq \min(h(x), \text{lfs}_\Omega(x))$ for every point x in the domain. In Section 3, we will give a constructive formulation for users to define these two spacing functions, and to combine them.

As shown in Refs. [5,24,17,18], if M is a bounded aspect-ratio mesh whose size is optimal up to a constant factor, then its point set P ***respects*** the local spacing function ls() in the following sense: There is a constant $\beta \geq 1$ such that for every point x in the domain,

$$\text{ls}(x)/\beta \leq \text{lfs}_P(x) \leq \beta \text{ls}(x). \tag{1}$$

Figure 2 shows a bounded aspect-ratio mesh around a singularity.

Condition (1) implies that the point set P of the mesh M defines a local feature function that is linearly related to the spacing function ls(). In other words, the upper bound on $\text{lfs}_P()$ given in (1) implies that the point set P must be dense enough point-wise with respect to the local spacing function ls(), whereas the lower bound on $\text{lfs}_P()$ given in (1) implies that points in P are not too close to each other in terms of the local spacing. The notion of an ideal well-spaced point set is given more formally in the following definition.

Definition 2 (Maximally Spaced Points) *Suppose* ls() *is a Lipschitz local spacing function and P is a point set over a domain Ω. Then (1) P is* ls**-spaced** *if for any two points p and $q \in P$, $||p - q|| \geq \min(\text{ls}(p), \text{ls}(q))$. (2) S is* **maximally**

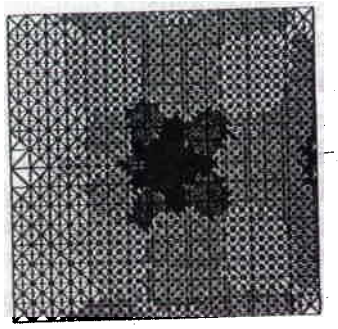

Fig. 2. Triangulation of well-spaced point set around a singularity. The mesh was generated by Omar Ghattas and Xiaogang Li.[21]

ls-*spaced if no point from* Ω *can be added to* S *without violating the* ls-*spacing condition.*

For mesh generation, we can relax the maximality condition in Definition 2.

Definition 3 *Let* δ *be a positive constant. A point set* P *is a* δ-**sample** *with respect to* ls() *if for all* $x \in \Omega$, *the ball of radius* $\delta\text{ls}(x)$ *centered at* x *contains at least one point of* P.

Lemma 1 *If* ls *has Lipschitz constant* α, P *is an* ls-*spaced* δ-*sample, and* $\alpha\delta < 1$, *then for every point* x *in the domain,*

$$\text{ls}(x)/(2+\alpha) \leq \text{lfs}_P(x) \leq \frac{1+3\delta}{1-\alpha\delta}\text{ls}(x).$$

Proof. We first prove the lower bound. Let B be the ball centered at x of radius $\text{lfs}_P(x)$, i.e., B is the smallest ball that contains two points from P. Let p and q be these two points. Clearly one of p and q must be on the surface of B. Because P is ls-spaced, we have $\min(\text{ls}(p), \text{ls}(q)) \leq \|p-q\| \leq 2\text{lfs}_P(x)$. Because ls is α-Lipschitz, we have $\text{ls}(x) \leq \min(\text{ls}(p), \text{ls}(q)) + \alpha\text{lfs}_P(x) \leq (2+\alpha)\text{lfs}_P(x)$.

We now prove the upper bound. Let C be the ball centered at x of radius $\delta\text{ls}(x)$. Because P is a δ-sample, C must contain a point, say u, from P. Let $v \in P$ be the point that is closest to u. We now estimate the distance from u to v. Let $K = \delta/(1-\alpha\delta)$. If we look at a point v' at distance $K\text{ls}(u)$ from u, its ls is at most $\text{ls}(u) + \alpha K\text{ls}(u) = \text{ls}(u)/(1-\alpha\delta)$, and therefore a $\delta\text{ls}(v')$ ball centered at v' does not touch u and must contain another input point v. The distance from u to v is therefore smaller than $2\delta\text{ls}(u)/(1-\alpha\delta)$. Thus, we have $\text{lfs}_P(x) \leq \delta\text{ls}(x) + \text{ls}(u)\frac{2\delta}{1-\alpha\delta}$. Because ls is α-Lipschitz, $\text{ls}(u) \leq \text{ls}(x) + \alpha\delta\text{ls}(x)$. Therefore,

$$\text{lfs}_P(x) \leq \frac{1+2\delta+\alpha\delta^2}{1-\alpha\delta}\text{ls}(x) \leq \frac{1+3\delta}{1-\alpha\delta}\text{lfs}(x).$$

□

Point set P is ls()-*spaced* δ-*sample* if P is ls-spaced and a δ-sample with respect to ls(). Such sets partially capture the intuition of well-spaced point sets.

In Section 4, we will show that if the Lipschitz constant of ls() is α and $\alpha\delta < 1$, then the Delaunay triangulation of any ls()-spaced δ-sample satisfies the aspect-ratio condition defined in Ref. [18]. Notice that in the results above we ignore boundary effects. More discussion and techniques for coping with the boundary effects will be given in Section 4.3.

3. Constructive Feature Geometry

In order to automatically generate test data using local spacing functions, we need to have a *constructive* formulation for a user to specify the properties of a distribution from his/her applications. A distribution is described in terms of its size, locations, intensities, shape of singularities, and its boundary.

Motivated by *constructive solid geometry*[23,31] which defines complex geometric objects as union and intersection of a set of basic geometric objects, we propose to use *constructive feature geometry* (CFG) to define complex feature functions/distributions from a set of basic feature functions/distributions. The global feature function is then defined by applying a *super-imposing* operator to a set of basic feature functions.

