A Contribution to Triangulation Algorithms for Simple Polygons

Marko Lamot[1], Borut Žalik[2]

[1]Hermes Softlab, Ljubljana, Slovenia
[2]Borut Žalik, University of Maribor, Faculty of Electrical Engineering and Computer Sciences, Maribor, Slovenia

Decomposing simple polygon into simpler components is one of the basic tasks in computational geometry and its applications. The most important simple polygon decomposition is triangulation. The known algorithms for polygon triangulation can be classified into three groups: algorithms based on diagonal inserting, algorithms based on Delaunay triangulation, and the algorithms using Steiner points. The paper briefly explains the most popular algorithms from each group and summarizes the common features of the groups. After that four algorithms based on diagonals insertion are tested: a recursive diagonal inserting algorithm, an ear cutting algorithm, Kong's Graham scan algorithm, and Seidel's randomized incremental algorithm. An analysis concerning speed, the quality of the output triangles and the ability to handle holes is done at the end.

Keywords: simple polygons, simple polygon triangulation, Steiner points, constrained Delaunay triangulation

1. Introduction

Polygons are very convenient for computer representation of the boundary of the objects from the real world. Because polygons can be very complex (they can include a few thousand vertices, they may be concave and may include nested holes), often there is the need to decompose the polygons into simpler components that can be easily and rapidly handled. There are many ideas how to perform this decomposition. Planar polygons can be, for example, decomposed into triangles, trapezoids or even star-shaped polygons. Computing the triangulation of a polygon is a fundamental algorithm in the computational geometry. It is also the most investigated partitioning method. In computer graphics, polygon triangulation algorithms are widely used for tessellating curved geometries, such as those described by spline.

The paper gives a brief summary of existing triangulation techniques and a comparison between them. It is organized into eight sections. The second chapter introduces the fundamental terminology, and in the third chapter diagonal inserting algorithms are dealt with. The fourth section explains the constrained Delaunay triangulation. The fifth section describes triangulation techniques that use Steiner points. The sixth section summarizes the common features of all three groups of polygon triangulation algorithms. The seventh section contains the comparison of triangulation methods based on the diagonal insertion. An analysis concerning speed, the quality of the output triangles, and the ability to handle holes is done. The last section summarizes the work.

2. Background

Every simple polygon P (a polygon is simple if its edges cross only in their endpoints — vertices) with n vertices has a triangulation. The key for proving the existence of the triangulation is the fact that every polygon has a diagonal, which exists if the polygon has at least one convex vertex. We can conclude that [OROU94]:

- every polygon has at least one strictly convex vertex,
- every polygon with $n \geq 4$ vertices has a diagonal,
- every polygon P of n vertices may be partitioned into triangles by adding the diagonals.

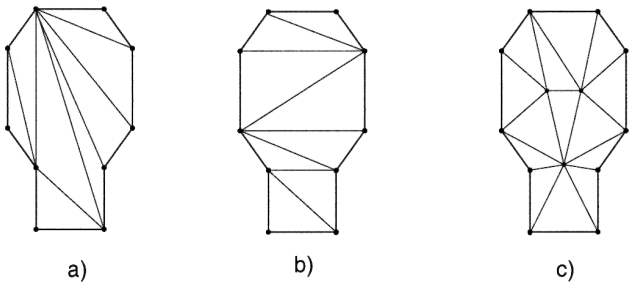

Fig. 1. a) Low quality triangulation; b) High quality triangulation; c) Triangulation with Steiner's Points.

There is a large number of different ways how to triangulate a given polygon. What all these possibilities have in common is that the number of diagonals is $n-3$ and the number of the triangles being generated is $n-2$. For details and proofs see [OROU94].

Following the fact of existence of the diagonal, a basic triangulation algorithm can be constructed as follows:

> Find a diagonal, cut the polygon into two pieces, and recurs each.

Finding diagonals is a simple task, which is repeated until all diagonals of the polygon are determined. This can be described as follows:

> For every edge e of the polygon not incident to either end of the potential diagonal s, check if e intersects s. As soon as an intersection is detected, it is known that s is not a diagonal. If no polygon edge intersects s, then s is a diagonal.

A rough analysis shows that such algorithm takes $O(n^4)$ time. Namely there are $\binom{n}{2} = O(n^2)$ diagonal candidates. Testing each of them with all polygon edges costs $O(n)$ time. Repeating this $O(n^3)$ process for each of the $n-3$ diagonals gives us an algorithm with at most $O(n^4)$ running time. Such a direct approach is, of course, too inefficient and therefore many authors proposed much faster triangulation algorithms.

There are different possibilities how to triangulate a given polygon (see Fig. 1). For some applications it is essential that the minimum interior angle of a triangle of the computed triangulation is as large as possible which defines quality. How the algorithms triangulated a simple polygon depended on the technique used by the algorithm. In Figure 1a for example, the triangulation can be considered as a low quality, because there are a lot of sliver triangles. The algorithms based on Delaunay triangulation ensure better triangulation (Figure 1b). The quality can be significantly improved by using so-called Steiner's points (Figure 1c).

3. Polygon triangulation algorithms based on diagonal inserting

History of the polygon triangulation algorithms began in 1911 [LENN11]. In that year Lennes proposed the "algorithm" which worked by recursively inserting diagonals between pairs of vertices of P and ran in $O(n^2)$. At that time mathematicians were interested in constructive proofs of existence of triangulation for simple polygons. Since then, this type of algorithm reappeared in many papers and books. Inductive proof for the existence of triangulation was proposed by Meisters [MEIS75]. He proposed an ear searching method and then cutting them off. Vertex v_i of simple polygon P is a principal vertex if no other vertex of P lies in the interior of the triangle v_{i-1}, v_i, v_{i+1} or in the interior of the diagonal v_{i-1}, v_{i+1}. A principal vertex v_i of simple polygon P is an ear if the diagonal v_{i-1}, v_{i+1} lies entirely in P. We say that two ears v_i, v_j are non-overlapping if interior $[v_{i-1}, v_i, v_{i+1}] \cap [v_{j-1}, v_j, v_{j+1}] = 0$ (see Figure 2).

