No Quadrangulation is Extremely Odd

Prosenjit Bose* Godfried Toussaint†

Abstract

Given a set S of n points in the plane, a quadrangulation of S is a planar subdivision whose vertices are the points of S, whose outer face is the convex hull of S, and every face of the subdivision (except possibly the outer face) is a quadrilateral. We show that S admits a quadrangulation if and only if S does not have an odd number of extreme points. If S admits a quadrangulation, we present an algorithm that computes a quadrangulation of S in $O(n \log n)$ time even in the presence of collinear points. If S does not admit a quadrangulation, then our algorithm can quadrangulate S with the addition of one extra point, which is optimal. We also provide an $\Omega(n \log n)$ time lower bound for the problem. Finally, our results imply that a k-angulation of a set of points can be achieved with the addition of at most $k-3$ extra points within the same time bound.

1 Introduction

Given a set of points in the plane, the problem of determining whether the set admits a certain combinatorial structure has received considerable attention. Of particular importance is the study of triangulations of point sets due to its many applications in graphics, medical imaging, Geographic Information Systems (GIS), finite element methods, statistics, scattered data interpolation, and pattern recognition to name a few ([1], [3], [24], [31], [35], [37], [38], [40]). However, in the study of *finite element methods* and *scattered data interpolation*, it has recently been shown that quadrangulations may be more desirable objects than triangulations. A quadrangulation of a set of points S is a planar subdivision whose vertices are the points of S, whose outer face is the convex hull of S, and every face of the subdivision (except possibly the outer face) is a quadrilateral.

A fundamental component of finite element methods is the automatic and semi-automatic generation of meshes for finite element analysis (see Ho-Le [15] for a survey of the area). The problem of converting a triangular mesh to a mesh consisting of quadrilaterals (quadrangular mesh) has been studied by Heighway [13] and Johnston et al. [16]. Johnston et al. integrate several heuristics into a system that automatically converts a triangular mesh into a quadrangular mesh which runs in $O(n^2)$ time and adds up to $O(n)$ extra points to complete the quadrangulation where n is the size of the input. Such extra points are often referred to as Steiner points. deBerg [9] has devised a very simple algorithm that converts a triangular mesh to a quadrangular mesh with the addition of $O(n)$ Steiner points (refer to Figure 1). Each triangle of the triangulation is decomposed into three quadrangles (quadrilaterals) using the following approach: place a Steiner point at the midpoint of each edge of the triangulation as well as in the center of each triangle of the triangulation. Then complete the quadrangulation by connecting the Steiner point in the center of each triangle to the three Steiner points on its edges.

*Research supported by a KILLAM and NSERC postdoctorate fellowship. Address: Department of Computer Science, University of British Columbia, 201 - 2366 Main Mall, Vancouver, British Columbia, V6T 1Z4. E-mail: jit@cs.ubc.ca.

†Research supported by grants NSERC-OGP0009293 and FCAR-93ER0291. Address: School of Computer Science, McGill University, 3480 University, Montréal, Québec, H3A 2A7. E-mail: godfried@cs.mcgill.ca.

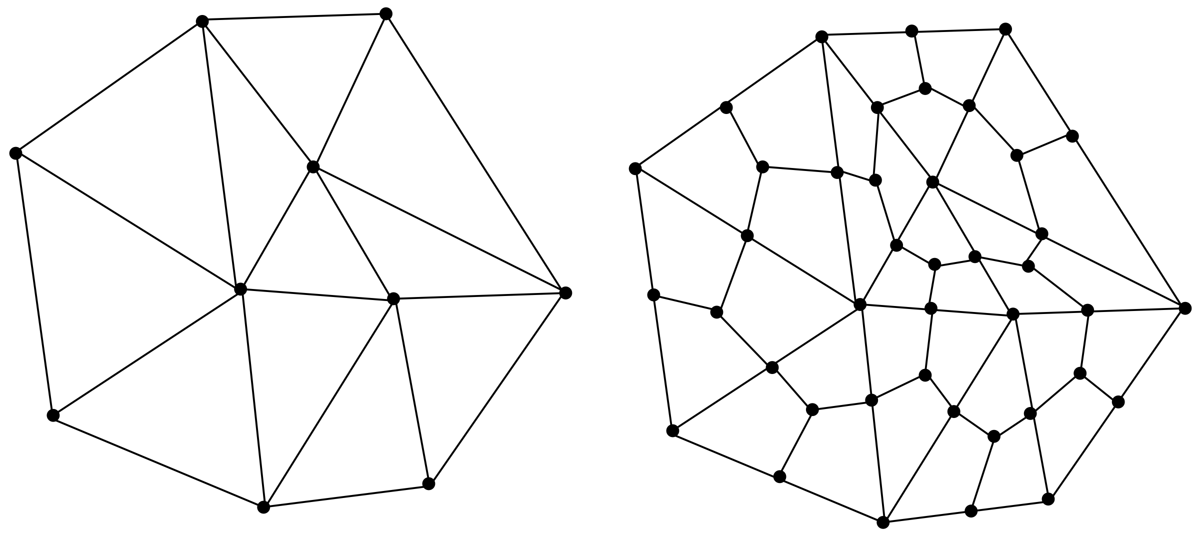

Figure 1: Example of deBerg's construction

The classical problem in scattered bivariate data interpolation can be stated as follows. Given a set $\mathcal{V} = (v_1, v_2, \ldots, v_n)$ of n points in the plane along with an elevation z_i, $i = 1, 2, \ldots, n$ associated with each point v_i, determine a function f such that $f(v_i) = z_i$, for all i. There is a large body of literature on this subject describing a variety of methods that yield functions with different properties. Most of these methods start with a triangulation of $\mathcal{V}$ which is subsequently refined in some way [37]. Since a triangulation of a point set always exists there is no problem starting in this way. Recently Lai [19] and Lai & Schumaker [20] showed that there are certain definite advantages that can be obtained by starting not with a triangulation but with a quadrangulation. They also give algorithms for computing such interpolation functions starting from quadrangulations. However, as they point out [20], a set of points does not always admit a quadrangulation. Nevertheless, they mention that it is always possible to start with a triangulation, and by adding a few extra points to $\mathcal{V}$, they can delete certain edges from the triangulation to obtain a quadrangulation.

The above two applications provide new motivation for the study of quadrangulations of point sets from the computational geometry point of view. We remark that quadrangulations of polygons have been investigated in the computational geometry literature for some time in the context of guarding or illumination problems. One of the earliest results on quadrangulations concerns orthogonal polygons, i.e., polygons whose sides are parallel to two orthogonal axes. The motivation here was not the quadrangulation itself but its exploitation as a tool for locating $n/4$ guards (lights) to cover (illuminate) the interior of the polygon. Sack and Toussaint [29] showed that a star-shaped orthogonal polygon of n vertices can always be decomposed into convex quadrangles in $O(n)$ time. Kahn, Klawe and Kleitman [17] proved this result for arbitrary simple orthogonal polygons. However, their existential proof did not lead to an algorithm for quadrangulating the polygon. Later, both Sack and Toussaint [30] and Lubiw [22] independently obtained constructive proofs of the Kahn, Klawe and Kleitman [17] result that led to an $O(n \log n)$ time algorithm. Lubiw [22] also showed under a fairly general computational model that $\Omega(n \log n)$ time is a lower bound on the complexity of this problem. Some work has also been done on computing minimum *ink* quadrangulations of simple orthogonal polygons. In this setting one is interested in the convex quadrangulation whose total length (sum of diagonal lengths) is a minimum. Independently, Keil & Sack [18] and Lubiw [22] showed this could be done in $O(n^4)$ time and $O(n^2)$ space. A very special case of convex quadrangulations, of great interest in VLSI, is that of decomposing an orthogonal polygon into rectangles. For example, Lipski et al. [21] and Ohtsuki [23] give polynomial time algorithms for partitioning orthogonal polygonal regions into the minimum number of rectangles.