Distributions with more than one singularity and with various intensities and shapes can be modeled by properly choosing the set and the number of basic feature functions. We use (and perhaps abuse) the term singularity to indicate the point spacing becoming smaller and point density becoming larger, but in a controlled manner, in a neighborhood of the domain. In the next section we present an efficient algorithm to generate a well-spaced point set from a constructive feature geometry formula.

3.1. Basic Feature Functions

The basic feature functions are defined using simple geometric objects such as points, lines (segments), spheres, and simplices.

3.1.1. Point Features

A point feature is defined over a domain Ω using a point $\boldsymbol{p}$. In addition, we have a value μ that specifies a desired spacing near $\boldsymbol{p}$. The feature $\Phi^{\mu}_{\boldsymbol{p}}(\boldsymbol{x})$ for a point $\boldsymbol{x} \in \Omega$, is equal to $\Phi^{\mu}_{\boldsymbol{p}}(\boldsymbol{x}) = \max(\mu, ||\boldsymbol{x} - \boldsymbol{p}||)$.

Clearly, $\Phi^{\mu}_{\boldsymbol{p}}(\boldsymbol{x})$ is 1-Lipschitz. By choosing $\boldsymbol{p}$ and μ, we can model a singularity of certain intensity near any point in the domain:

3.1.2. Features Around Geometric Objects

Points features are symmetrical in all directions. To model a wider variety of singularities, we use the features defined by higher dimensional geometric objects such as lines (segments), spheres, boxes, and simplices.

Let Γ be such a geometric object and μ be a parameter specifying a desired spacing near the boundary of Γ. The spacing function induced by Γ over the domain Ω is given as follows: for each $\boldsymbol{x} \in \Omega$, $\Phi^{\mu}_{\Gamma}(\boldsymbol{x}) = \max(\mu, ||\boldsymbol{x} - \Gamma||)$, where $||\boldsymbol{x} - \Gamma||$ is

the distance between x and Γ.

Clearly, $\Phi_\Gamma^\mu(x)$ is 1-Lipschitz.

3.2. Constructive Feature Geometry (CFG) by Super-imposing

A super-imposing operator can be used to combine the local features defined by points and basic geometric objects. These points define the centers of singularities whereas the objects model the locations and shape of singularities.

Suppose Ω is a domain and $\Phi_1(), ...\Phi_m()$ are k basic local spacing functions. The spacing function $\Phi()$ obtained by super-imposing these basic functions is given by

$$\Phi(x) = \min_{i=1}^{m} \Phi_i(x).$$

The simplest CFG formula is defined by a set of point feature functions. Let $P = \{p_1, ..., p_k\}$ be a set of points in Ω, representing the centers of singularities. Suppose the local spacing of p_i is μ_i. Then the global feature function is:

$$\Phi(x) = \min_{i=1}^{k} \Phi_{p_i}^{\mu_i}(x).$$

The following lemma shows that super-imposing of two functions preserves the smoothness of both functions.

Lemma 2 *If f_1 and f_2 are α-Lipschitz over Ω, then $\min(f_1, f_2)$ is also α-Lipschitz over Ω.*

Proof. Let x and y be two points in Ω. Let $f() = \min(f_1(), f_2())$. Without loss of generality, assume that $f(x) \leq f(y)$ and $f(x) = f_1(x)$. There are two cases. (1) If $f(y) = f_1(y)$, then because f_1 is α-Lipschitz, $|f(y) - f(x)| \leq \alpha||y - x||$. (2) If $f(y) = f_2(y)$, we have $|f(y) - f(x)| \leq |f_1(y) - f_1(x)| \leq \alpha||y - x||$. Thus, f is α-Lipschitz. □

Corollary 1 *$\Phi()$ defined above is 1-Lipschitz. It also respects the spacing restrictions in the sense that at each point x, $\Phi(x)$ is no larger than the spacing restriction of all local feature functions.*

4. Well-Spaced Point-Set Generation

In this section, we present two techniques, *oversampling* and *filtering*, and show how to apply them to generate a well-spaced point set according to a local spacing function. We will prove the following theorem.

Theorem 1 (Point Generation) *Suppose* ls *is a spacing function of Lipschitz constant α in a domain Ω. For any β and δ such that $\beta\delta < 1/\alpha$, a $\beta \cdot$ ls()-spaced δ-sample of size $O(K)$ can be generated in $O(K \log K)$ time, where K is the size of an optimal well-spaced point set for* ls *over the domain Ω.*

4.1. Point Generation for the Uniform Distribution

The following discussion of uniform distribution may be used to motivate our construction. Bern, Eppstein, and Yao[7] have shown that the aspect-ratio and degree

of the Delaunay diagram of a uniform point set generated by the Poisson process are all unbounded.

The homogeneous Poisson point process of intensity one is a standard model characterized by the property that the number of points in a region is a random variable that depends only on the d-dimensional volume of the region.[15,14,7] In this model,

- The probability of exactly k points appearing in any region of volume V is $e^{-V}V^k/k!$.
- The conditional distribution of points in any region given that exactly k points fall in the region is joint uniform.

Theorem 2 (BEY) *If P is a set points generated by the Poisson point process with intensity one over a $\sqrt{n} \times \sqrt{n}$ square, then with very high probability ($> 1 - 1/n$), the smallest angle in the Delaunay diagram of P is equal to $\Theta(1/\sqrt{n})$.*

Therefore, with very high probability, a point set from a uniform distribution is not well-spaced. Suppose instead of using the Poisson point process with intensity one, we oversample such that with high probability each unit area has at least one point. Then, still, there is no *a priori* bound on the degree nor the aspect-ratio, because with very high probability, there are two points in the uniform set as well as the oversampled set that are too close to each other.