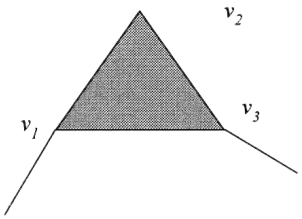

Fig. 2. Triangle $v_1v_2v_3$ is an ear of polygon.

Meisters proved the next theorem [MEIS75]:

> Except for triangles, every simple polygon P has at least two non-overlapping ears.

A direct implementation of this idea leads to a complexity of $O(n^3)$. But in 1990 it was discovered that prune and search technique finds an ear in the linear time [GIND93]. It is based on the following observation:

> A good subpolygon P_1 of a simple polygon P is a subpolygon whose boundary differs from that of P in one edge at the most.

The basic observation here is that a good subpolygon P_1 has at least one proper ear. Strategy is as follows:

> Split the polygon P of n vertices into two subpolygons in $O(n)$ time such that one of these subpolygons is a good subpolygon with at most $\lfloor n/2 \rfloor + 1$ vertices. Each subpolygon is then solved recursively.

The worst case running time of the algorithm is $T(n) = cn + T(\lfloor n/2 \rfloor + 1)$, where c is a constant. This recurrence has solution $T(n) \in O(n)$. That leads to implementation of Meisters's algorithm with the complexity of $O(n^2)$.

Garey, Johnson, Preparata and Tarjan proposed a divide — and — conquer algorithm which first broke $O(n^2)$ time complexity (1978) [GARE78]. Algorithm runs in $O(n \log n)$ time. Their approach includes two steps: the first one decomposes simple polygon into monotone subpolygons in $O(n \log n)$. The second step triangulates these monotone sub-polygons, which can be done in a linear time. A different divide — and — conquer approach by Chazelle also achieves $O(n \log n)$ running time [CHAZ82]. Very complicated data structures are used in Tarjan and Van Wyk's algorithm running in $O(n \log \log n)$ time [TARJ89]. However, Kirkpatrik introduced an algorithm with the same time complexity but with simple data structures [KIRK90].

Next improvement of speed was gained by algorithms with time complexity $O(n \log^* n)$. Such algorithms were not just faster but also simpler to implement. They all have in common a randomized ("Las Vegas") approach. The best known algorithm was suggested by Seidel [SEID91]. His algorithm runs in practice almost in linear time for the majority simple polygons. The algorithm has three steps:

- trapezoidal decomposition of the polygon,
- determination of monotone polygon's chains, and finally,
- the triangulation of these monotone polygon's chains.

The efficiency of Seidel's algorithm is achieved by very efficient trapezoidal decomposition, which works in two steps:

- first a random permutation of edges is determined, and
- these edges are inserted incrementally into trapezoidal decomposition.

With two corresponding structures containing current decomposition and search structure presented algorithm runs in $O(n \log^* n)$ time.

Researches also searched for classes of polygons that can be triangulated in a linear time. They determined that monotone polygons (Fig.

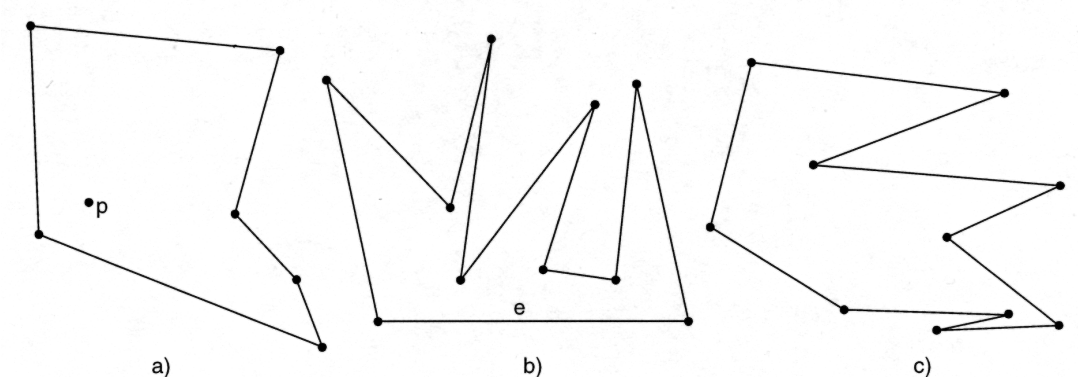

Fig. 3. Star polygons (a), edge visible polygons (b) and monotone polygons (c) can be triangulated in linear time.

3c), star-shaped polygons (each point in the polygon can be connected to star point p that no edge of polygon boundary is intersected) (Fig. 3a), spiral polygons, L-convex polygons, edge visible polygons (each point in the polygon can be connected to one point of edge e that no edge of polygon boundary is intersected) (Fig. 3b), intersection-free polygons and palm-shaped polygons could be triangulated in linear time

Some researches designed adaptive algorithms that run fast in many situations. Hertel and Mehlhorn described a sweep-line based algorithm that runs faster if a polygon has fewer concave vertices [HERT83]. Algorithm's running time is $O(n + r \log r)$ where r denotes the number of concave vertices of P.

Chazelle and Incerpi also presented algorithm where time complexity depends on shape of the polygon [CHAZ84]. They describe a triangulation algorithm that runs in $O(n \log s)$ time where $s < n$. The quantity s measures the sinuosity of the polygon representing how many times the polygon's boundary alternates between complete spirals of opposite orientation. In practice, quantity s is very small or a constant, even for very winding polygons. Consider the m otion of a straight line $L[v_i, v_{i+1}]$ passing through edge v_i, v_{i+1} where $0 < i < n$. Every time L reaches the vertical position in a clockwise (counter-clockwise) manner we decrement (increment) a winding counter by one. L is spiraling (anti-spiraling) if the winding counter is never decremented (incremented) twice in succession. A new polygonal chain is restarted only when the previous chain ceases to be spiraling or anti-spiraling.