Whereas an orthogonal polygon always admits a convex quadrangulation, an arbitrary simple polygon does not necessarily admit a quadrangulation, even if convexity of the resulting quadrangles is not a requirement. In fact, Lubiw [22] has shown that the problem of deciding whether a polygon with holes admits a quadrangulation is NP-complete. On the other hand, if we are allowed to add new points that act as vertices (usually called Steiner points) then, depending on how many such points we are allowed to add, we may always obtain convex quadrangulations. However, it is not difficult to construct polygons that require $\Omega(n)$ Steiner points in order to complete a quadrangulation. Inspired by deBerg's method, Everett et al. [10] have shown that a simple polygon can always be quadrangulated into $5(n-2)/3$ strictly convex quadrangles and that $n-2$ quadrangles are sometimes necessary. For polygons with holes, they showed that a polygon on n vertices with h holes can always be decomposed into $8(n+2h-2)/3$ strictly convex quadrangles.

Some work has also been done on computing minimum *ink* quadrangulations of simple polygons. Conn & O'Rourke [8] showed this could be done in $O(n^3 \log n)$ time and $O(n^3)$ space. To close this section we mention a special case of convex quadrangulations of polygons that has also been studied before and this concerns the problem of decomposing a polygon into trapezoids. In fact, many polygon triangulation algorithms start out by first obtaining a trapezoidization of the polygon and subsequently converting this trapezoidization into a triangulation [12]. Here the trapezoidization is merely a step towards another goal. Furthermore, we should point out that in the work on trapezoids, triangles are allowed and considered as degenerate trapezoids. On the other hand, trapezoidizations are used in the manufacturing industry as the main goal in electron beam lithography systems [33] where subsequent processing time is proportional to the number of trapezoids in the decomposition. Asano, Asano and Imai [2] have shown that a partition of a polygonal region into the minimum number of trapezoids can be obtained in $O(n^2)$ time.

In this paper we characterize those sets of points that admit a quadrangulation. We show that S admits a quadrangulation if and only if S does not have an odd number of extreme points. If S admits a quadrangulation, we present an algorithm that computes a quadrangulation of S in $O(n \log n)$ time, even in the presence of collinear points. If S does not admit a quadrangulation, then our algorithm can quadrangulate S with the addition of one extra point, which is optimal. Our algorithm is conceptually simple, but to achieve the $O(n \log n)$ time complexity, we need to use some complicated data structures. However, from the conceptual description of the algorithm, a very simple $O(n^2)$ time algorithm is implied which may be more desirable from a practical point of view. We also provide an $\Omega(n \log n)$ time lower bound for the problem. Finally, our results imply that a set of points S admits a k-angulation for any k with the addition of at most $k-3$ Steiner points. A k-angulation of a set of points S is a planar subdivision whose vertices are the points of S, whose outer face is the convex hull of S, and every face of the subdivision (except possibly the outer face) is a simple polygon with k vertices.

2 Preliminaries

Let us first introduce some terminology. Most of the geometric and graph theoretic terminology used is standard and for details, we refer the reader to O'Rourke [25], Bondy and Murty [5], and Preparata and Shamos [27]. We begin by reviewing some of the main geometric and graph theoretic terminology used in this paper.

Let E^2 denote the Euclidean space of dimension two. A domain D in E^2 is *convex* if, for any two points q_1 and q_2 in D, the segment $[q_1 q_2]$ is entirely contained in D. The *Convex Hull* of a set of points S in E^2 is the boundary of the minimum area convex domain in E^2 containing S. A set of points S is said to be in *general position* provided that no three points in S are collinear.

A *graph* $G = (V(G), E(G))$ consists of a finite nonempty set $V(G)$ of *vertices*, and a set $E(G)$ of unordered pairs of vertices known as *edges*. A planar graph is a graph G that can be embedded in the plane such that a vertex of G is represented by a point in the plane, an edge of G is represented by a simple curve between the two vertices, and no two edges of G intersect except at vertices. It is well-know that any planar graph admits a planar embedding where all edges are mapped to straight line segments (Fary [11], Stein [32], Wagner [36]), such an embedding is referred to as a *planar subdivision* or *planar map*. A planar subdivision consists of internal faces and a single external face. An *internal face* of a planar subdivision is a bounded region of the embedding and the *external face* is the single unbounded region.

3 Existence of Quadrangulation

In this section, we characterize the sets of points that admit a quadrangulation. The characterization is surprisingly simple. We first describe a necessary condition for a point set to admit a quadrangulation.

Lemma 3.1 *If a set of points S admits a quadrangulation then S has an even number of points on its convex hull.*

Proof: We use a counting argument to prove the lemma. Let $Q(S)$ represent a quadrangulation of S. Let $E(S)$ be the number of edges, $V(S)$ the number of vertices and $F(S)$ the number of faces of $Q(S)$.

Let CH represent the number of vertices on the outer face of $Q(S)$. Let $F_I(S)$ represent the number of internal faces of $Q(S)$. Since every internal face is a quadrilateral, we have that $4F_I(S) + CH = 2E(S)$ (see [5] for a detailed proof of this fact). Therefore, $CH = 2(E(S) - 2F_I(S))$. Since $2(E(S) - 2F_I(S))$ is an even number, CH must be even for the relation to hold. Therefore, we conclude that if S admits a quadrangulation then the convex hull of S must contain an even number of extreme points. ■

We now show that the above necessary condition is also sufficient. In order to simplify the discussion, we will first prove sufficiency under the condition that no three points are collinear in the point set S. We remove this condition in the following section.

Lemma 3.2 *If S has an even number of points on its convex hull then S admits a quadrangulation.*

Proof: We proceed by induction on the number of points inside the convex hull of S, $CH(S)$.
Basis: Let S be a set of points with an even number of points on $CH(S)$ and no points inside $CH(S)$. Let $1, 2, \ldots, n$ represent the n points as they occur in counter-clockwise order on the convex hull of S. We construct a quadrangulation of S in the following manner (see Figure 2).

Join i to $i + 1$ for all $i = 1, 2 \ldots, n$. Join n to 1. So far we have constructed a convex polygon. Finally, join 1 to j for all j satisfying the following: $j = 1, \ldots, n - 2$, $j \neq 2$ and $j = 0 \bmod 2$. The result is a quadrangulation of S.
Inductive Hypothesis: A set of points S with an even number of points on $CH(S)$ and k points inside $CH(S)$ where $k \geq 0$ and $k \leq m$ for a fixed constant m admits a quadrangulation.
Inductive Step: Let S be a set of points with an even number of points on $CH(S)$ and $m + 1$ points inside $CH(S)$. We now must show that S admits a quadrangulation.

