Ann. Acad. Rom. Sci.
Ser. Math. Appl.
ISSN 066-6594 Vol. 12, No. 1-2/2020

# CONSTRAINING TRIANGULATION TO LINE SEGMENTS: A FAST METHOD FOR CONSTRUCTING CONSTRAINED DELAUNAY TRIANGULATION*

Bozhidar Angelov Stanchev† Hristo Ivanov Paraskevov‡

Dedicated to Dr. Vasile Drăgan on the occasion of his 70th anniversary

**Abstract**

In this paper we present an edge swapping approach for incorporating line segments into triangulation. If the initial triangulation is Delaunay, the algorithm tends to produce optimal Constrained Delaunay triangulation by improving the triangles' aspect ratios from the local area being constrained. There are two types of methods for constructing Constrained Delaunay Triangulation: straight-forward ones which take both points and line segments as source data and produce constrained triangulation from them at once; and post-processing ones which take an already constructed triangulation and incorporate line segments into it. While most of the existing post-processing approaches clear the triangle's edges intersected by the line segment being incorporated and fill the opened hole (cavity) by re-triangulating it, the only processing that our algorithm does is to change the triangulation

*Accepted for publication in revised form on April 15, 2020

†bstanchev@gmail.com University of Shumen, Bulgaria

‡h.paraskevov@shu.bg University of Shumen, Bulgaria; Paper written is partially supported of the National Scientific Program Information and Communication Technologies for a Single Digital Market in Science, Education and Security (ICTinSES), financed by the Ministry of Education and Science.

connectivity and to improve the triangles' aspect ratios through edge swapping. Hereof, it is less expensive in terms of both operating and memory costs. The motivation behind our approach is that most of the existing straight-forward triangulators are too slow and not stable enough. The idea is to use pure Delaunay triangulator to produce an initial Delaunay triangulation and later on to constrain it to the line segments (in other words, to split the processing into two steps, each of which is stable enough and the combination of them works much faster). The algorithm also minimizes the number of the newly introduced triangulation points - new points are added only if any of the line segment's endpoints does not match an existing triangulation point.
**MSC**: 32B25, 42A15, 94A08

**keywords:** Delaunay triangulation, constrained Delaunay triangulation, edge swapping, triangle aspect ratio

# 1 Introduction

The Edge Swapping method [11, 12, 13, 14] (also known as Edge Flipping) is a powerful approach which allows conversions between different triangulations of the same set of points.

Some denotes: Let $P_n = \{p_0, \ldots, p_{n-1}\}$ is a set of distinguishable points on the plane. A partition of the convex hull of $P_n$ into a set of non-overlapping triangles $T = \{t_0, \ldots, t_{m-1}\}$ (which fully cover the convex hull) is called triangulation of $P_n$. The points from $P_n$ are called vertices in terms of the triangulation and the edges of the triangles are called edges of the triangulation. An edge is swappable if it is contained in the boundary $B_{t_i t_j} = t_i \bigcup t_j$ formed by two adjacent triangles $t_i$ and $t_j$ of $T$ which can happen only if this boundary $B_{t_i t_j}$ is a convex quadrilateral (see figure 1). By edge swapping is meant the operation of replacing the existing edge (the existing diagonal in $B_{t_i t_j}$) by the other diagonal of $B_{t_i t_j}$ (see figure 2).

Sleator et al. [14] proved that a transformation of different triangulations of same convex polygon into each other by using edge swapping operations is always possible. Their more important for us result is that if there is at least one edge in a triangulation T1 which can be swapped (is swappable) so that an edge of triangulation T2 is produced, then there is a sequence of edge swapping operations that transforms T1 into T2. Cai and Hirsch [15] extended this results for the triangulations of planar surfaces and showed that the complexity of the edge swapping converting between two triangulations of same set of N points is at most $O(N^2)$. Later on Hurtado et al.

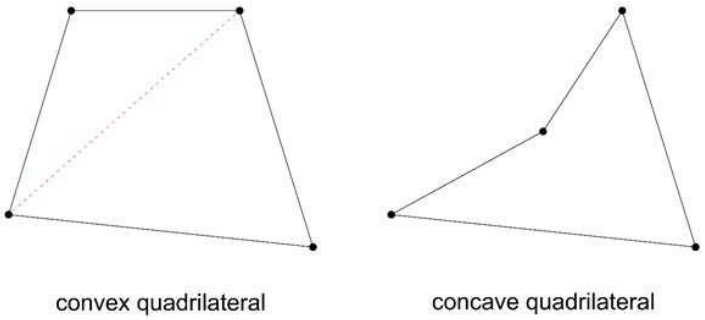


Figure 1: Convex quadrilateral condition for swappable edge.