Toussaint proposed in [TOUS91] another adaptive algorithm which runs in $O(n(1 + t_0))$; $t_0 < n$. The quantity t_0 measures the shape-complexity of the triangulation delivered by the algorithm. More precisely, t_0 is the number of triangles contained in the triangulation that share zero edges with the input polygon. The algorithm runs in $O(n^2)$ in the worst case, but for several classes of polygons it runs in the linear time. The algorithm is very simple to implement, because it does not require sorting or the use of balanced tree structures.

Kong, Everett and Toussaint algorithm is based on the Graham scan [KONG90]. The Graham scan is a fundamental backtracking technique in computational geometry. It has been shown how to use the Graham scan for triangulating simple polygon in $O(kn)$ time where $k - 1$ is the number of concave vertices in P. Although the worst case algorithm's time complexity is $O(n^2)$, it is easy to be implemented and there-

Time complexity	Author	Year	Technique/Algorithm
$O(n^2)$	Lennes	1911	Recursive diagonal insertion
$O(n^3)$	Meisters	1975	Ear cutting
$O(n^2)$	ElGindy, Everrett, Toussaint	1990	Prune and search
$O(n \log n)$	Garey, Johnson, Preparata, Tarjan	1978	Decomp. into monotone polygons
$O(n \log n)$	Chazelle	1982	Divide and conquer
$O(n + r \log r)$	Hertel & Mehlhorn	1983	Sweep — line
$O(n \log s)$	Chazelle & Incerpi	1983	—
$O(n(1 + t_0))$	Toussaint	1988	—
$O(kn)$	Kong, Everett, Toussaint	1990	Graham scan
$O(n \log \log n)$	Tarjan, Van Wyk	1987	—
$O(n \log \log n)$	Kirkpatrik	1990	—
$O(n \log^* n)$	Clarkson, Tarjan, Van Wyk	1989	Randomized incremental
$O(n \log^* n)$	Kirkpatrik, Klawe, Tarjan	1990	Using bounded integer coordinates
$O(n \log^* n)$	Seidel	1990	Randomized incremental
$O(n)$	Chazelle	1990	—

Table 1. Algorithms for computing triangulation of simple polygon based on diagonal inserting.

fore it is useful in practice. The algorithm adapts the Graham scan as following:

> The vertices of polygon P are scanned in order starting with v_2. At each step the current vertex is tested to determine if it is the top of an ear. If it is not, the current vertex is advanced, otherwise it is an ear and can be cut off. In that case current vertex is not advanced, except for a special case where the next vertex following cut ear is v_0.

Finally, in 1991 Chazelle presented $O(n)$ worst-case algorithm [CHAZ91]. Basic idea is in deterministic algorithm that computes structure called visibility map. This structure is a generalization of a trapezoidation (horizontal chords towards both sides of each vertex in a polygonal chain are drawn). His algorithm mimics merge sort. The polygon of n vertices is partitioned into chains with $n/2$ vertices, and these into chains of $n/4$ vertices, and so on. The visibility map of a chain is found by merging the maps of its subchains. This actually takes $O(n \log n)$ time at the most. But Chazelle improves the process by dividing it into two phases. The first phase includes computing coarse approximations of the visibility maps. This visibility maps are coarse enough that merging can be accomplished in a linear time. The second phase refines the coarse map into a complete visibility map, also in the linear time. A triangulation is then produced from the trapezoidation defined by the visibility map. The algorithm has a lot of details and therefore it remains open to find a simple and fast algorithm for triangulating a polygon in the linear time. All mentioned algorithms are summarized in Table 1.

4. Polygon triangulation algorithms based on Delaunay triangulation

Triangulation of the simple polygons can also be achieved by the well-known Delaunay triangulation [FLOR92] on a set of points. Namely, the vertices of a polygon can be considered as individual input points in the plane. When computing the Delaunay triangulation we have to consider that some line segments (edges of polygon) must exist at the output. That problem is known as a constrained Delaunay triangulation (CDT).

Let V be a set of points in the plane and L set of non-intersecting line segments having their extreme vertices at points of V. The pair $G = (V, L)$ defines a constraint graph.

Two vertices $P_iP_j \in V$ are said to be mutually visible if either segment P_iP_j does not intersect any constraint segment or P_iP_j is a subsegment of a constraint segment of L.

Now the visibility graph of G is a pair $G_v = (V_v, E_v)$; $V_v = V$ and $E_v = \{(P_i, P_j) \mid P_i,\ P_j \in V_v$ and $P_i,\ P_j$ are mutually visible with respect to set $L\}$ (see Fig. 4b).

An edge in E_v joins a pair of mutually visible points of V with respect to all straight-line segments belonging to L.

So, triangulation of V constrained by L is defined as a graph $T(V;L) = (V_t, E_t)$; $V_t = V$ and E_t is a maximal subset of $E_v \cup L$ such that $L \subseteq E_t$, and no two edges of E_t intersect, except at their endpoints.