Since $m + 1 \geq 1$, S must have at least one point inside $CH(S)$. Let q be such a point. Let S' be the set of points S with q removed. By construction, $CH(S)$ is the same as $CH(S')$, which means that S' has an even number of vertices on the convex hull and at most m points inside its convex hull. Therefore, by the inductive hypothesis, S' admits a quadrangulation, denoted by $Q(S')$. If we re-insert q, then q must be contained in some quadrilateral of $Q(S')$, say D. The point q cannot lie on an edge of $Q(S')$ since we assumed that no three points in S are collinear. Let $D = a, b, c, d$ be the four vertices of the quadrilateral containing q. See Figure 3 for the following two cases. If D is convex, then we can adjust $Q(S')$ to be a quadrangulation of S by adding the edge $[qa]$ and $[qc]$. If D is not convex, without loss of generality, let

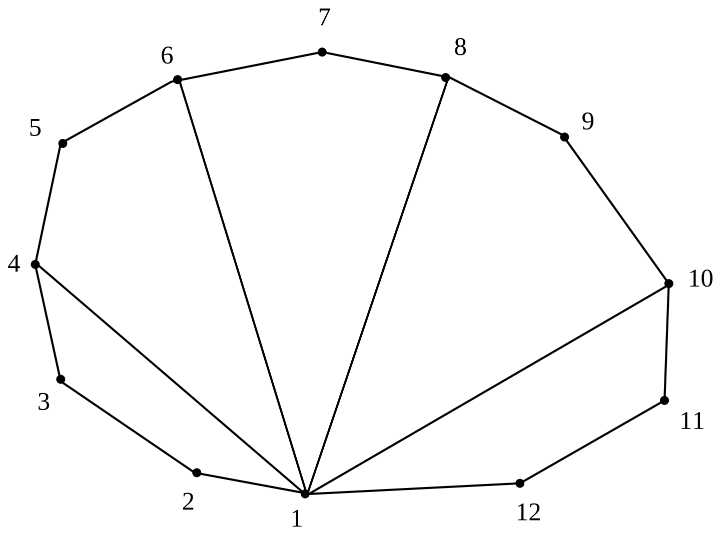

Figure 2: Quadrangulation of a point set with no points in the interior of the convex hull.

us assume that c is a reflex vertex. Again, we can adjust $Q(S')$ to be a quadrangulation of S by adding the edge $[qa]$ and $[qc]$. ■

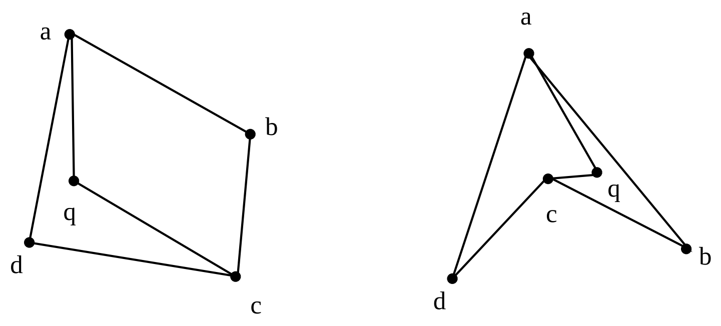

Figure 3: Re-adjustment of $Q(S')$ to achieve $Q(S)$.

We conclude with the following theorem:

Theorem 3.1 *A set of points S admits a quadrangulation if and only if the convex hull of S has an even number of points.*

4 Algorithms

The existence proof in the previous section (Lemma 3.2) immediately implies the following Sequential Insertion (SI) Algorithm. Compute the convex hull of S in $O(n \log n)$ time with any of several algorithms available (see [25]). If the number of convex hull vertices is even, partition the convex hull into quadrilaterals in $O(n)$ time by joining one vertex to every other vertex of the polygon in a clockwise fashion as shown in Figure 2. Subsequently, the remaining points are inserted one at a time. The insertion stage of the algorithm is the crucial part of the entire procedure. The obvious method of inserting the points results in an $O(n^2)$ time algorithm.

However, a variety of algorithms are available in the literature for point insertion [4]. The main difficulty in obtaining efficient implementations of the SI algorithm is determined by the fact that the data structure that supports fast point location must be dynamic so that subsequent point locations after an insertion remain efficient. If we maintain the quadrangulation as a triangulation we can exploit the additional structure over arbitrary planar subdivisions. Furthermore in our context we do not need fully dynamic algorithms since we are not interested in deletions. If we demand $O(n)$ storage space then the best relevant algorithms available, from the worst-case complexity point of view, are the algorithms of Preparata & Tamassia [28] and Cheng & Janardan [7]. The former algorithm has both a point location query time and an insertion (update) time of $O(\log^2 n)$. The latter algorithm is slightly better because it has a point location query time of $O(\log^2 n)$ but only an $O(\log n)$ insertion time. However, in our context every point location query is followed by an insertion and hence the worst-case complexities of both algorithms are the same. Therefore the Sequential Insertion (SI) Algorithm for quadrangulating S runs in worst-case time $O(n \log^2 n)$ and uses $O(n)$ storage space when implemented in this manner. This approach yields a rather complicated algorithm which also has $O(n \log^2 n)$ expected time complexity. However, by embedding the quadrangulation in a triangulation (as above) but using a very simple randomized triangulation algorithm [1], we can obtain a very simple quadrangulation algorithm with $O(n^2)$ worst-case and $O(n \log n)$ expected time complexities.

Another implementation of the SI Algorithm is to use a sweep-line approach (see [25] or [27] for examples of sweep-line algorithms). Once the convex hull has been partitioned into quadrilaterals (say $Q_1, \ldots, Q_j$), determine into which quadrilaterals Q_i, the remaining points fall. A simple sweep-line can be used to quadrangulate the set of points lying inside each Q_i, $i = 1, \ldots, j$ in $O(n_i \log n_i)$ time where n_i is the number of points inside Q_i. Therefore the total complexity of the SI algorithm implemented in this way is $O(n \log n)$.

Theorem 4.1 *Let S be a set of n points in general position in the plane. If n is even, then S can be quadrangulated in $O(n \log n)$ time, using the SI algorithm.*

From a theoretical computational complexity point of view, the SI algorithm is optimal. However it has its drawbacks. For one thing, the SI algorithm with random insertion of the points, fails if the points are not in general position, although Joe Mitchell has shown that inserting the points using a sweep-line approach, allows a non-trivial modification of the algorithm that can handle collinearities and still run in $O(n \log n)$ time. The main drawback, however, of the sequential insertion algorithm, as Figure 4 illustrates, is that it yields quadrangulations that are not desirable in practice since they tend to yield long non-convex quadrangles when fat convex quadrangles are preferred. Therefore we omit the implementation details of the SI algorithm and instead describe another algorithm below which has much greater promise of yielding nice quadrangulations in practice and can handle collinearities with little effort.