Our idea is to selectively remove some of the extra points after this oversampling. By carefully using these two techniques, *oversampling* and *filtering*, we can efficiently generate a point set whose Delaunay diagram has a constant degree and a bounded aspect-ratio.

Suppose, a point set P is produced by oversampling the uniform distribution, such that each unit ball contains a point from P. The filtering process uses a *conflict graph* G, in which each vertex corresponds to a unit ball centered at a sample point, and the vertices of two balls are connected if each of the two balls contains the other's center in its interior. The output of the filtering process is simple a *maximal independent set* S of G.

Lemma 3 *The set S constructed above is a 2-sample for the uniform spacing function.*

Proof. Because S is an independent set of the conflict graph, the distance between any two points from S is at least 1.

We now show that S is a 2-sample with respect to the uniform spacing function. Let x be a point in the domain. Let B be the ball of radius 2 centered at x, as drawn in Figure 3. We assume by contradiction that B contains no point from S. Let C be the unit ball centered at x. By the assumption of oversampling, C contains a point from P. Call this point p. Let D be the unit ball centered at p. Note that D is properly contained in B, and hence contains no point from S.

This implies that p does not conflict with any point in S, contradicting the assumption that S is a maximal independent set of the conflict graph. Thus, B must contain a point from S and hence S is a 2-sample with respect to the uniform spacing function. □

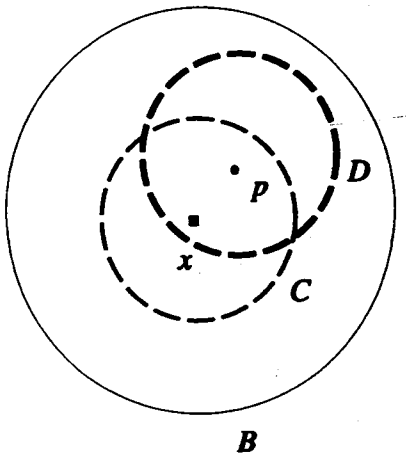

Fig. 3. An Example for Lemma 3.

4.2. *Well-Spaced Point Set Generation*

Our approach for well-spaced point set generation is to first oversample a random point set P such that for each point $x \in \Omega$, the ball of radius $\text{ls}(x)$ centered at x contains a sample point. We then apply filtering to P to obtain an ls-spaced $(2+\alpha)$-sample.

4.2.1. Hierarchical Trees for Feature Approximation

The oversampling process is guided by a balanced decomposition of Ω, i.e., a quad-tree decomposition in two dimensions and an oct-tree decomposition in three dimensions.

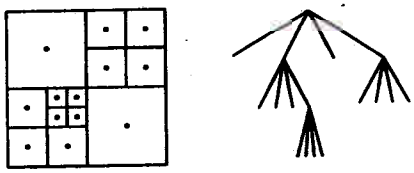

Fig. 4. Quad-tree

A *quad-tree*[25] is a recursive partition of a region of the plane into axis-aligned squares. One square, the *root*, covers the entire domain Ω. It is often chosen to be a constant factor of the smallest square that contains the domain. A square can be divided into four *child* squares, by splitting it with horizontal and vertical line segments through its center. The collection of squares then forms a tree, with smaller squares at lower levels of the tree. The recursive decomposition is often adaptive to the local geometry and local spacing function (See Figure 4). *Oct-tree* is the three-dimensional version of quad-tree. Oct-trees are constructed by recursively and adaptively dividing a box into eight child-boxes, by splitting it with hyper-planes normal to each axis through its center (See Figure 5).

We refer to both quad-trees and oct-trees as *hierarchical trees*. Balanced hierar-

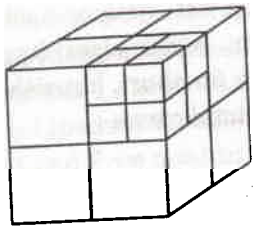

Fig. 5. Oct-tree

chical trees, introduced by Bern, Eppstein, and Gilbert,[5] can approximate a feature function. A *balanced hierarchical tree* has no leaf-box adjacent to another leaf-box more than twice its side length.

Definition 4 *Let T be a balanced hierarchical tree over a domain Ω. Define a function f_T such that $f_T(x) = l$ if l is the side length of the box that contains the point x. The function f_T approximates a local spacing function* ls *if $\exists c_2, c_1 > 0$ such that $c_1 f_T \leq \text{ls} \leq c_2 f_T$ point-wise in Ω.*

Lemma 4 *Given a domain Ω with a Lipschitz spacing function* ls *(in the CFG form), a hierarchical tree T of size $O(K)$ that approximates* ls *can be generated in $O(K \log K)$ time, where K is the size of an optimal well-spaced point set for* ls *over the domain Ω.*

Proof. We generate two separate hierarchical trees, one is for the local spacing function ls and another is for the domain itself. The union of these two tree define the final hierarchical tree.

The function ls is given as a combination (using the min operator over N basic feature functions). We make the assumption that the final size of the tree, K, is larger than N. This is true if each basic feature function contributes new information about the structure of the tree. We also assume that the hierarchical tree boxes have rational binary co-ordinates and lengths which are a power of two. Consequently, when a point is given together with the size of the box containing it in the hierarchical tree, we can create in $O(1)$ time a description of that box in terms of its corner and length. For each basic feature function, we create a set of leaf-boxes that intersect with its geometric object: A point feature has one leaf-box, but a general geometric objects may have many more.