A CDT $T(V;L)$ of set of points V with respect to a set of straight-line segments L is a constrained

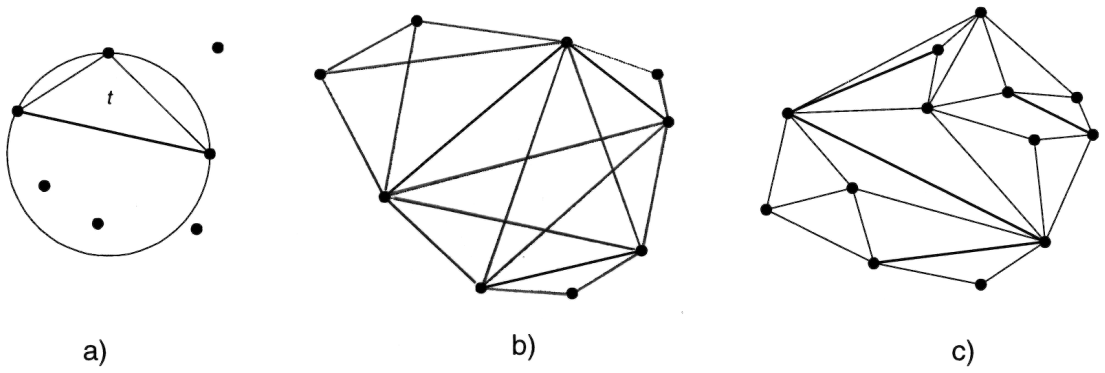

Fig. 4. a) Empty circle property; b) Visibility map; c) Constrained Delaunay triangulation.

triangulation of V in which the circumcircle of each triangle t of T does not contain in its interior any other vertex P of T which is visible from the three vertices of t (see Fig. 4c). Another characterization of CDT is given by the empty circle property: a triangle t in a constrained triangulation T is a Delaunay triangle if there does not exist any ot her vertex of T inside the circumcircle of t and visible from all three vertices of t (see Fig. 4a). See details in [FLOR92].

The Delaunay triangulation of simple polygon can be generally computed as follows: the first step computes CDT of edges of simple polygon and the second step removes triangles that are in exterior of simple polygon. The information that input is a simple polygon (not just general constraint graph) could be useful in step one and therefore algorithms for building a CDT can be subdivided into two groups:

- algorithms for computing the CDT when the constraint graph is a simple polygon,
- algorithms for computing a CDT for general constraint graph.

4.1. Constrained Delaunay triangulation algorithms for simple polygons

Lewis and Robinson described an $O(n^2)$ algorithm based on divide-and-conquer approach with internal points of simple polygon [LEWI79]. The boundary polygon is recursively subdivided into almost equally sized subpolygons that are separately triangulated together with their internal points. The resulting triangulation is then optimized to produce CDT.

A recursive $O(n^2)$ algorithm for CDT based on visibility approach is described by Floriani [FLOR85]. The algorithm computes the visibility graph of the vertices of the simple polygon Q in $O(n^2)$ time and the Voronoi diagram P of set of its vertices in $O(n \log n)$. The resulting Delaunay triangulation is built by joining each vertex Q of P to those vertices that are both visible from Q and Voronoi neighbors of Q.

Another $O(n \log n)$ algorithm was described by Lee and Lin in [FLOR92]. The algorithm is based on Chazelle's polygon cutting theorem. Chazelle has shown that for any simple polygon P with n vertices, two vertices t_1 and t_2 of P can be found in a linear time such that segment t_1t_2 is completely internal to P. Each of the two simple subpolygons resulting from the cut of P by t_1t_2 has at least $n/3$ vertices. Lee and Lin's algorithm subdivides the given polygon Q into two subpolygons Q_l and Q_r and recursively computes the constrained Delaunay triangulations T_l and T_r. The resulting triangulation T of Q is obtained by merging T_l and T_r. They also proposed a similar algorithm for general constraint graph which runs in $O(n^2)$.

4.2. Constrained Delaunay triangulation algorithms for general constraint graphs

Chew describes an $O(n \log n)$ algorithm for the CDT based on the divide-and-conquer approach. The constraint graph $G = (V, L)$ is assumed to be contained in a rectangle, which is subdivided into vertical strips [CHEW87]. In each strip there is exactly one vertex. The CDT is computed for each strip and adjacent strips are recursively merged together. After last merge we got the final CDT. The major problem here is merging those strips that contain edges, which cross some strip having no endpoint in it.

Algorithm for computing CDT, which includes preprocessing on the constraint segments, is proposed by Bossiant [BOIS88]. By preprocessing CDT, the problem is transformed into standard Delaunay problem on a set of points. The idea is to modify the input data by adding points lying on the constraint segments in such a way that resulting Delaunay triangulation is guaranteed to contain such segments. Constraint segment e is a Delaunay edge if the circle having e as diameter does not intersect any other constraint segment. If the circle attached to e intersects some other segment, then e is split into a finite number of subsegments such that none of the circles attached to those segments intersect any constraint. When two constraint segments intersect at an endpoint, one new point is inserted into both segments. The circumcircle of the triangle defined by the common endpoint and by the two new points does not intersect any other constraint segment. This algorithm takes at most $O(n \log n)$ time and generates at most $O(n)$ additional points.

For CDT all the above algorithms require that all points are defined at the beginning of the

triangulation process. An algorithm proposed by Floriani and Puppo [FLOR92] resolves CDT problem by incrementally updating CDT as new points and constraints are added. The problem of incrementally building of CDT is reduced to the following three subproblems:

– computation of an initial triangulation of the domain,

– insertion of a point,

– insertion of a straight-line segment.

An initial triangulation of the domain can be obtained by different approaches. For example, we can determine a triangle or rectangle (made of two triangles), which contain the whole domain. Then, points and straight-lines are incrementally inserted. After each insertion we get new CDT which has more elements than the previous one. After inserting the last point or straight-line, the bounding triangle is removed. Algorithm runs at most in $O(ln^2)$ where n is number of points and l the number of straight-line segments in the final CDT.

Table 2 shows the algorithms for computing triangulation of a simple polygon based on Delaunay triangulation.

5. Polygon triangulation algorithms by using Steiner points

Finally, the algorithms that care also about the quality of triangulation are considered. The quality is checked regarding the minimum interior angle of triangles in the output triangulation. Generally, that feature is possible only if the use of so-called Steiner points is allowed. In that case the number of output triangles is increased regarding the minimum number of triangles in output triangulation. In other words, we want to provide shape guarantee (minimum interior angle is as high as possible) with minimum triangles in the output triangulation (size guarantee).