Let $P = (p_1, p_2, ..., p_n)$ denote a simple polygonal chain spanning the set S of n planar points given. A triangulation of P, denoted by $T(P)$, is a triangulation of S, $T(S)$, such that it contains all the edges of P as a subset of the edges of $T(S)$. The edges in $T(P)$, other than the edges of P, are called the *diagonals* of $T(P)$. The dual graph of $T(P)$ is a graph $G(V, E)$ whose vertex set V corresponds to the set of triangles in $T(P)$ and two vertices are connected with an edge if, and only if, their corresponding triangles share a diagonal. A triangulation of P is called *serpentine* if its dual graph is a chain.

Our approach will be to show that a set of points S always admits a simple spanning polygonal chain (whose vertex set is precisely the set of points in S) that in turn always admits a serpentine triangulation. In such a triangulation the diagonals (and hence triangles) are ordered in accordance to the ordering of the vertices of the dual chain. By *removing* every other diagonal starting at one end of the chain, we will

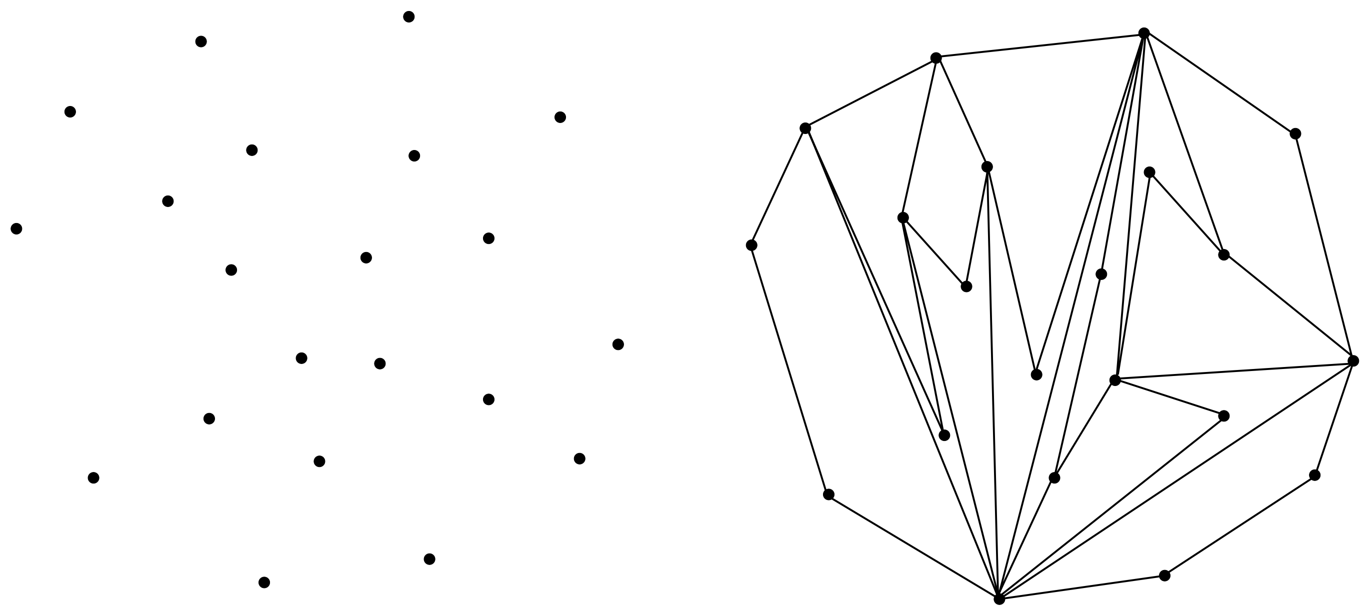

Figure 4: A set of points S and the quadrangulation obtained with the Sequential Insertion Algorithm

obtain the desired quadrangulation except perhaps a single triangle at the end. In such an eventuality we add one Steiner point outside the convex hull of S to convert the final triangle to a quadrangle. We then show that the Steiner point is necessary if, and only if, the number of points h on the convex hull of S is odd. We will first discuss this approach when the given points are in general position and then we will show the minor modifications that can be made in order to handle point sets with collinearities.

We should point out that in a different context concerned with fast rendering in computer graphics, Arkin et al. [1] proposed an $O(n \log n)$ time algorithm for obtaining a serpentine triangulation (Hamiltonian in their terminology) of a set of points. However, their algorithm is based on sequentially inserting triangles in a triangulation of the points and hence, like our SI algorithm, its application to our problem leads to very poor quadrangles similar to those in Figure 4. The serpentine triangulation algorithm that we propose yields very nice and usually convex quadrilaterals.

A simple spanning polygonal chain most convenient for our purpose is the convex spiral of S. We define the convex spiral of S constructively as follows. Let $p_{x-\min}$ denote the point of S with the minimum x-coordinate. If more than one point satisfies this property select the point which also has minimum y-coordinate. Construct an infinite directed half-ray anchored at $p_{x-\min}$ and pointing in the $+y$ direction. Mark the point $p_{x-\min}$. This marked point is the first vertex of the spiral chain. Now rotate the half-ray in a clockwise direction until it coincides with an unmarked point of S. This point is marked, it becomes the second vertex of the spiral chain and the ray is translated in the direction it is pointing so that it is anchored at the newly marked vertex just found. Continue this process until no unmarked points remain in S. We will refer to this procedure as the *Spiraling Procedure*. For the set of points in Figure 4 the resulting convex spiral is illustrated in Figure 5. This structure has appeared in the literature before and is closely related to the *onion peeling* of a set or the convex layers [6], [35], [39]. In fact one can compute the spiral and the convex layers, one from the other, in $O(n)$ time [27]. This structure has also been used for example in statistics to define a generalization of the concept of the median to two-dimensional data. In this application the median is defined as the last vertex of the spiral chain. We now show that the convex spiral of a set S admits a serpentine triangulation. Refer to Figure 6.