[11] investigated the edge swapping in triangulations of point sets and they proved that any triangulation of N points in the plane has at least $(N-4)/2$ swappable edges. Thus, every triangulation of at least six distinguishable points has at least one swappable edge. Following the result of Sleator et al. [14], it means that every triangulation of at least six points can be converted to every other triangulation of the same points. This is a promising result we take advantage of: Having a convex triangulation of a set of at least six points, we can always constrain it to a segment specified by any two points from the set.

The Delaunay triangulation [1, 4, 5] of set of positions on the plane has the property that the circumcircle of any Delaunay triangle does not contain any other Delaunay vertices in its interior. It is optimal in the sense that it maximizes the minimum triangle's angle. However, there are often cases when beside positions it is desirable that the produced triangulation also presents a set of predefined segments (connections between positions). Obviously, when triangulating a mixed set of both positions and segments, the resulting triangulation cannot be Delaunay any more. Still, the goal stays the same - to maximize the minimum triangle angle (in other words, to optimize the triangles aspect ratios). Thus we come to the definition of Constrained Delaunay triangulation.

**Definition 1.** *Visibility - two vertices p and q are visible to each other if the line (pq) does not intersect any edge in the triangulation.*

**Definition 2.** *Constrained Delaunay triangulation (CDT) of a mixed set of positions and segments on the plane is such that the circumcircle of any triangle does not contain any other triangulation vertex in its interior which*

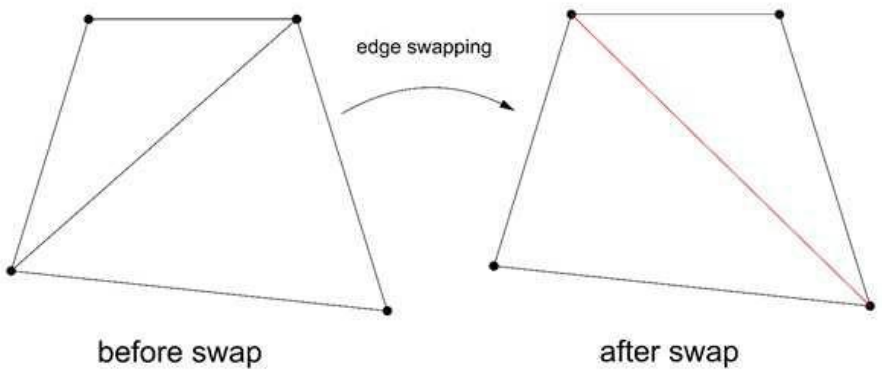


Figure 2: Edge swapping operation.

*is visible from all the three triangle's vertices.*

The motivation behind the CDT methods is their applications in CAD modeling, geographic information systems (GIS), digital terrain modeling (DTM) and others where already exist predefined features coming from the presentation of specific objects (for example, rivers, elevation contours, property lines, roadways, etc.). In general, CDT is appropriate for the cases that require incorporating non-convex boundary, holes and feature lines in the triangulation.

The existing algorithms for constructing CDT could be grouped in different ways. With the "Divide-and-Conquer" CDTs [6] the input set is usually subdivided into smaller sub-sets. Having a small enough sub-set, its triangulation is trivial, but the merging of the triangulated sub-sets into an overall triangulation of the data is complex and resource-consuming. The Sweep-Line CDT algorithms [7, 8] divide the processed area into two sub-areas. The triangulated sub-area gradually grows by integrating elements from the non-triangulated sub-area while keeping the triangulation conforms the Constrained Delaunay requirements. Being simpler to implement, the Incremental CDT methods [3, 2, 9, 10] are the most popular ones. The triangulation is built gradually by inserting new vertices and segments while again keeps the triangulation be conforming the CDT rules. As far as we know, when incorporating a segment all known incremental CDTs proceed as follows: first, they find all the edges (respectively, triangles) intersected by the segment, next, the intersected edges are removed from the triangu-

lation, and finally, the opened hole is re-triangulated and merged into the overall triangulation.