The algorithm in Ref. [6] can be used to generate the balanced hierarchical tree over these boxes. The optimality of K follows from the proof of Mitchell and Vavasis[19] for 3D and Bern, Eppstein, and Gilbert[5] for 2D.

Because the domain Ω is given in the CSG form, we can use a planar straight-line graph (PSLG) to model it in 2D (as in Bern-Eppstein-Gilbert and Ruppert) and a polyhedral region with holes in 3D (an in Mitchell-Vavasis). Therefore, computationally, we can apply Bern-Eppstein-Gilbert's balanced quad-tree and Mitchell-Vavasis's oct-tree algorithms and software to support our hierarchical decomposition. The $O(K \log K)$ time bound follows directly from the results in Refs. [5,19]. □

A simpler algorithm, which is not theoretically optimal, is to create the tree level by level. At each level we have a set of active (non-leaf) tree boxes, and a set

of input boxes. We split a tree box if it contains a set of input boxes of smaller size, or it has a small leaf box neighbor. In this algorithm we pay linearly for each level, and therefore will be optimal only for short, logarithm depth trees. Nonetheless, it is a simple first-cut algorithm to implement.

4.2.2. Oversampling and Filtering

Given the hierarchical tree T that approximates ls, an oversampling of ls can be generated as follows: In each leaf-box of T, we place a set of sample points such that the $\text{ls}(\boldsymbol{x})$-ball at each point $\boldsymbol{x}$ contains at least one sample point. One way to generate such sample points is to uniformly and randomly generate $\Omega(\log K)$ points at each leaf-box. Note that the number of leaf-boxes is $\Theta(K)$. Then with high probability, the $\text{ls}(\boldsymbol{x})$-ball at each point $\boldsymbol{x}$ contains at least one sample point. An alternative way to oversample is to choose a constant integer $L = \lceil c_2/c_1 \rceil$ and divide each leaf-box into L^d smaller boxes of equal size. We then choose a random point from each smaller box.

Let P be the set of sampling points constructed in the oversampling procedure. The filtering is guided by a conflict graph G_P defined on P. The set of vertices of G_P is P. Two vertices $\boldsymbol{p}$ and $\boldsymbol{q} \in P$ are connected by an edge of G_P if $||\boldsymbol{p} - \boldsymbol{q}|| < \min(\text{ls}(\boldsymbol{p}), \text{ls}(\boldsymbol{q}))$. Geometrically, this condition implies that the $\text{ls}(\boldsymbol{p})$-ball at $\boldsymbol{p}$ contains $\boldsymbol{q}$ and the $\text{ls}(\boldsymbol{q})$-ball at $\boldsymbol{q}$ contains $\boldsymbol{p}$. Let S be a maximal independent set of G_P. We call S a *filtering* of P.

Lemma 5 *Let* ls() *be a local spacing function over a domain* Ω *with Lipschitz ratio* α *and let* P *be a 1-sample with respect to* ls(). *If* S *is a filtering of* P, *then* S *is a* ls-*spaced* $(2+\alpha)$-*sample.*

Proof. Because S is an independent set of G_P, it follows directly from the definition of G_P that S is an ls-spaced point set.

We now prove that it is a $(2+\alpha)$-sample with respect to ls. Let $\boldsymbol{x}$ be a point in Ω. Let B be the $(2+\alpha) \cdot \text{ls}(\boldsymbol{x})$-ball centered at $\boldsymbol{x}$. To prove the lemma by contradiction, we assume that B does not contain a point from S. Let C be the $\text{ls}(\boldsymbol{x})$-ball center at $\boldsymbol{x}$. Because P is a 1-sample with respect to ls, C contains a point from P. Call this point $\boldsymbol{p} \in P$. Because ls is α-Lipschitz, $\text{ls}(\boldsymbol{p}) \leq \text{ls}(\boldsymbol{x})(1+\alpha)$. Let D be the $\text{ls}(\boldsymbol{p})$-ball centered at $\boldsymbol{p}$. Note that D is properly contained in B and hence does not contain a point from S. Therefore, $\boldsymbol{p}$ does not conflict with any point in S, contradicting with the assumption that S is a maximal independent set of G_P. Therefore B must contains a point from S and hence S is a $(2+\alpha)$-sample with respect to ls. □

4.2.3. A Well-Shaped Point Generation Algorithm

Lemma 4 shows that a quad-tree or a oct-tree can be used to approximate a Lipschitz function efficiently and optimally in two and three dimensions, respectively. Lemma 5 guarantees that after oversampling and filtering, we construct an approximate ls-spaced point set. Lemma 1 shows that an approximate ls-spaced point set is a well-spaced point set.

However, most of our lemmas assume that the spacing-function ls() has Lipschitz ratio $\alpha < 1$, but the local spacing function defined by a CFG formula has Lipschitz ratio equal to 1. Moreover, Lemma 5 shows that the filtering of a hierarchical-tree based oversampling for an α-Lipschitz function is an ls-spaced δ-sample, where $\delta = (2+\alpha)$. To apply Lemmas 1 and 7 we need to have $\alpha\delta < 1$, which implies that α should be less than $\sqrt{2}-1 \approx 0.4142$.

If the Lipschitz constant of ls is larger than $\sqrt{2}-1$, than we can use the following lemma to construct a Lipschitz function $g()$ of Lipschitz constant strictly less than $\sqrt{2}-1$ that is a point-wise approximation of ls().

Lemma 6 *If* ls() *is α-Lipschitz, then for any positive constant c, $c \cdot$ ls() is (αc)-Lipschitz.*

Proof. $|c \cdot \text{ls}(x) - c \cdot \text{ls}(y)| = c|\text{ls}(x) - \text{ls}(y)| \leq c\alpha(|x-y|)$. □

The following is our well-spaced point generation algorithm.

