An Overview of Triangulation Algorithms for Simple Polygons

Marko Lamot
Hermes Softlab, Kardeljeva 100, 2000 Maribor, Slovenia

Borut Zalik
Faculty of Electrical Engineering and Computer Sciences, Smetanova 17, 2000 Maribor, Slovenia

Abstract

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. Different techniques for triangulating simple polygon were designed. The first part of the paper is an overview of triangulation algorithms based on diagonal inserting. In the second part we present algorithms based on Delaunay triangulation. The basic ideas and approach for each algorithm are presented in this paper. The last part of the paper some representative algorithm by its efficiency are compared.

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 (can include a few thousand vertices, may be concave and may include nested holes), there is many times the need to decompose the polygons into simpler components which can be easily and faster handled. There are many ideas how to perform this decomposition. Planar polygons can be, for example, decomposed into triangles, convex parts, trapezoids or even star-shaped polygons. Computing the triangulation of a polygon is a fundamental algorithm in computational geometry. It also seems to be 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 comparison between them. We will try to cover all techniques for triangulating simple polygons. It is organized into five sections. The second chapter introduces the fundamental terminology, in the third chapter a diagonal inserting algorithms are considered. Fourth section explains the constrained Delaunay triangulation, together with the approaches which introduce Steiner points. The last fifth section contains the comparison of mentioned triangulation methods. For each algorithm, the basic idea and worst time complexity estimation is given. The quality of triangulation is also considered.

2. Background

Every simple polygon P (polygons with edges crossing only in its endpoints) with n vertices has a triangulation. The key for proving of existence of 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 [17]:

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

There is a large number of different ways how to triangulate a given polygon. All these possibilities have in common that the number of diagonals is $n - 3$ and the number of the triangles being generated is $n - 2$. See for details and proofs in [17].

Following the fact of existence of 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 simple application which repeats until all diagonals of polygon are determined. For every edge e of the polygon not incident to either end of the potential diagonal s, see 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.

Such direct approach is too inefficient (it takes $O(n^4)$ time) and therefore many authors proposed much faster triangulation algorithms.

As mentioned above, there is a large number of different ways how to triangulate a given polygon (see figure 1). For some applications it is essential that the minimum interior angle of a triangle of the computed triangulation is as large as possible what defines a quality measure. The way that algorithm triangulates a simple polygon is dependent on technique used in algorithm. In figure 1a, for example, the triangulation can be considered

as a low quality, because there exist a lot sliver triangles. The algorithms based on Delaunay triangulation ensure better quality triangulation (figure 1b). The quality can be significantly improved by using so-called Steiner's points (figure 1c).

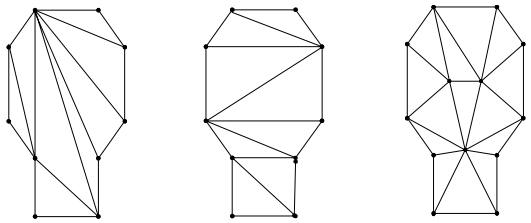

Figure 1: a) Low quality triangulation; b) High quality triangulation; c) Triangulation with Steiner's Points

It is also important into which class of the polygons the input polygon can be classified. It has been shown that monotone polygons, star-shaped polygons, spiral polygons, L-convex polygons, edge visible polygons, intersection-free polygons, palm-shaped polygons and anthropomorphic polygons can be triangulated in linear time [13].

3. Polygon triangulation algorithms based on diagonal inserting

History of polygon triangulation algorithms begins in the year 1911 [14]. That year Lennes proposed an "algorithm" which works by recursively inserting diagonals between pairs of vertices of P and runs in $O(n^2)$ time. At that time mathematicians have been 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 existence of triangulation was proposed by Meisters [16]. 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 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 $[v_{i-1}, v_i, v_{i+1}] \cap [v_{j-1}, v_j, v_{j+1}] = 0$.

Meisters proved another theorem: except for triangles every simple polygon P has at least two non-overlapping ears [16]. 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 [11]. 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 at most one edge. The basic observation here is that a good subpolygon P_1 has at least one proper ear. Strategy now is as follows: split 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 $O(n^2)$ complexity [10]. Algorithm runs in $O(n \log n)$ time. Their approach includes two steps: the first one decomposes simple polygon into monotone sub-polygons in $O(n \log n)$. The second step triangulates these monotone sub-polygons what can be done in a linear time. Different divide and conquer approach by Chazelle also achieves $O(n \log n)$ running time. Very complicated data structures are used in Tarjan and Van Wyk algorithm that runs in $O(n \log \log n)$ time. However, the same complexity was introduced by Kirkpatrik using simple data structures.