One of such techniques of triangulation points and straight-lines is Delaunay refinement technique. Chew presented a Delaunay refinement algorithm that triangulates a given polygon into a mesh. In mesh all triangles are between 30° and 120°. The algorithm produces a uniform mesh to obtain all triangles of the roughly the same size [CHEW89].

Ruppert extended Chew's work [RUPP94] by giving an algorithm such that all triangles in the output have angles between $\pi - 2\alpha$. Parameter α can be chosen between 0° and 20°. The triangulation maintained here is a Delaunay triangulation set of points which is computed at the beginning. Vertices for Delaunay triangulation are in that case endpoints of segments and possible isolated vertices. After computing Delaunay triangulation, vertices are added for two reasons: to improve triangle shape, and to ensure that all input segments are presented in Delaunay triangulation. Two basic operations in the algorithm are splitting a segment by adding a vertex at its midpoint, and splitting a triangle with a vertex at its circumcenter. In each case, the new vertex is added to set of vertices. When a segment is split, it is replaced in set of segments by two subsegments. Such algorithms runs in $O(M^2)$ time, where M is number of vertices at the output, but in practice are very fast.

Time complexity	Author	Year	Input
$O(n^2)$	Lewis, Robinson	1979	Simple polygon
$O(n \log n)$	Floriani	1985	Simple polygon
$O(n \log n)$	Lee, Lin	1980	Simple polygon
$O(n^2)$	Lee, Lin	1980	General
$O(n \log n)$	Chew	1987	General
$O(n \log n)$	Boissonnat	1988	General
$O(ln^2)$	Floriani, Puppo	1992	General

Table 2. Triangulation algorithms based on Delaunay triangulation.

Time complexity	Author	Year	Input
$O(n \log n + k)$	Bern, Eppstein	1991	General
$O(n \log n)$	Bern, Dobkin	1995	Simple polygon
$O(n \log 2n)$	Bern, Mitchell	1995	Simple polygon
$O(M^2)$	Chew	1989	Simple polygon
$O(M^2)$	Ruppert	1994	General

Table 3. Triangulation algorithms based on using Steiner points.

Some other algorithms that give shape guarantees are available. They are more complicated to implement and are not based on Delaunay triangulation. Baker [BAKE88] has given an algorithm to triangulate the interior of a simple polygon with elements whose angles are between 13° and 90°. The number of triangles used by their algorithm may be unnecessarily large. But they suggested that quadtrees might improve the size of the triangulation. Bern, Eppstein and Gilbert [BERN92] followed up this suggestion, as well as giving a new size bound. They showed how to triangulate a planar point set or poligonally bounded domain with triangles of bounded aspect ratio. Triangulation has size (number of triangles) within a constant factor of optimal and runs in optimal time $O(n \log n + k)$ with input of size n and output of size k. Bern, Dobkin, and Eppstein [BERN95] showed how to triangulate polygonal regions with triangles of guaranteed quality. Algorithm guarantees, using $O(n)$ triangles, that the smallest height (shortest dimension) of a triangle in a triangulation of an n-vertex polygon (with holes) is a constant factor of the largest possible. Using $O(n \log n)$ triangles for simple polygons ($O(n^{3/2})$ for polygons with holes) they guarantee the largest angle is not greater than 150°. Such triangulation can be obtained in $O(n \log n)$ time ($O(n \log n + k)$ for polygons with holes). Another algorithm presented by Bern, Mitchell and Ruppert [BEMI95] considers triangulation of n-vertex polygonal regions so that no angle in the final triangulation measures more than $\pi/2$. The number of triangles in the triangulation is only $O(n)$ and the running time is $O(n \log^2 n)$. Algorithm also considers holes in polygons. Basic new technique used in the algorithm is recursive subdivision by disks and consists of two stages: first stage packs the domain with non-overlapping disks, tangent to each other and to sides of the domain. Disk packing is such that each region not covered has at most four sides. The algorithm then adds edges between ce ntres of disks and points of tangency on their boundaries, thereby dividing the domain into small polygons. Second stage triangulates the small polygons using Steiner points located only interior to the polygons or on the domain boundary.

Table 3 shows algorithms and their time complexities described in this section.

6. Common features of groups of algorithms for polygon triangulation

In this section, general properties of all three groups of the algorithms for polygon triangulation (algorithms based on diagonal insertion, algorithms based on Delaunay triangulation, and algorithms using Steiner points) are considered. Attributes interesting for comparisons are [SHEW]:

- quality of a triangulation,
- number of output triangles, and
- the possibility of triangulating polygons with holes.

Figure 5 shows (extended figure from [RUPP94]) how different triangulation is obtained while triangulating the same polygon by different algorithms:

- triangulation in Fig. 5a has been generated by Seidel's randomized incremental algorithm generates the output in Fig. 5a [SEID91] (based on diagonal inserting),
- De Floriani and Puppo's constrained Delaunay triangulation generated the Fig. 5b [FLOR92] (based on Delaunay triangulation),