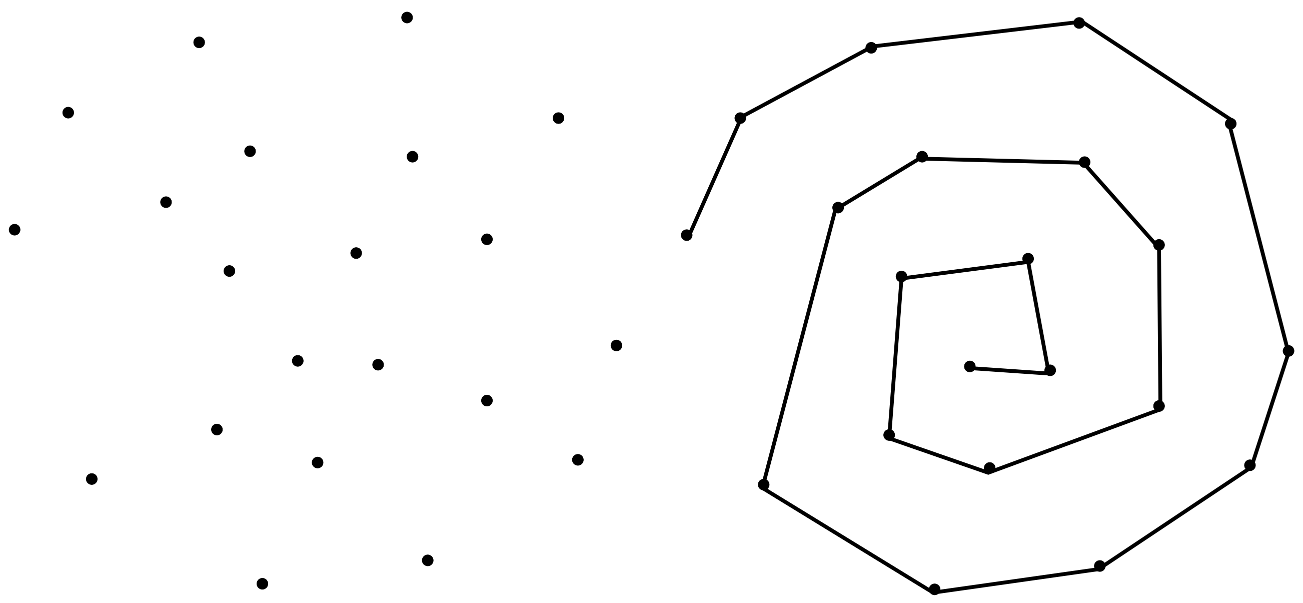

Figure 5: A set of points S and the convex spiral of S.

Lemma 4.1 *Let S be a set of n points in the plane. The convex spiral of S admits a serpentine triangulation.*

Proof: If all the points of S lie on their convex hull then the required triangulation can be obtained trivially by connecting any vertex of the convex hull to all the others. Therefore we assume there are points of S in the interior of the convex hull. To simplify the proof we divide the spiral region to be triangulated into two regions: an outer (spiral) region and an inner (star-shaped) region. We then show that each region admits a serpentine triangulation in such a way that these can be concatenated into one final serpentine triangulation.

Without loss of generality, assume that the points in S are re-labelled so that they are ordered in accordance with the convex spiral, i.e., $p_1 = p_{x-\min}$, p_2 is the next vertex of the spiral and so on until the last vertex of the spiral is labelled p_n. Since S has h vertices on the convex hull there exists one edge of the convex hull, namely $[p_1, p_h]$, which is a diagonal in every triangulation of the convex spiral P. The union of this edge $[p_1, p_h]$ with P partitions its complement, i.e., that part of the plane (E^2) excluding P and $[p_1, p_h]$, into two connected regions, one unbounded (exterior to the convex hull) and one bounded (the spiral region). Now extend the edge $[p_{n-1}, p_n]$ of P in the direction of p_n and let X be the first intersection point of this extension with P. At this point X, construct a line locally tangent to P. Now rotate this line in a counter-clockwise direction until it meets the first vertex of P, say Y, such that the line is parallel to the edge $[p_{n-1}, p_n]$. Finally, insert the diagonal $[p_n, Y]$. This diagonal partitions the spiral polygonal region into two regions: the outer spiral polygonal region $P_o = [p_1, p_2, \ldots, Y, p_n, p_{n-1}, \ldots, p_h]$ and the inner polygonal region $P_i = [p_n, p_{n-1}, ..., Y]$. By this construction, P_i is star-shaped from p_n. Therefore we can obtain a serpentine triangulation of P_i by simply inserting diagonals between p_n and all vertices of P_i other than p_{n-1} and Y.

It remains to triangulate the spiral polygonal region P_o. This can be accomplished with a variant of the *rotating calipers* [34]. The region P_o can be viewed locally as a convex polygonal annulus with outer chain $C_o = [p_1, p_2, \ldots, Y]$ and inner convex chain $C_i = [p_h, p_{h+1}, ..., Y, ..., p_{n-1}, p_n]$. We place one supporting line of the *rotating calipers* tangent to C_o and the other parallel to it and tangent to C_i. Note that a portion of

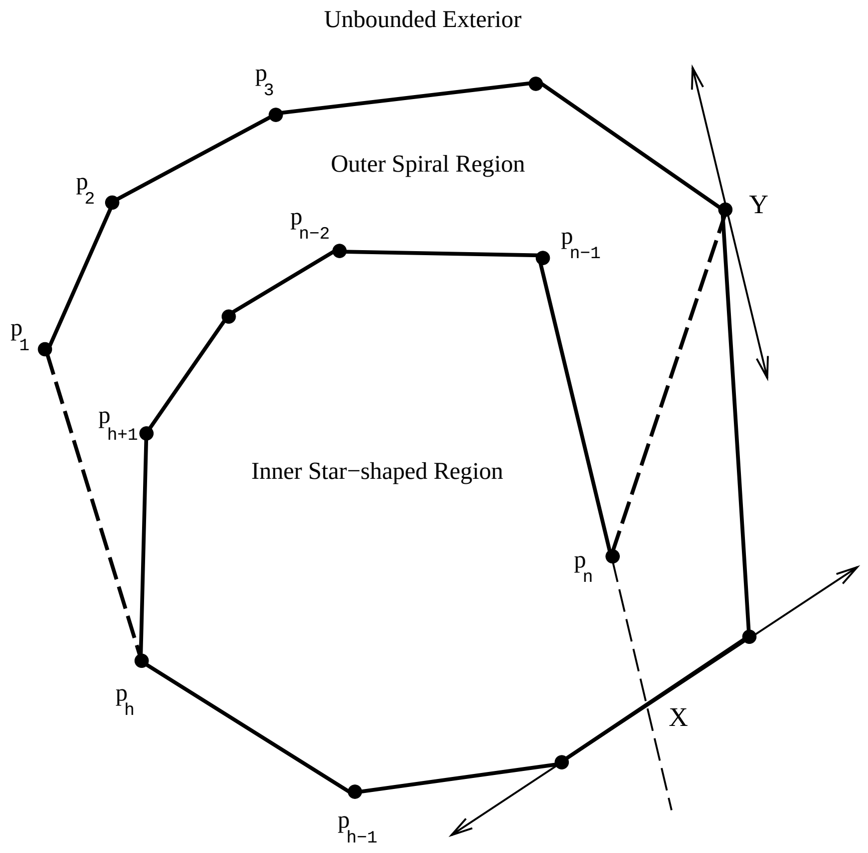

Figure 6: Illustrating the partition of the region inside the convex hull of S into the *outer* spiral region and the *inner* star-shaped region.

P may be common to both C_o and C_i when P has high enough winding number (as is the case in Figure 5). If we make two copies of this common portion we can view C_o and C_i as separate polygonal chains. To initialize the calipers we locate the first through p_1 and the second through p_h. At the start of the rotation both lines are set collinear with $[p_1, p_h]$. Next, we *rotate* the calipers in a clockwise direction until one of the lines meets a new vertex. This vertex will be either p_2 or p_{h+1} - depending on which has the smaller angle. This identifies a new co-podal pair of vertices between the chains C_o and C_i and this pair in turn determines the new diagonal to be inserted between them in the triangulation. This process is continued until the inner caliper line contains $[p_{n-1}, p_n]$. Since at each step one vertex on one chain is advanced, the insertion of the new diagonal creates one triangle in the triangulation. Since by this construction diagonals can never be inserted between two vertices of the same chain, we obtain a serpentine triangulation of the region P_o. Since the final triangle of the triangulation of P_o is adjacent to (shares a diagonal with) the first triangle of the triangulation of P_i, it follows that their concatenation is also serpentine. ■

By Lemma 4.1 the Spiraling Rotating Caliper (SRC) Algorithm produces a serpentine triangulation. Therefore the diagonals and triangles of $T(P)$ are ordered according to the order of the vertices comprising the dual chain of $T(P)$. Let $D = (d_1, d_2, \ldots, d_m)$ denote the diagonals in this order, where $d_1 = [p_1, p_h]$. If m is even, then if we delete from the triangulation every other diagonal $d_2, d_4, \ldots$ and so on, we obtain a quadrangulation. If m is odd we may delete every other diagonal starting from the last diagonal d_m. This will quadrangulate P except for the presence of triangle $\Delta(p_1, p_{h+1}, p_h)$ or $\Delta(p_1, p_2, p_h)$. Finally, by inserting one Steiner point just outside the convex hull of S near the edge $[p_1, p_h]$ we may convert this triangle to a quadrangle.