Algorithm *Well-Spaced-Point-Generation*(ls, Ω)

- Let α be the Lipschitz ratio of ls.
- Choose a β in the range $0 < \beta < \sqrt{2}-1$ and let $g() = \frac{\beta}{\alpha}\text{ls}()$.
- Apply the balanced hierarchical tree algorithm (Lemma 4) to approximate g.
- In each cell, place a random point set to generate a 1-sample with respect to g. Call this set P.
- Generate the conflict graph G_P and compute a maximal independent set S of G_P.
- Return S.

Figure 6 show an example of well-space point set.

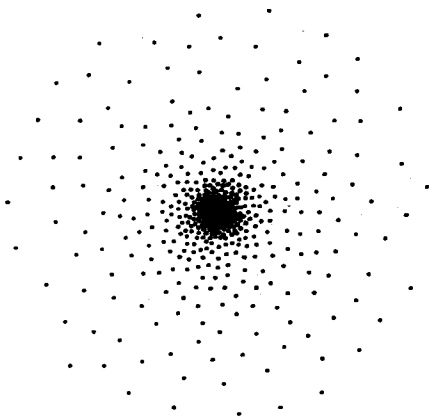

Fig. 6. A well-space point set after smoothing the point set given in Fig. 7 of Section 5.

Ruppert's result,[24] showing that all optimal meshes conform to the local spacing function, can be combined with results showing the balanced quad/oct-tree is an optimal mesh (see Bern, Eppstein and Gilbert[5] in 2D, and Mitchell and Vavasis[19] for 3D), to yield the following theorem:

Theorem 3 (BEG-MV-R) *Let* ls *be a local spacing function with Lipschitz ratio no more than* 1. *For all constants* $0 < \beta < 1$, *there is a constant* C *depending only on* β *and the dimension such that the size of the balanced hierarchical tree for* $g = \beta h$ *is no more than* C *time the size of the balanced hierarchical tree for* ls.

Theorem 1 then follows from Lemmas 5, 4, and 1, and Theorem 3.

We use a hierarchical tree as an initial approximation to the local spacing function ls and then perform oversampling and filtering to obtain a well-spaced point set. Notice, however, that the vertices of the hierarchical tree themselves are well-spaced and sampled with respect to ls. We do not use this set of vertices as our final point set because they are highly degenerated, containing many co-linear and co-planar points. In theory, our algorithm can be viewed as a smoothing algorithm which perturbs the balanced tree nodes. However, in practice we can use a much coarser hierarchical tree to approximate the feature function ls and apply random points in leaf-boxes to generated a 1-sample. By filtering, we get a well-spaced point set whose local spacing is closer to the best possible, compared with the hierarchical tree vertices which is experimentally shown to have high constants.[24]

4.3. Maximally Spaced Points and Aspect-Ratio

In this section, we relate the maximally ls-spaced point set with the following aspect-ratio condition presented in Ref. [18].

Definition 5 (Radius-edge aspect-ratio) *A point set* $P \in \mathbb{R}^d$ *has radius-edge aspect-ratio bounded by* $\rho \geq 1$ *if for each Delaunay simplex of the Delaunay diagram of* P, *the ratio of its circumscribed sphere radius to the smallest edge is bounded by* ρ.

In two dimensions, if the smallest angle of a triangle is θ, then the radius-edge aspect-ratio of the triangle is given by $1/(2\sin\theta)$. Hence, a triangle is well-shaped with respect to the radius-edge aspect-ratio "iff" it is well-shaped with respect to the standard aspect-ratio. In three or higher dimensions, if the standard aspect-ratio is bounded from above by a constant α, then the radius-edge aspect-ratio is bounded from above by a constant α' depending only on α and d, the dimension. However, the other direction is not true. For example, a three dimensional silver (a tetrahedron whose four vertices are formed by a small perturbation of one vertex of a square to a non-coplanar point on the sphere that subscribes the square by a great circle) has an radius-edge aspect-ratio close to $1/\sqrt{2}$, but its standard aspect-ratio is unbounded. Miller *et al*[18] have shown that meshes with bounded radius-edge aspect-ratio are useful for numerical simulation employing the control-volume formulation.

To make the paper self-contained, we briefly review the definition of Delaunay triangulations and Voronoi diagrams. Suppose $P = \{p_1, \ldots, p_n\}$ is a point set in

d dimensions. The convex hull of $d+1$ affinely independent points from P forms a ***Delaunay simplex*** if the circumscribed ball of the simplex contains no point from P in its interior. The union of all Delaunay simplices forms the ***Delaunay diagram***, $DT(P)$. If the set P is not degenerate then $DT(P)$ is a simplicial decomposition of the convex hull of P. Associated with $DT(P)$ is a collection of balls, called ***Delaunay balls***, one for each cell in $DT(P)$. The Delaunay ball circumscribes its cell. The geometric dual of the Delaunay Diagram is the ***Voronoi Diagram***. For more details, see Refs. [22,10].

Theorem 4 *Suppose* ls() *is α-Lipschitz with $\alpha < 1$. Then any maximally* ls*-spaced point set S is of radius-edge aspect-ratio bounded by* $1/(1-\alpha)$.