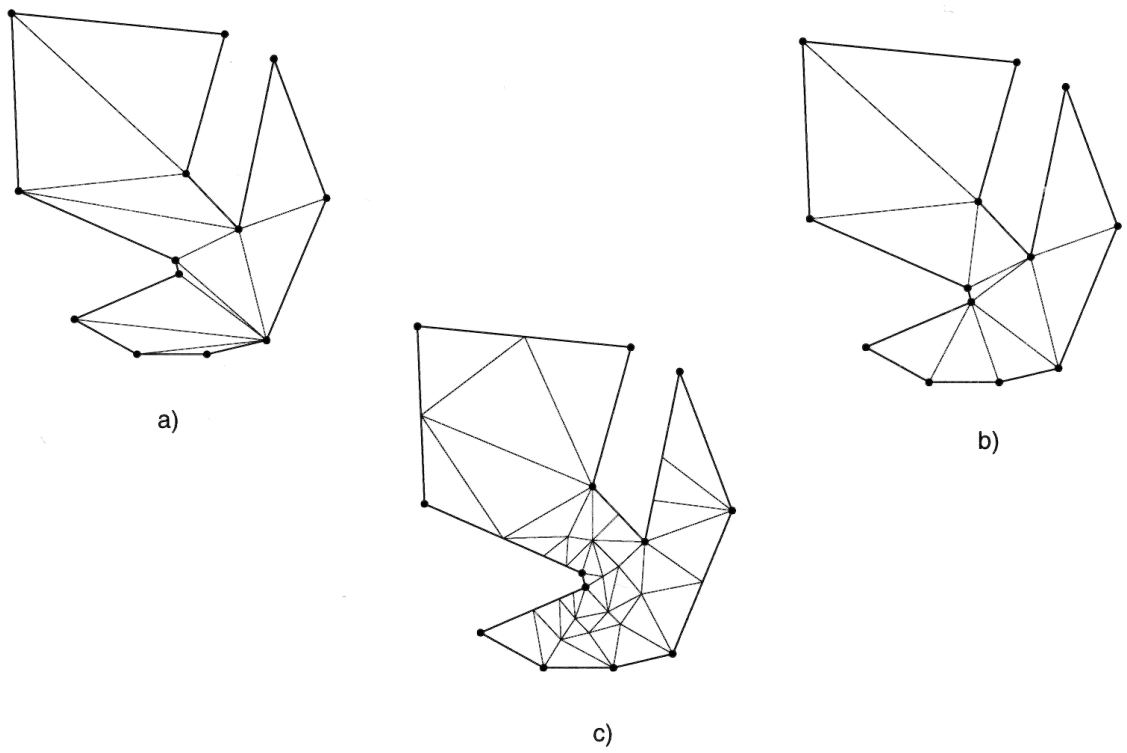

Fig. 5. a) Diagonal inserting (Seidel); b) Constrained Delaunay triangulation (De Floriani & Puppo); c) Delaunay refinement (Ruppert).

- Fig. 5c shows the output of Ruppert's Delaunay refinement algorithm [RUPP94] (using Steiner points).

It is obvious; the algorithms that use Steiner points achieve the best output triangulation. They have built-in mechanism ensuring the quality of the output triangulation. However, they produced also the larger number of triangles. Delaunay-based algorithms provide the highest quality possible on the original vertices (Fig. 5c). During the construction of the triangulation, they consider so-called Delaunay empty-circle property already mentioned in Section 4. Algorithms based on the diagonal insertion do not care at all about the quality of the triangulation and because of this, different outputs are obtained by different algorithms (see section 7 where four algorithms from this group are compared). However, Delaunay-based algorithms and the algorithms based on diagonal insertion always generate exactly $n-2$ triangles. If we want to obtain the smallest number of triangles possible, and we want to be sure about the quality of the triangulation, then one of the algorithms based on Delaunay triangulation has to be used.

The algorithms based on Delaunay triangulation and those using Steiner points handle polygons with holes very easily. They triangulate also the holes, but because we know which edges of the polygon belong to the holes, the triangles inside the holes can be easily removed. The majority of algorithms based on diagonal insertion cannot perform the triangulation of polygons containing holes. One of the exceptions is Seidel's randomized incremental algorithm. Originally, it has not being designed to handle polygons with holes, but it turned out that this extension is very simple. The solution is described in the next section.

Concerning speed of the algorithms it is really difficult to compare them. In each group there are the solutions working in $O(n \log n)$ time (we do not take into account Chazelle's algorithm which has not yet been successfully implemented). However, concerning the implementation, the algorithms based on diagonal inserting are the easiest to implement.

7. Comparison of triangulation algorithms based on diagonal insertion

We have already mentioned that the algorithms based on diagonal insertion do not use any criteria regarding the quality of the generated triangulation and therefore they produce also very different results. This is the reason why in this section some algorithms based on diagonal insertion are analyzed.

The following algorithms have been chosen: a recursive diagonal inserting algorithm [TOUS91], an ear cutting algorithm [MEIS75], Kong's Graham scan algorithm [KONG90], and Seidel's randomized incremental algorithm [SEID91]. All the algorithms have been implemented in C++ using the same data structures. Figure 6 shows how different output is generated by these four algorithms.

We tested the speed of the algorithms on two types of polygons: concave polygons and convex polygons. We also took convex polygons to determine how fast the algorithms are on polygons where triangulation can be obtained in linear time with the simple algorithm (see table 4, column Actual Running Time).