Consider now the complexity of the SRC algorithm. The first step of the algorithm is to compute the convex spiral of the given set of points. If the spiraling procedure, as described in the paragraph above the statement of Lemma 4.1, is used, then $O(n^2)$ time is needed for this step. However, we may compute the convex layers of S in $O(n \log n)$ time using the algorithm of Chazelle [6] or the algorithm of Hershberger and Suri [14]. This is the most difficult step in the algorithm, as both these algorithms are fairly involved. From an implementation point of view, it might be preferable to use the simpler algorithm. From the convex layers, we can compute a convex spiral P of S in $O(n)$ time with the procedure of Preparata and Shamos [27]. The spiraling rotating caliper algorithm for obtaining a serpentine triangulation of P described in the proof of Lemma 4.1 runs in $O(n)$ time since no backtracking is involved. Finally, triangulating the star-shaped interior, deleting the unwanted diagonals and inserting the Steiner point all require no more than $O(n)$ time. Therefore the entire algorithm runs in $O(n \log n)$ time. We conclude with the following theorem.

Theorem 4.2 *Let S be a set of n points in general position in the plane. Then S may be quadrangulated with at most one Steiner point in $O(n \log n)$ time.*

We have shown above that the Spiraling Rotating Caliper (SRC) Algorithm computes a quadrangulation for any set of n points and that this can be done with at most one Steiner point. The above theorem implies the following corollary:

Corollary 4.1 *A k-angulation of a set of points can be achieved with the addition of at most $k-3$ extra points.*

Now we show that the Steiner point is necessary only if the number of convex hull vertices h is odd and when h is even the SRC-Algorithm always yields a quadrangulation without the need for Steiner points.

Theorem 4.3 *Let S be a set of n points in general position in the plane. Then S may be quadrangulated with at most one Steiner point if the number of vertices on the convex hull of S is odd.*

Proof: Let h denote the number of points on the convex hull of S and let k denote the number of points in the interior of the convex hull. Therefore $n = h + k$. From Euler's theorem it follows that the number of edges in any triangulation $T(S)$ equals $2h + 3k - 3$. Now, any spanning simple polygonal chain on n vertices contains $n - 1$ edges. Since the outer shell of the convex spiral contains $h - 1$ edges and the inner spiral contains k edges, it follows that the entire convex spiral contains $h + k - 1$ edges. Therefore the number of diagonals in $T(S)$ equals $(2h + 3k - 3) - (h + k - 1) = h + 2k - 2$. Since 2 is even, $2k$ is even for any value of k, it follows that the expression is even if and only if h is even. Since the diagonals are ordered it follows that by removing every other diagonal the SRC-Algorithm yields a quadrangulation without a Steiner point if, and only if, h is even. Furthermore, one Steiner point is needed if, and only if, h is odd. ■

Figure 7 illustrates the convex spiral and resulting quadrangulation obtained with the SRC-Algorithm. Note that the quadrangulation is much *nicer* than the quadrangulation (shown in Figure 4) obtained with the sequential insertion method. In fact for this example not only are the quadrangles obtained with the SRC-Algorithm fat but they are all convex.

4.1 Handling Collinear Points

Only a few minor modifications need to be made to the SRC-Algorithm in order to handle sets of points that have collinearities. We assume that the convex hull of the points is not a line segment.

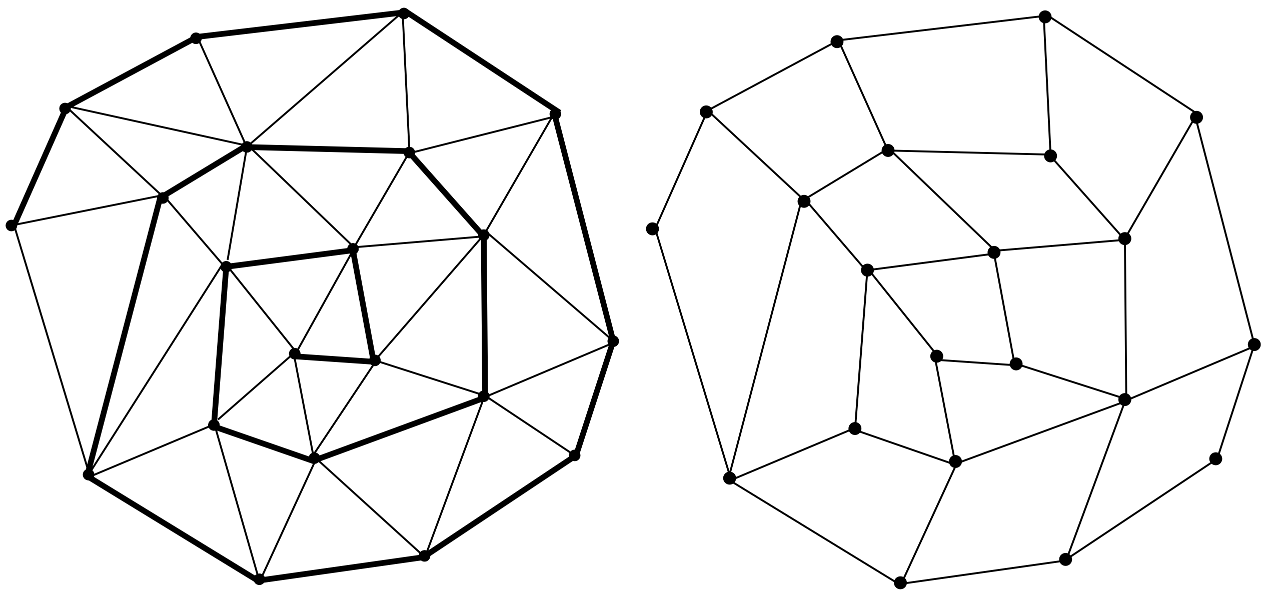

Figure 7: Illustrating the serpentine triangulation and the resulting quadrangulation of the set of points in Figures 4 and 5.