Proof. Let D be the Delaunay simplex of S. Let R be the radius of the Delaunay sphere of D. Let l be the length of the smallest edge of D. Therefore, one of the end points, say p, of the smallest edge has $\text{ls}(p) \leq l$. The value of the spacing function ls at the center of the sphere is therefore smaller than $l + \alpha R$ because we assume that ls is α-Lipschitz. Notice that there is no point of S in the interior of the Delaunay sphere of D. Thus, if $l + \alpha R < R$ then the center can be added to S, which contradicts maximality of S. Therefore, we have $R \leq l + \alpha R$, and hence $R/l \leq 1/(1-\alpha)$, implying that the radius-edge aspect-ratio of D is bounded by $1/(1-\alpha)$. □

We can extend Theorem 4 to ls-spaced δ-samples.

Lemma 7 *Suppose* ls *is α-Lipschitz with $\alpha < 1$ and let δ be a constant such that $\alpha\delta < 1$. Then any* ls*-spaced δ-sample S has radius-edge aspect-ratio bounded by $\delta/(1-\alpha\delta)$.*

Proof. Let D be the Delaunay simplex of S. Let R be the radius of the Delaunay sphere of D. Let l be the length of the smallest edge of D. Therefore, one of the end points, say p, of the smallest edge has $\text{ls}(p) \leq l$. The value of the spacing function ls at the center o of the sphere is therefore smaller than $l + \alpha R$ because we assume that ls is α-Lipschitz. Notice that there is no point of S in the interior of the Delaunay sphere of D. Thus, if $\delta(l + \alpha R) < R$ then the $\delta\text{ls}(o)$-ball centered at o does not contain any point from S, which contradicts the assumption that S is an ls-spaced δ-sample. Therefore, we have $R \leq \delta(l + \alpha R)$, and hence $R/l \leq \delta/(1-\alpha\delta)$, implying the radius-edge aspect-ratio of D is bounded by $\delta/(1-\alpha\delta)$. □

4.4. Coping with Boundary Effects

Notice that in the theorem and the lemma above we ignore boundary effects, i.e., we assumed that the Delaunay ball is fully contained in the domain. In order to account for boundary effects, the boundary edges will be given a g-spaced point set, for a spacing function g that is β-Lipschitz with $\beta = \alpha/c$ for a large enough constant. In effect, we first place points on the boundary edges (a lower dimensional process) with respect to g, so that points in the interior will be slightly repelled away from the boundary edges, and will not have too large Delaunay spheres that are mostly outside of the domain. A more rigorous treatment of the boundary, following the general ideas sketched here, can be found in Ref. [18].

4.5. Components of a Well-Spaced Point Set Generator

The well-spaced point generator has four major components: the CFG interface, and point, mesh, and sparse-matrix generators. Below, we discuss the functionalities of each component.

The **CFG Interface** provides a tool for a user to specify a Lipschitz distribution and the geometry of the domain. In two dimensions, the domain is defined using general planar straight-line graphs (see Bern-Eppstein-Gilbert[5] and Ruppert[24]). In three dimensions, the domain is defined by an polyhedron based constructive solid geometry expression (see Mitchell-Vavasis[19]).

A Lipschitz distribution is given in a CFG formula. Both the domain CSG and the CFG expression can be read from a file. A two dimensional graphic interface will allow the user to define a domain and its CFG formula adaptively. The three dimensional graphic-input-interface is less direct. It allows a user to choose and define basic feature functions, such as point features and features around geometric objects, by specifying the locations of points and the structures of geometric objects, and the feature parameter δ.

The **Point Generator** is built on three functions: the hierarchical-decomposition, the oversampling, and the filtering functions. We have already developed a 2D quad-tree based approximation, oversampling and filtering program. Vavasis's 3D balanced oct-tree based mesh software QMG[30] is used to approximate the local spacing function. Based on the oct-tree approximation, we use our program to perform oversampling and filtering.

The **Mesh Generator** is based on the Delaunay triangulation algorithm given in Miller-Talmor-Teng-Walkington.[18] This algorithm is designed for a well-spaced point set. It uses the geometric partitioner of Miller-Teng-Thurston-Vavasis.[17] The geometric partitioner code in our software is derived from a Matlab implementation by Gilbert and Teng.[13] Using the geometric partitioner to support a divide-and-conquer scheme, we reduce the Delaunay Triangulation problem for a well-spaced point set to a collection of convex-hull problems in the same dimension. Moreover in practice, the size of each convex-hull problem is bounded by 50 in the three dimensional triangulation. See Ref. [18] for more details.

The **Sparse-Matrix Generator** receives a mesh computed by the mesh generator and applies the finite element, finite difference, and/or finite control-volume formulations to construct a sparse matrix and a linear system.

5. The Smoothness of Galaxies: The Plummer Distribution

The *Plummer distribution* is used in astrophysics to model the distribution of star clusters in galaxy formation.[1] This distribution was found to be convenient for the comparison of numerical methods for the study of star cluster dynamics, and might be adopted as a standard model for such comparisons. In this section, we show that, interestingly, the Lipschitz constant of the spacing function of the Plummer distribution is bounded.

The Plummer distribution is a non-uniform distribution defined around a point.

The spacing density of the cluster at radius r away from the center is given by:

$$\rho(r) = \frac{3}{4\pi}([1+r^2])^{-5/2}.$$

The total mass within a sphere of radius r is then given by

$$M(r) = \int_{R=0}^{r} 4\pi r^2 \rho(R) dR = r^3(1+r^2)^{-3/2}.$$

Hence the total mass, when r approaches infinity is equal to 1. Assume that there is a total of N stars, each of mass $1/N$. Then the number of stars within a ball of radius r, denoted by $N(r)$, can be obtained by scaling the total mass formula by N:

$$N(r) = \frac{r^3 N}{(1+r^2)^{3/2}} = \frac{N}{(1+(1/r^2))^{3/2}}.$$

Figure 7 shows a point set generated by the Plummer distribution.