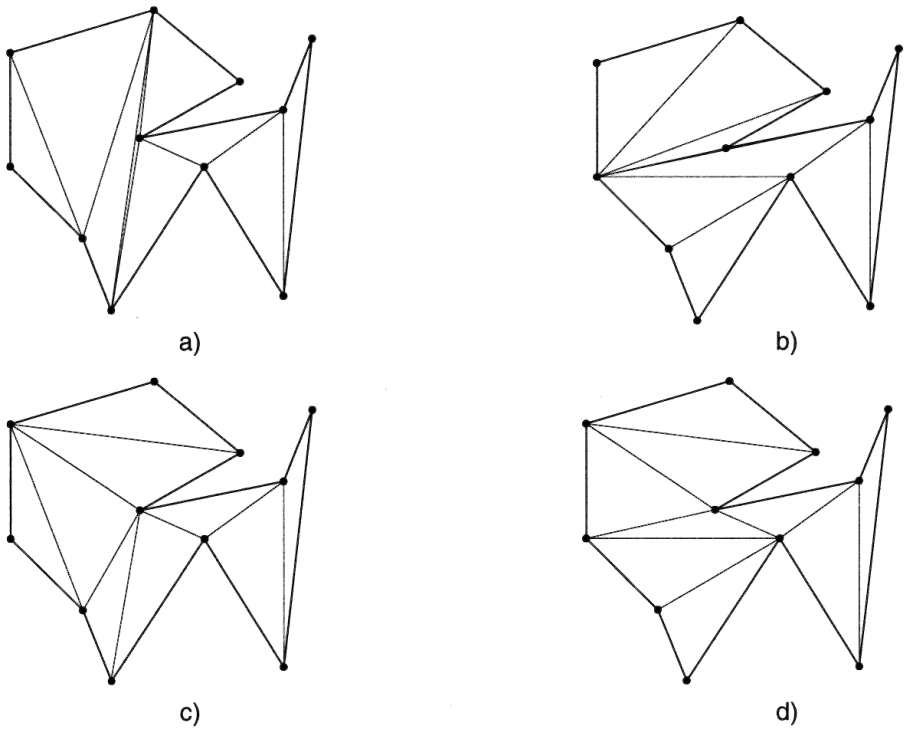

Fig. 6. a) Diagonal inserting; b) Ear cutting; c) Graham scan; d) Randomized incremental.

Algorithm	Running time	Average angle	Actual running time in ms for 5000 points		Holes
			Convex	Concave	
Recursive diagonal inserting	$O(n^2)$	3.40°	5523	5671	No
Ear cutting	$O(n^3)$	2.45°	10285	10325	No
Kong's Graham scan	$O(kn)$	4.06°	6970	6950	No
Seidel's rand. Incremental	$O(n \log^* n)$	7.94°	330	331	Yes

Table 4. Comparison of algorithms based on diagonal insertion.

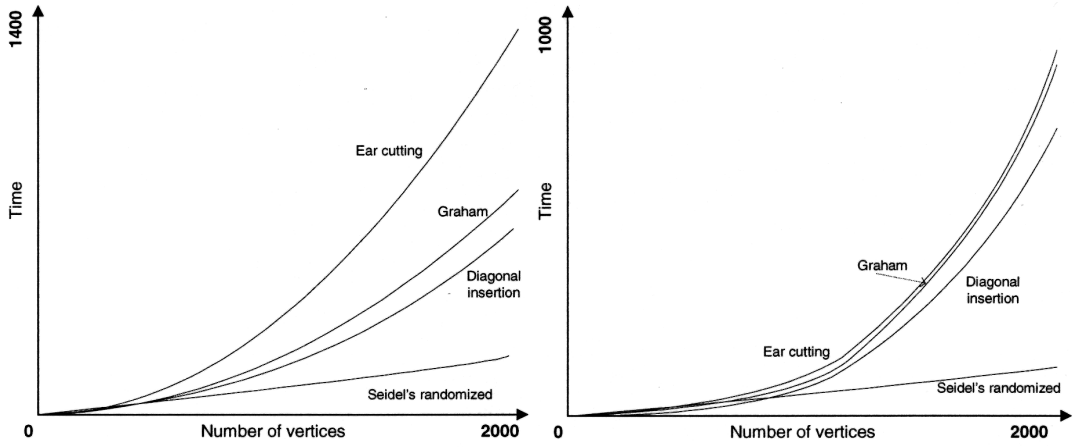

Fig. 7. Speed of algorithms (left — concave polygon; right — convex polygon).

Fig. 7 presents the graph showing the running time of the considered algorithms. For each type of polygons we measured the speed when polygons contain from 3 to 2000 vertices (with step 50). It is clearly seen that Seidel's randomized algorithm is in practice the fastest algorithm among all compared algorithms. Seidel's algorithm is also not sensitive to on the type of the input polygon. Other three algorithms are considerably slower and their run-time depends on the shape of the input polygon (compare left and right Fig. 7).

As we already said, the algorithms from this group do not contain any mechanism for ensuring the quality, and because of different techniques used, the algorithms provide different quality of the output triangles. Table 4 shows the average minimum interior angle of obtained triangles for concave polygons with 500 vertices (test was repeated 100 times on different polygons with 500 vertices). Again, the Seidel's randomized incremental algorithm provides the best result. The reason is that Seidel's algorithms first decompose the input polygon into monotone polygons followed by partitioning of polygons into triangles.

Only Seidel's algorithm is capable to solve situations when the polygons containing holes are considered (see Table 4, column holes). This is possible because Seidel's randomized algorithm takes edges randomly on input and builds trapezoidation. Therefore, we can also add edges of holes on input. After building such trapezoidation we only construct monotone polygons, which are lying in the polygons but not in holes.

As shown from Table 4 the Seidel's algorithm is the best regarding all other three algorithms. The price for this is more complicated implementation.

8. Conclusions

Triangulation of a simple polygon is the frequent task in computational geometry and its applications (geographic information systems, finite element mesh generation) and today, a large number of different algorithms are known. These algorithms can be classified into three major groups: methods based on diagonal insertion, constrained Delaunay approaches, and algorithms using Steiner points. All three groups are considered in the paper and the most popular algorithms are briefly described. We have pointed on the most important characteristics of each group regarding the number of output triangles, the quality of the triangulation, and the possibility to handle the algorithms with holes. Because algorithms based on diagonal insertion do not include any strategy for optimization their output, the resulting triangulations differ noticeably. Because of this we compared four algorithms from this group regarding the run time, the quality of the output triangulation, and the ability of solving the polygons with holes. We have shown that Seidel's algorithm is the best in this category.

The development of polygon triangulation algorithms is still attractive research topic in computational geometry. The real challenge is Chazelle diagonal inserting algorithm [CHAZ91]. Although theoretically known from 1991, up to now there is no any successful implementation. The new algorithms are still expected among the constraint Delaunay triangulation algorithms and especially among the algorithms using Steiner points.

9. Acknowledgement

This work has been carried out within a Valvasor/ALIS program supported by British Council (title of the project Computational geometry and its applications in medical visualization).

References

[1] B. BAKER, E. GROSSE, C. RAFFERTY Nonobtuse triangulation of polygons. *Discrete and Comp. Geometry*, **3** (1988), 147-168.