The first issue to address is to triangulate in a serpentine fashion a set of points where all points lie on the convex hull. Instead of connecting one vertex to all other vertices as is currently done in the algorithm, the following alternate algorithm may be used. Let $p_{x-\min}$ be the vertex with smallest x-coordinate. If more than one such vertex exists, then $p_{x-\min}$ has the smallest y-coordinate of all these vertices. Order the vertices on the convex hull from $p_{x-\min}$. Let $p_1 = p_{x-\min}$, p_2 the next vertex on the convex hull and so on until p_n. By construction p_1, p_2, p_n must form a triangle. If no other vertices are collinear with line segment $[p_1, p_2]$ or line segment $[p_1, p_n]$, the triangulation can be achieved by connecting p_1 to all other vertices. If there are some vertices collinear with $[p_1, p_2]$ and $[p_1, p_n]$, let p_j be the farthest vertex from p_1 on the line containing line segment $[p_1, p_2]$. Note that by construction p_n is the vertex on the line containing line segment $[p_1, p_n]$ closest to p_1. To triangulate the set, join p_n to $p_2, \ldots, p_{j-1}$ and join p_{j-1} to all other vertices $p_{n-1}, \ldots, p_j$.

The next issue is computing the spiral. Collinearities are handled with the spiraling procedure as long as the order of the collinear points is maintained on the polygonal chain. This can be accomplished in the simple $O(n^2)$ algorithm by sorting the sets of collinear points as they are found. Also, both Chazelle's algorithm [6] and Hershberger and Suri's algorithm [14] handle collinear points with minor modifications.

Another issue to address is how to compute a serpentine triangulation of the inner star-shaped region if there are some points on the line segment $[p_n, p_{n-1}]$ (Refer to Figure 6). If this situation occurs, then simply find the point on line segment $[p_{n-1}, p_{n-2}]$ closest to p_{n-1}, call this point p_j. Join p_j to all points on line segment $[p_n, p_{n-1}]$ and join p_n to the remaining points in the inner star-shaped region.

Finally, if collinear points are present it may happen that during the rotation of the calipers, both lines may arrive at groups of points simultaneously. In this case, they may be triangulated by *zig-zagging* between the groups in the order in which they are encountered. We conclude with the following:

Theorem 4.4 *A set of points S may be quadrangulated with at most one Steiner point in $O(n \log n)$ time with the SRC-Algorithm even in the presence of collinear points.*

5 Lower Bound

In this section, we provide a lower bound for the problem of quadrangulating a set of points. Paul and Simon [26] have shown that on a unit cost RAM with indirect addressing, branching based on comparisons, and the arithmetic operations +,-,*, the problem of sorting requires $\Omega(n \log n)$ time. We show that the problem of sorting N integers is linear-time transformable to the problem of quadrangulating a set of points in the above model of computation. Therefore, quadrangulating a set of points requires $\Omega(n \log n)$.

Theorem 5.1 *The problem of sorting n positive integers is $O(n)$ time transformable to the problem of quadrangulating a set of points on a unit cost RAM with indirect addressing, branching based on comparisons, and the arithmetic operations +,-,*. Therefore, quadrangulating a set of points requires $\Omega(n \log n)$ time in this model of computation.*

Proof: Let S be a set of n positive integers , $x_1, \ldots, x_n$. Assume without loss of generality that n is even. For each point x_i, we construct a corresponding point (x_i, x_i^2). Let P denote this set of points. Now, since every point of P is on the convex hull of P and n is even, we know that the set of points P can be quadrangulated. Let $Q(P)$ represent the quadrangulation of P. In $O(n)$ time, we can find the two points in P with smallest x coordinate, say a and b respectively. Given that a and b have been found, we can recover the convex hull of P with a simple traversal of $Q(P)$ similar to the Jarvis March. Note that the convex hull of P is contained in $Q(P)$ by definition. By construction, we know that the edge $[ab]$ is an edge of the convex hull of P. We know that there must be another convex hull edge sharing an endpoint with b. Draw a directed line L from a to b. Notice that the next edge on the convex hull has the property of making the smallest counterclockwise angle with respect to the line L. This implies that it is only necessary to verify all points adjacent to b and selecting the point that makes the smallest angle with L since $Q(P)$ contains the convex hull of P. By continuing in this way, we can recover the convex hull of P. Since at each step we only verify the vertices adjacent to a vertex in $Q(P)$, the complete traversal takes $O(n)$ time, since a quadrangulation only has $O(n)$ edges. For a more detailed treatment of the algorithm used to recover the convex hull, the reader is referred to O'Rourke [25].

Notice that the convex hull of P consists of a list of points sorted by abscissa. Therefore, the sorted order of S can be recovered in $O(n)$ time from the convex hull of P. ■

6 Conclusions

In this paper we have characterized those sets of points that admit a quadrangulation. We showed that S admits a quadrangulation if and only if S does not have an odd number of extreme points. If S admits a quadrangulation, we presented an algorithm that computes a quadrangulation of S in $O(n \log n)$ time, even in the presence of collinear points. If S does not admit a quadrangulation, then our algorithm quadrangulated S with the addition of one extra point. We also provided an $\Omega(n \log n)$ time lower bound for the problem. Finally, our results imply that a k-angulation of a set of points can be achieved with the addition of at most $k - 3$ extra points.

As a side benefit of our quadrangulation algorithm, we have obtained an optimal algorithm for computing serpentine (Hamiltonian) triangulations of point sets that yield very nice triangles. Arkin et al. [1] have shown that such triangulations have applications in fast rendering algorithms in computer graphics. Our algorithm yields much nicer triangulations than the algorithm of Arkin et al.

It is easy to see that a set of points does not always yield a quadrangulation where every quadrangle is convex. In Figure 8, the point x cannot be added to a quadrangulation of the given points without adding a non-convex quadrilateral. This observation leads to the following interesting questions. Can one decide if a set of points admits a quadrangulation where every quadrangle is convex? Can one compute the

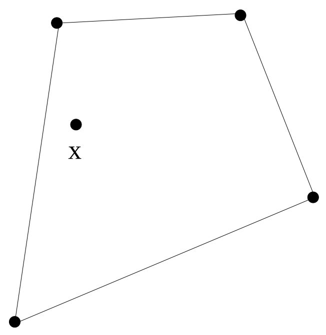

Figure 8: An example of a point set that does not admit a convex quadrangulation.

quadrangulation of a set of points that has the maximum number of convex quadrilaterals? As Lubiw [22] has shown that the decision problem is NP-complete for simple polygons with holes, it seems reasonable to believe that the same might hold for point sets.

There is an intriguing open question concerning the SI (Sequential Insertion) algorithm for quadrangulating point sets when collinearities are allowed. Joe Mitchell has shown that the SI algorithm can be modified so that it works in spite of collinearities, provided that we insert the points in some translation order, say, sorted by x-coordinate. Does an optimal algorithm exist for points inserted in arbitrary order? Finally, we have seen that paper-and-pencil examples suggest that the rotating-caliper algorithm gives significantly nicer quadrangulations than the SI algorithm. It would be interesting to compare these two algorithms both quantitatively and experimentally to determine the extent of the improvement afforded by the rotating-caliper algorithm for different measures of the quality of a quadrangulation.

Acknowledgments: We thank Larry Schumaker for bringing the problem to our attention. We thank Bernard Chazelle, Joe Mitchell, and Subhash Suri for discussions on implementation issues concerning point sets with collinearities. We thank Mike McAllister and T.S. Michael for noting a simplification of our proof of Lemma 3.1. We thank Anna Lubiw for discussions on models of computation in the proof of the lower bound.