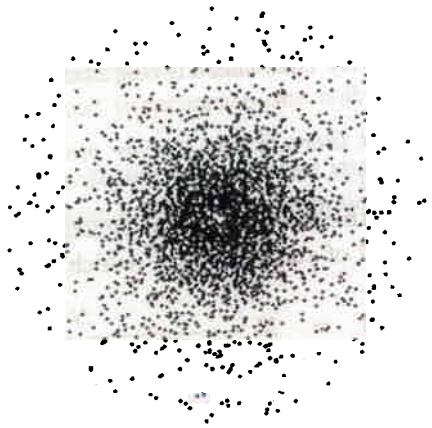

Fig. 7. A point set from the Plummer distribution.

In 3D the local spacing function is related to the density by $\mathrm{ls} = \rho^{-1/3}$, and therefore the local spacing function imposed by the Plummer distribution is

$$\mathrm{ls}(r) = N^{-1/3}(\frac{4\pi}{3})^{1/3}[1+r^2]^{5/6}.$$

We are interested in estimating the Lipschitz constant of the Plummer spacing function. The derivative of the Plummer spacing function is:

$$\mathrm{ls}'(r) = N^{-1/3}\frac{40\pi}{18}^{1/3}[1+r^2]^{-1/6}r.$$

To simplify our discussion, let K be the constant in the above equation. We have

$$\mathrm{ls}'(r) = \frac{Kr}{N^{1/3}[1+r^2]^{1/6}}.$$

The derivative $\mathrm{ls}'(r)$ of the Plummer spacing function is an increasing function. However the Lipschitz constant of $\mathrm{ls}(r)$ is unbounded when r is approaching the infinity. We now calculate the cut-off value when $\mathrm{ls}'(r) = 1$.

Lemma 8 *There is a constant K_1 such that $\mathrm{ls}'(r) \leq 1$ if $r \leq K_1\sqrt{N}$.*

Proof. In order to find the cutoff value of r, we solve the equation $\mathrm{ls}'(r) = 1$, i.e.,

$$\frac{Kr}{N^{1/3}[1+r^2]^{1/6}} = 1.$$

Using the Taylor expansion, we can find the constant K_1. An alternative way to find K_1 is to solve the above equation using $r = K_1\sqrt{N}$. We obtain

$$\frac{KK_1N^{1/2}}{N^{1/3}[1+K_1^2N]^{1/6}} = 1.$$

We can assume $N \gg 1$. Thus the left hand of the equation can be approximately reduced to $KK_1^{2/3}$. So $K_1 \approx 1/K^{2/3}$. □

We now bound the probability that there is one star or more outside of the ball of radius $K_1\sqrt{N}$.

Lemma 9 *The probability that there is one star or more outside the ball of radius $K_1\sqrt{N}$ is bounded by $\Theta(1/\sqrt{N})$.*

Proof. In order to prove the lemma, it is sufficient to calculate the expected number of stars outside the ball of radius $K_1\sqrt{N}$ which is given by

$$\begin{aligned} N - N(K_1\sqrt{N}) &= N - \frac{N}{(1+(1/(K_1\sqrt{N})^2))^{3/2}} \\ &= N - \frac{N}{(1+1/(K_1{}^2N))^{3/2}}. \end{aligned}$$

By the Taylor expansion, there is a constant K_2 such that $1/(1+x)^{3/2} \leq 1 - K_2x^{3/2}$, we have

$$N - \frac{N}{(1+1/K_1{}^2N)^{3/2}} \leq N - N(1 - K_2/(K_1^3N)^{3/2} = \Theta(1/\sqrt{N}).$$

Therefore, the probability that there is a star outside the ball of radius $K_1\sqrt{N}$ is bounded by $\Theta(1/\sqrt{N})$. □

In practical galaxy modeling, the radius of the ball chosen is much smaller than $O(\sqrt{N})$. The two lemmas above show, in practical numerical galaxy simulation, that the spacing function of the Plummer distribution has Lipschitz constant less than 1. Notice that the local feature at the center is of order $1/N^{1/3}$.

The Plummer feature function is an interesting variant of a point feature function where the local spacing at the point is of order $1/N^{1/3}$. Similar to the fact

that a point set in uniform distribution, though the underlying spacing function is Lipschitz with ratio 0, is not well-spaced, points of the Plummer distribution themselves are not well-spaced with respect to its spacing function; there could be clustering and gaps among the stars. But the Plummer distribution provides a natural distribution whose underlying spacing function is Lipschitz. We can make use of a super-imposing operation (given in Section 3) to model a universe that has more than one galaxy.

6. Final Remarks

In this paper, we characterize and present a constructive formulation for modeling non-uniform distributions that arise in scientific computing. We give an efficient algorithm for generating well-spaced point sets from these distributions. Our algorithms provide an automatic procedure for the generation of experimental data for Delaunay-based mesh generation algorithms. Our work complements the result of Bern, Eppstein, and Yao[7] that the uniform distribution fails to give good test data for this class of scientific applications.

Our result has potential impact on the design of geometric software and benchmarks. It provides numerical analysts a much needed automatic procedure for the generation of unstructured sparse matrices. We will continue our effort in implementing our algorithms and incorporate our software into numerical scientific simulator.

We hope that our work will help to raise the issue of careful theoretical treatment of characterizing problems from practical applications and generating relevant experimental data.

Acknowledgments

Gary Miller was supported in part by National Science Foundation grant CCR-91-96113. The work of Dafna Talmor was done while at School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, USA. Shang-Hua Teng was supported in part by an NSF CAREER award (CCR-9502540) and an Alfred P. Sloan Research Fellowship. Part of this work was done while at Department of Computer Science, University of Minnesota, Minneapolis, Minnesota 55455.