[2] M. BERN, S. MITCHELL, J. RUPPERT Linear-Size Nonobtuse Triangulation of Polygons. *Discrete Comput. Geometry*, **14** (1995), 411-428.

[3] M. BERG, M. KREVELD, M. OVERMARS, O. SCHWARZKOPF, *Computational Geometry, Algorithms and Application.* Springer Verlog, 1997.

[4] M. BERN, D. EPPSTEIN, J. R. GILBERT, Provably good mesh generation. Presented at the *Proceedings of the* 31^{st} *Annual Symposium on Foundation of Computer Science*, (1990), 231-241.

[5] M. BERN, D. DOBKIN, D. EPPSTEIN, Triangulating polygons without large Angles. *Computationl Geometry & Applications*, **5** (1995), 171-192.

[6] J. D. BOISSONNAT, Shape reconstruction from planar cross sections. *Comput. Vision Graphics Image Process.*, **44** (1988), 1-29.

[7] B. A. CHAZELLE, Theorem on polygon cutting with applications. Presented at the *Proceedings of the* 23^{rd} *IEEE Symposium on Foundation of Computer Science*, (1982), Chicago, 339-349.

[8] B. CHAZELLE, J. INCERPI, Triangulation and shape complexity. *ACM Transactions on Graphics*, **3** (1984), 135-152.

[9] B. CHAZELLE, Triangulating a simple polygon in linear time. *Discrete Computational Geometry*, **6** (1991), 485-524.

[10] L. P. CHEW, Constrained Delaunay triangulation. Presented at the *Proceedings, Third ACM Symposium on Computational Geometry*, (1987), Waterloo, 216-222.

[11] L. P. CHEW, Guaranteed — quality triangular meshes. Technical report, No. TR-89-983, Cornell University, 1989.

[12] L. DE FLORIANI, B. FALCIDIENO, C. A. PIENOVI, Delaunay-based representation of surfaces defined over arbitrarily-shaped domains. *Comput. Vision Graphics Image Processing.* **32** (1985), 127-140.

[13] L. DE FLORIANI, E. PUPPO, An On-Line Algorithm for Constrained Delaunay Triangulation, *Graphical Models and Image Processing*, **3** (1992), 290-300.

[14] M. R. GAREY, D. S. JOHNSON, F. P. PREPARATA, R. E. TARJAN, Triangulating a simple polygon, *Inform. Process.*, **7** (1978), 175-180.

[15] H. ELGINDY, H. EVERETT, G. T. TOUSSAINT, Slicing an ear in linear time. *Pattern Recognition Letters*, **14** (1993), 719-722.

[16] S. HERTEL, K. MEHLHORN, Fast triangulation of simple polygons. Presented at *Proceedings* 4^{th} *Internat. Conf. Theory*, (1983) pp. 207-215.

[17] D. G. KIRKOATRICK, M. M. KLAWE, R. E. TARJAN, $O(n \log\log n)$ polygon triangulation with simple data structures. Presented at *ACM Symposium on Computational Geometry*, 6 (1990) pp. 34-43, Berkeley, California.

[18] X. KONG, H. EVERETT, G. T. TOUSSAINT, The graham scan triangulates simple polygons. *Pattern Recognition Letters*, **11** (1990), 713-716.

[19] N. J. LENNES, Theorems on the simple finite polygon and polyhedron. *American Journal of Mathematics*, **33** (1911), 37-62.

[20] B. A. LEWIS, J. S. ROBINSO, Triangulating of planar regions with applications. *Comput. J.*, **4** (1979), 324-332.

[21] G. H. MEISTERS, Polygons have ears. *American Mathematical Monthly*, **82** (1975), 648-651.

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

[23] J. RUPPERT, A Delaunay Refinemt Algorithm for Quality 2-Dimensional Mesh Generation. NASA Arnes Research Center, Submission to Journal of Algorithms, (1994), `http://jit.arc.nasa.gov/nas/abs.html`.

[24] R. SEIDEL, A simple and fast incremental randomized algorithm for computing trapezoidal decompositions and for triangulating polygons. *Computational Geometry: Theory and Applications*, **1** (1991), 51-64.

[25] J. R. SHEWCHUK, Triangle: Engieering a 2D Quality Mesh Generator and Delaunay Triangulator. Carnegie Mellon University, `ftp:cs.cmu.edu`.

[26] R. E. TARJAN, C. J. VAN WYK, An $O(n \log\log n)$ — time algorithm for a simple polygon triangulation and its evaluation. IEICE Technical report, No. PRU89-41, September 1989.

[27] G. T. TOUSSAINT, Efficient triangulation of simple polygons. *The Visual Computer*, **7** (1991), 280-295.

Received: October, 2000
Accepted: November, 2000

Contact address:
Marko Lamot
Hermes Softlab
Litijska 51
1000 Ljubljana
Slovenia
Tel: +386 61 1865 815;
Fax: +386 61 1865 816;
e-mail: marko.lamot@hermes.si

Borut Žalik
University of Maribor
Faculty of Electrical Engineering and Computer Sciences
Smetanova 17
2000 Maribor
Slovenia

MARKO LAMOT currently works for at Hermes SoftLab computer company, Ljubljana, Slovenia. He received his BSc in computer engineering in 1996 from the University of Maribor, Slovenia. Currently he is working on his PhD degree at University of Maribor. His research interests include computational geometry especially triangulating algorithms and uniform grids.

BORUT ŽALIK is currently an associate professor at the Faculty of Electrical Engineering and Computer Science at the University of Maribor, Slovenia. He also has a position of a senior research fellow at De Montfort University, U. K. He received his BSc in electrical engineering in 1985, MSc and PhD in computer science, both from the University of Maribor in 1989 and 1993, respectively. His research interests include computational geometry, geometric modeling, GIS applications, and multimedia applications.