References

[1] Arkin, E., M. Held, J. Mitchell, and S. Skiena, Hamiltonian triangulations for fast rendering, *Algorithms-ESA'94*, J. van Leeuwen, ed., Utrecht, NL, LNCS 855, pp. 36-47, September 1994.

[2] Asano, T., T. Asano, and H. Imai, Partitioning a polygonal region into trapezoids, *Journal of the ACM*, **33, 2**, pp. 290–312, 1986.

[3] Barequet, G., and M. Sharir, Piecewise-linear interpolation between polygonal slices, *Proceedings of the 10th Annual Symposium on Computational Geometry*, pp. 93-102, 1994.

[4] Baumgarten, H., H. Jung, and K. Mehlhorn, Dynamic point location in general subdivisions, *Journal of Algorithms*, **17**, pp. 342-380, 1994.

[5] Bondy, J. and U.S.R. Murty, *Graph Theory with Applications.* Elsevier Science, New York, New York, 1976.

[6] Chazelle, B., On the convex layers of a planar set, *IEEE Transactions on Information Theory*, **IT-31**, pp. 509-517, 1985.

[7] Cheng, S., and R. Janardan, New results on dynamic planar point location, *SIAM Journal on Computing*, **21**, pp. 972-999, 1992.

[8] Conn, H.E., and J. O'Rourke, Minimum weight quadrilateralization in $O(n^3 \log n)$ time, in *Proc. of the 28th Allerton Conference on Comm. Control and Computing*, pp. 788-797, 1990.

[9] deBerg, M., appears as "personal communication, 1992" in [10].

[10] Everett, H., W. Lenhart, M. Overmars, T. Shermer, and J. Urrutia, Strictly convex quadrilateralizations of polygons, in *Proceedings of the 4th Canadian Conference on Computational Geometry*, pp. 77-83, 1992.

[11] Fary, I., On straight lines representation of planar graphs, *Acta Sci. Math. Szeged*, **11**, pp. 229-233, 1948.

[12] Fournier, A., and D.Y. Montuno, Triangulating simple polygons and equivalent problems, in *ACM Transactions on Graphics*, **3, 2**, pp. 153–174, 1984.

[13] Heighway, E., A mesh generator for automatically subdividing irregular polygons into quadrilaterals, *IEEE Trans. Magn.*, **19, 6**, pp. 2535-2538, 1983.

[14] Hershberger, J., and S. Suri, Applications of a semi-dynamic convex hull algorithm, in *Proceedings of the second S.W.A.T., Lecture Notes in Computer Science 447*, Bergen, Sweden, pp. 380-392, 1990.

[15] Ho-Le, K., Finite element mesh generation methods: A review and classification, *Computer Aided Design*, **20**, pp. 27-38, 1988.

[16] Johnston, B., J. Sullivan, and A. Kwasnik, Automatic conversion of triangular finite element meshes to quadrilateral elements, *International Journal for Numerical Methods in Engineering*, **31**, pp. 67-84, 1991.

[17] Kahn, J., M. Klawe, D. Kleitman, Traditional galleries require fewer watchmen, *SIAM J. Algebraic Discrete Methods*, **4**, pp.194-206, 1983.

[18] Keil, J.M., and J.R. Sack, Minimum decompositions of polygonal objects, in *Computational Geometry*, Ed., G. T. Toussaint, North-Holland, Amsterdam, pp. 197-216, 1985. .

[19] Lai, M., *Scattered data interpolation and approximation by using C^1 piecewise cubic polynomials*, submitted for publication.

[20] Lai, M., and L. Schumaker, Scattered data interpolation using C^2 piecewise polynomials of degree six, *Third Workshop on Proximity Graphs*, Mississippi State University, Starkville, Mississippi, December 1-3, 1994.

[21] Lipski, W., E. Lodr, F. Luccio, C. Mugnal and L. Pagli, On two dimensional data organization II, *Fundanmenta Informaticae*, **2**, pp. 245-260, 1979.

[22] Lubiw, A., Decomposing polygonal regions into convex quadrilaterals, in *Proceedings of the 1st ACM Symposium on Computational Geometry*, pp.97-106, 1985.

[23] Ohtsuki, T., Minimum dissection of rectilinear regions, in *Proc. IEEE International Symposium on Circuits and Systems*, Rome, pp. 1210-1213, 1982.

[24] Okabe, A., B. Boots, and K. Sugihara, *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams.* John Wiley & Sons, Chichester, England, 1992.

[25] O'Rourke, J., *Computational Geometry in C*, Cambridge University Press, 1994.

[26] Paul, W., and J. Simon, *Decision trees and random access machines*, Logic and Algorithmics, Monograph 30, L'Enseignement Mathematique, 1980.

[27] Preparata, F., and M. Shamos, *Computational Geometry: An Introduction*, Springer-Verlag, New York, 1985.

[28] Preparata, F., and R. Tamassia, Fully dynamic point location in a monotone subdivision, *SIAM Journal on Computing*, **18**, pp. 811-830, 1989.

[29] Sack, J., and G. Toussaint, A linear-time algorithm for decomposing rectilinear star-shaped polygons into convex quadrilaterals, *Proc. 19th Annual Conf. on Communications, Control and Computing*, Allerton, pp. 21-30, 1981.

[30] Sack, J., and G. Toussaint, Guard placement in rectilinear polygons, in *Computational Morphology*, G. Toussaint, ed., North-Holland, pp. 153-175, 1988.

[31] Schroeder, W., and M. Shephard, Geometry-based fully automatic mesh generation and the Delaunay triangulation, *International Journal for Numerical Methods in Engineering*, **24**, pp. 2503-2515, 1988.

[32] Stein, S., Convex maps, *Proc. Amer. Math. Soc.*, **2**, pp. 464-466, 1951.

[33] Sugiyama, N., and K. Saitoh, Electron-beam exposure system AMDES," in *Computer Aided Design*, **11**, pp. 59-65, 1979.

[34] Toussaint, G., Solving geometric problems with the rotating calipers, *Proc. IEEE MELECON 83*, Athens, Greece, pp. A10002/1-4, 1983.

[35] Toussaint, G., New results in computational geometry relevant to pattern recognition in practice, *in Pattern Recognition in Practice II*, E. S. Gelsema and L. N. Kanal, Eds., North-Holland, pp. 135-146, 1986.

[36] Wagner, K., Bemerkungen zum Vierfarbenproblem, *Jber. Deutsch. Math.-Verein*, **46**, pp. 26-32, 1936.

[37] Wang, T., A C^2-quintic spline interpolation scheme on triangulation, *Computer Aided Geometric Design*, **9**, pp. 379-386, 1992.

[38] Wang, Y., and J. Aggarwal, Surface reconstruction and representation of 3-d scenes, *Pattern Recognition*, **19**, pp. 197-207, 1986.

[39] Wismath, S., *Triangulations: An algorithmic study,* Tech. Report 80-106, Queens University, Kingston, Canada, July 1980.

[40] Yoeli, P., Compilation of data for computer-assisted relief cartography, in *Display and Analysis of Spatial Data*, J. Davis, and M. McCullagh, eds., John Wiley & Sons, New York, 1975.