Next improve of speed were 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 most well-know algorithm has been suggested by Seidel [19]. His algorithm runs in practice almost in linear time for the majority of simple polygons. The algorithm has three steps: trapezoidal decomposition of 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: first a random permutation of edges is determined and then 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.

Some researchers designed adaptive algorithms that run fast in many situations. Hertel and Mehlhorn described a sweep-line based algorithm that runs faster the fewer reflex vertices it has [12]. Algorithm's running time is $O(n + r \log r)$ where r denotes the number of reflex vertices of P.

Chazelle and Incerpi also presented an algorithm which time complexity depends on shape of the polygon [4]. 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 motion 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 [20] 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 obtained that share zero edges with the input polygon. The algorithm runs in $O(n^2)$ 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. The Graham scan is a fundamental backtracking technique in computational geometry. In [13] it is 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 of the algorithm is $O(n^2)$, it is easy to implement and therefore is useful in practice.

Finally, in 1991 Chazelle presented $O(n)$ worst-case algorithm [5]. Basic idea is in deterministic algorithm, which computes structure, called visibility map. This structure is a generalization of a trapezoidalization (horizontal chords towards both sides of each vertex in a polygonal chain are drown). His algorithm mimics merge sort. The polygon of n vertices is partitioned into chains of 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 takes actually at most $O(n \log n)$ time. But Chazelle improves 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 linear time. A second phase refines the coarse map into a complete visibility map also in linear time. A triangulation is then produced from the trapezoidation defined by the visibility map. The algorithm has a lot of details and therefore remains open to find a simple and fast algorithm for triangulating a polygon in the linear time.

Table 1 shows all algorithms presented above and is expanded table presented in [17]. Algorithms are grouped by time complexity.

4. Polygon triangulation algorithms based on Delaunay triangulation

Triangulation of the monotone polygons can also be achieved by well-known Delaunay triangulation of a set of points (figure 2a). Namely, the vertices of 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 exists 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 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 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 figure 2b).

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

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.

So, a 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.

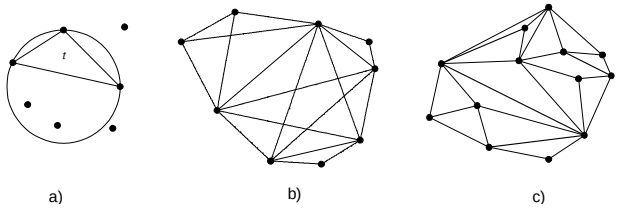

Figure 2: a) Empty circle property; b) Visibility map; c) Constrained Delaunay triangulation

A CDT *T(V; L)* of set of points V with respect to a set of straight-line segments L is a constrained 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 figure 2c) Another characterisation 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 exists any other vertex of T inside the circumcircle of t and visible from all three vertices of t (see figure 2a). See details in [9].

Triangulation of simple polygon can be in generally computed as follows: first step computes CDT of edges of simple polygon and second step removes triangles which are in exterior of simple polygon. The information that input is 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 and algorithms for computing a CDT for general constraint graph.

4.1 Constrained Delaunay triangulation algorithms for simple polygons

Lewis and Robinson describe an $O(n^2)$ algorithm based on divide-and-conquer approach with internal points [15]. 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 optimize d to produce CDT.

A recursive $O(n^2)$ algorithm for CDT based on visibility approach is described by Floriani [8]. The algorithm computes the visibility graph of the vertices of the simple polygon Q in $O(n^2)$ time and the Voronoi diagram 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 neighbours of Q.

Another $O(n \log n)$ algorithm describe Lee and Lin [9]. 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 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's 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 proposed also similar algorithm for general constraint graph which runs in $O(n^2)$ time.

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 [6]. 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 final CDT which is CDT for input graph. The major problem here is merging such strips that contains edges which crosses some strip having no endpoint in it.

Algorithm for computing CDT which includes pre-processing on the constraint segments is proposed by Boissonnat [3]. By pre-processing CDT problem is transformed into standard Delaunay problem on 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 intersect 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.

All mentioned algorithms for CDT demand that all points are defined at the beginning of triangulation. Algorithm proposed by Floriani and Puppo [9] 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 contains the whole domain. In the following incrementally points and line segments are inserted. After each insertion we got new CDT which has more elements than the previous one. After inserting last point or line segment, the bounding triangle is removed. Algorithm runs at most in $O(ln^2)$ where n is number of points and l number of straight-line segments in final CDT.