References

1. S.J. Aarseth and M. Henon and R. Wielen. A Comparison of Numerical Methods for the study of star cluster dynamics. *Astronomy and Astrophysics*, 37, 183–187, 1974.
2. I. Babuška and A.K. Aziz. On the angle condition in the finite element method. *SIAM J. Numer. Anal.*, 13(2):214–226, 1976.
3. J. L. Bentley and M. I. Shamos. Divide and conquer for linear expected time, In *Information Processing Letter*, 7 87–91, 1978.
4. J. L. Bentley, B. W. Weide and A. C. Yao. Optimal expected-time algorithms for closest point problems. In *ACM Trans. on Mathematical Software*, 6(4) 563–580, 1980.

5. M. Bern, D. Eppstein, and J. R. Gilbert. Provably good mesh generation. In *FOCS90*, 231–241, 1990. IEEE.
6. M. Bern, D. Eppstein, and S.-H. Teng. Parallel construction of quad-trees and quality triangulations. In *Workshop on Algorithms and Data Structures, Springer LNCS 709*, pages 188–199, 1993.
7. M. Bern, D. Eppstein, and F. F. Yao. The expected extremes in a Delaunay triangulation. *Inter. J. of Computational Geometry and Appl.*, 1(1):79–91, 1991.
8. K. Q. Brown. Geometric transforms for fast geometric algorithms. Technical Report CMU-CS-80-101, CMU, 1980. PhD Thesis.
9. R. A. Dwyer. Higher-dimensional Voronoi diagrams in linear expected time. In *Proc. 5th Annual ACM Sym. on Computational Geometry*, 326–333, 1989.
10. H. Edelsbrunner. *Algorithms in Combinatorial Geometry*, volume 10 of *EATCS Monographs on Theoretical CS*. Springer-Verlag, 1987.
11. H. Edelsbrunner and R. Seidel. Voronoi diagrams and arrangements. *Discrete and Computational Geometry*, 1:25–44, 1986.
12. D. Eppstein, G. L. Miller, and S.-H. Teng. A deterministic linear time algorithm for geometric separators and its applications. In *Proceedings of 9th ACM Symposium on Computational Geometry*, pages 99–108, 1993.
13. J. R. Gilbert, G. L. Miller, and S.-H. Teng. Geometric mesh partitioning: Implementation and experiments. In *International Conference of Parallel Processing*, pp 418–427 1995.
14. R. M. Karp and J. M. Steele. Probabilistic analysis of heuristics. In Lawler, Lenstra, Rinnooy Kan, and Shmoys, editors, *The Traveling SAlesman Problem: A Guided Tour of Combinatorial Optimization*. Wiley, N.Y., 1985.
15. R. E. Miles. On the homogeneous planar Poisson point process. *Mathematical Biosciences*, 6:85–127, 1970.
16. G. L Miller and D. Talmor. Mesh generation using well spaced points. Manuscript, 1995.
17. G. L. Miller, S.-H. Teng, W. Thurston, and S. A. Vavasis. Automatic mesh partitioning. In A. George, J. Gilbert, and J. Liu, editors, *Sparse Matrix Computations: Graph Theory Issues and Algorithms*, IMA Volumes in Mathematics and its Applications. Springer-Verlag, 1993.
18. G. L. Miller, D. Talmor, S.-H. Teng, and N. Walkington. A Delaunay Based Numerical Method for Three Dimensions: generation, formulation, partition. In *the proceedings of the twenty-sixth annual ACM symposium on the theory of computing*, 47–61, 1995.
19. S. A. Mitchell and S. A. Vavasis. Quality mesh generation in three dimensions. In *Proceedings of the ACM Computational Geometry Conference*, 212–221, 1992.
20. S. A. Mitchell. Cardinality bounds for triangulations with bounded minimum angle. In *Proc. 6th Canadian Conference on Computational Geometry*, 326–331, 1994.
21. Omar Ghattas and Xiaogang Li. Private communications, 1996.
22. F. P. Preparata and M. I. Shamos. *Computational Geometry An Introduction*. Texts and Monographs in Computer Science. Springer-Verlag, 1985.
23. A. A. G. Requicha. Representations of rigid solids: theory, methods, and systems. In *ACM Computing Survey*, 12, 437–464, 1980.
24. J. Ruppert. A new and simple algorithm for quality 2-dimensional mesh generation. In *Third Annual ACM-SIAM Symposium on Discrete Algorithms*, 83–92, 1992.

25. H. Samet. The quadtree and related hierarchical data structures. *ACM Computing Surveys*, pages 188–260, 1984.
26. G. Strang and G. J. Fix. *An Analysis of the Finite Element Method.* Prentice-Hall, Englewood Cliffs, New Jersey, 1973.
27. P. Su. Efficient parallel algorithms for closest point problems. Technical Report PCS-TR94-238, Dartmouth College, NH, 1994. PhD Thesis.
28. S.-H. Teng. *Points, Spheres, and Separators: a unified geometric approach to graph partitioning.* PhD thesis, Carnegie-Mellon University, School of Computer Science, 1991. CMU-CS-91-184.
29. Y. A. Teng, F. Sullivan, I. Beichl, and E. Puppo. A data-parallel algorithm for three-dimensional Delaunay triangulation and its implementation. In *SuperComputing 93*, pages 112–121. ACM, 1993.
30. `http://simon.cs.cornell.edu/home/vavasis/vavasis.html`.
31. F. F. Yao. Computational Geometry. In *Algorithms in Complexity*, R. A. Earnshaw and B. Wyvel editors, 345–490, Elsevier, Amsterdam, 1990.