4.3 Delaunay refinement algorithms

Finally we have to mention the algorithms which cares also about the quality of triangulation. That is the algorithms allow to determine the minimum interior angle of triangles in outputting 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 size triangles in output triangulation (size guarantee).

One of such techniques of triangulation points and line segments 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^o and 120^o. The algorithm produces a uniform mesh to obtain all triangles of the roughly the same size [7].

Ruppert extended Chew's work [18] by giving an algorithm such that all triangles in the output have angles between $\pi - 2a$. Parameter a 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 insure that all input segments are presented in Delaunay triangulation. Two basic operations in the algorithm are: splitting of a segment by adding a vertex at its midpoint, and splitting of 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.

Some other algorithms which give shape guarantees are available. They are more complicated to implement and are not based on Delaunay triangulation. They use such structures as grids and quadtrees. See details in algorithm presented by Baker, Grosse and Rafferty [1] and algorithm presented by Bern, Eppstein and Gilbert [2].

Table 2 shows algorithms for computing triangulation of simple polygon based on Delaunay triangulation. First part of the table shows constrained Delaunay triangulation algorithm, last two lines shows Delaunay refinement algorithms. First part of the table shows algorithms based on Delaunay triangulation without use of Steiner points and second part shows algorithms with use of Steiner points. Parameter M is in that case the number of outputting points.

Time com.	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
$O(M^2)$	Chew	1989	Simple polygon
$O(M^2)$	Ruppert	1994	General

Table 2: Delaunay based triangulation algorithms

5. Properties of some polygon triangulating algorithms

The issue of this chapter is to briefly show the basic properties of triangulation algorithms and difference between them. In that matter we will take some algorithms for triangulating a simple polygon described above.

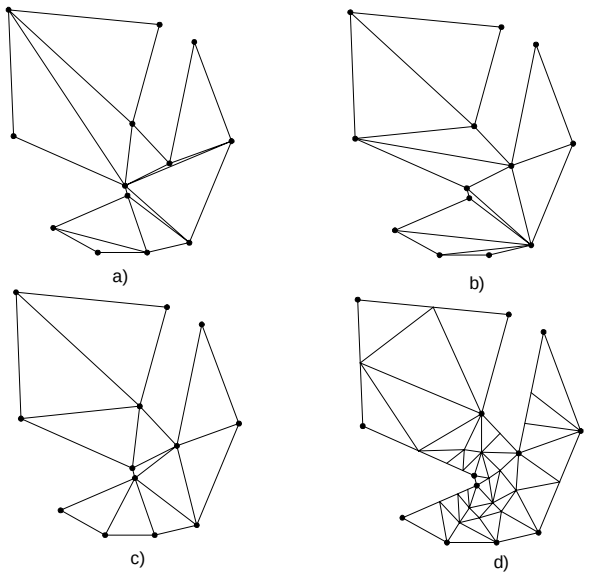

Figure 3: a) Ear cutting; b) Randomized incremental c) Constrained Delaunay triangulation; d) Delaunay refinement

We will try to compare its properties despite sometimes is obvious that difference is present because of different issues of selected algorithms. We will compare attributes as quality of triangulation, asymptotical running time, and the possibility of triangulating polygons with holes. We will rely on each algorithm definitions.
We compared the following algorithms: Meister's ear cutting algorithm [16], Seidel's randomized incremental algorithm [19], constrained Delaunay triangulation [9], Ruppert's Delaunay refinement algorithm [18].

Algorithms based on Delaunay triangulation provides the triangulation with the highest quality of output triangles. Delaunay refinement algorithms are in that way the most quality algorithms because they ensure that the minimum interior angle is as high as possible. Of course, that is possible only with the use of Steiner's points.

Figure 3 shows an example how all four algorithms actually triangulate simple polygon. For that matter we extended the example in [18] with additional triangulations. In figure 3a we can see that ear cutting provides the lowest quality. A better triangulation is provided by randomized incremental algorithm (see figure 3b). Those two algorithms have no mechanism of ensuring quality and therefore is understandable why this low quality. The highest quality (with no using of Steiner's points) provides constrained Delaunay triangulation, but there is impossible to avoid some sliver triangles what we can see in figure 3c. If the Steiner's points are allowed the quality of obtained triangulation is ensured (see figure 3d for Ruppert's Delaunay refinement algorithm).

All algorithms provide triangulating polygons with holes except ear cutting. Ear cutting is designed as recursive algorithm which cuts off ears and therefore can not detect holes. Seidel's randomized incremental algorithm actually wasn't expected to triangulate holes at the beginning but the extension for holes is very simple because of structure of the algorithm. The less problems with holes has Delaunay based algorithms, because in basic inputs for those algorithms are line-segments in the plane. So after triangulation the triangles from holes are removed.

For all four algorithms is known that they are rather simple to implement regarding to other algorithms in the same category. If we look at the worst case time complexity of these algorithms we notice that Seidel's randomized incremental gives us the best results.

If we have to choose from among those four algorithms we will take Seidel's randomized algorithms if we want very fast algorithm. If we need quality in the first place, then we will take Ruppert's Delaunay refinement algorithm. If we need quality but we don't want Steiner's points then perhaps the best choose will be Constrained Delaunay triangulation algorithm.

References

[1] Baker, B., Grosse, E., and Rafferty C. S., *Nonobtuse triangulation of polygons*. Disc. and Comp. Geom., Vol. 3, 1988, pp. 147-168.
[2] Bern, M., Eppstein, D., and Gilbert, J. R., *Provably good mesh generation.* In Proceedings of the 31st Annual Symposium on Foundation of Computer Science, 1990, pp. 231-241 IEEE. To apear in J. Comp. System Science.
[3] Boissonnat, J.D., *Shape reconstruction from planar cross sections*, Comp. Vision Graphics Image Process., Vol. 44, 1988, pp. 1-29.
[4] Chazelle, B., Incerpi, J., *Triangulation and shape complexity*, ACM Transactions on Graphics, Vol. 3, 1984, pp. 135-152.
[5] Chazelle, B., *Triangulating a simple polygon in linear time*, Disc. Comp. Geom. 6, 1991, pp. 485-524.
[6] Chew, L. P., *Constrained Delaunay triangulation*, in Proceedings, Third ACM Symposium on Computational Geometry, Waterloo, June, 1987, pp. 216-222.
[7] Chew, L. P., *Guaranteed - quality triangular meshes*, Technical report, Cornell University, No. TR-89-983, 1989.
[8] De Floriani, L., Falcidieno, B. and Pienovi, C., *A Delaunay-based representation of surfaces defined over arbitrarily-shaped domains*, Comput. Vision Graphics Image Process. 32, 1985, pp. 127-140.
[9] De Floriani, L. and Puppo, E., *An On-Line Algorithm for Constrained Delaunay Triangulation*, Graphical Models and Image Processing, Vol. 54, No. 3, 1992, pp. 290-300.
[10] Garey, M.R., Johnson, D.S., Preparata, F.P. and Tarjan, R.E., *Triangulating a simple polygon*, Inform. Process., Lett. 7, 1978, pp. 175-180.
[11] ElGindy, H., Everett, H. and Toussaint, G. T., *Slicing an ear in linear time*, Pattern Recognition Letters, Vol. 14, 1993, pp. 719-722.
[12] Hertel, S., Mehlhorn, K., *Fast triangulation of simple polygons*, Proc. FCT, LNCS 158, 1983, pp. 207-215.
[13] Kong, X., Everett, H., Toussaint, G. T., *The Graham scan triangulates simple polygons*, Pattern Recognition Letters, Vol. 11, 1990, pp. 713-716, http: //cgm.cs.mcgill.ca/-godfried/research/triangulations.html.
[14] Lennes, N. J., *Theorems on the simple finite polygon and polyhedron*, American Journal of Mathematics, Vol. 33, 1911, pp. 37-62.
[15] Lewis, B.A., Robinson J. S., *Triangulating of planar regions with applications*, Comput. J., Vol. 4, No. 21, 1979, pp. 324-332.
[16] Meisters, G. H., *Polygons have ears*, American Mathematical Monthly, Vol. 82, June/July, 1975, pp. 648-651.
[17] O'Rourke, J., *Computational Geometry in C*, Cambridge University, Press, 1994, pp. 1-65.
[18] Ruppert, J., *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.
[19] Seidel, R., *A simple and fast incremental randomized algorithm for computing trapezoidal decompositions and for triangulating polygons*, Computational Geometry: Theory and Applications, Vol. 1, No. 1, 1991, pp. 51-64.
[20] Toussaint, G. T., *Efficient triangulation of simple polygons*, The Visual Computer, Vol. 7, No. 3, 1991, pp. 280-295.