Another grouping of the CDT methods depends on whether the triangulation is constructed by simultaneous use of both the positions and the segments (straight-forward CDT methods), or the process is divided into two steps, where during the first step is constructed a pure Delaunay triangulation of all positions (including the positions of the segments' ends) and during the second step the triangulation is being "constrained" by incremental integration all of the segments one by one in it (post-processing CDT methods). Actually, most of the incremental CDT methods can be implemented as post-processing CDTs. For example, Shewchuk and Brown [16] presented an algorithm for inserting a segment in CDT in time linear in the number of structural changes (i.e. the number of the edges that the segment intersects). Again, their approach first removes the intersected edges and next re-triangulates the opened hole. They analyze the complexity of the operations like: segment location (that precedes the integration of the segment), collecting and removal of the intersected edges and cavity (hole) re-triangulation. Other post-processing approaches are presented by Agarwal et al. [17] and by Domiter [10], both are similarly removing the edges intersected by the segment and re-triangulate the cavity.

Unlike the described approaches, our algorithm does not remove the intersected edges. What it does is to locally modify the triangulation, only by swapping its edges, until the segment is presented by a single edge in it. A simultaneous process improves the aspect ratios of the affected triangles which is again based on edge swapping. There is also another group of triangulation methods which introduce additional regularizing vertices into the triangulation besides the integrated source points and segments. It comes to the so called mesh refinement methods. The additional vertices are integrated so they provide certain triangulation properties. Usually the goal is to produce optimal number of triangles while ensuring they have good (reasonable) aspect ratios. In particular, we are interested in the mesh refinement methods which produce Delaunay triangulation called **Delaunay refinement methods**.

In the pioneer work on the Delaunay refinement Ruppert [18] proposed an algorithm for triangulating a set of both points and segments in the plane which produces Delaunay triangulation where all triangles' angles are between  and -2 where  can be chosen between 0 and 20 degrees. The algorithm's idea is to maintain good triangulation by making local improvements in order to remove the skinny triangles. Each improvement involves adding a new vertex and re-triangulating. The edges sharing a common end-vertex

are split at the same distance from the end-vertex which is a simple solution giving good upper and lower bounds on the output triangles' angles. However, it is not naturally adoptable to curved input (set of curves instead of segments). Pav and Walkington [19] analyze the second splitting approach proposed by Ruppert [18] for the cases of curved input. Their algorithm resembles 2.5D corner lopping, which places a protective ball around the acute corners in the input. An implementation of the Ruppert's Delaunay refinement is proposed by Miller et al. [20]. It gradually adds the source points one-by-one, "opens" the affected triangulation area by removing all the triangles (edges) and re-triangulates the opened cavity.

The idea of the refinement approaches is to improve and regularize the triangulation by introducing additional vertices and edges. In contrast, the goal of our approach is not to improve the triangulation but to ensure that the desired segments are presented in it without introducing any new triangulation elements (whenever it is possible) while keeping the triangles' aspect ratios as good as possible.

## 2 Constraining Triangulation to Line Segments

The motive for our research is to overcome the inconsistencies of the existing CDT implementations and their unsatisfactory work. Most of the existing approaches are quite complex to implement and that is why the developed solutions often fail to deal with specific cases. The specific area of application where we faced problems with some of the existing implementations is the digital terrains modeling. The data from the terrain measurements are usually ordered sets of 3D positions. Also, very often are supplied sets of terrain contours (2D curves or polygons with assigned elevations) extracted directly from digitized maps.

The idea is to avoid using complex 3D Constrained Delaunay tetrahedralization [21] and instead to produce a 2D CDT mesh of the projections of the terrain data (both elevated positions and curves). By elevating the mesh vertices after the terrain elevations is simulated a 3D surface (while in fact it is not). Such surface representation is called **2.5D surface or height field**.

In order to be optimal, the triangulation of the terrain data has to be Delaunay (for the case of 3D positions only) or Constrained Delaunay (for the case of elevated contours). There might be cases when beside contours and/or 3D positions are also provided 2.5D polygons (usually used as terrain surface modifiers). The CDT would be the optimal triangulation of the

source data in each of these cases. The presented approach consists of three steps:

**A. Producing pure Delaunay triangulation from all 3D positions, including the ones that are ends of segments.**

During this step the segments are not only passed to the triangulator but we keep the correspondence between the segments and the positions of their ends in the triangulation. Thus, during the next step, when the segments are being incorporated into the triangulation (the constrain-to-segments step), we do not have to additionally locate the segment position on the triangulation as we already have vertices in the Delaunay triangulation for every segment ending point and pointers to them. This highly speeds up the overall processing.

For producing the pure Delaunay triangulation we use the Shewchuk's algorithm [22] for triangulating of positions only.

**B. Constraining the triangulation to the segments.**

Having every segment defined by two pointers to already existing vertices in the triangulation reduces the constraining task to changing the topology of the existing triangulation. In other words, no new triangulation elements will be introduced (neither vertices not edges). The only processing to do is changing the triangulation topology (the connectivity between its vertices) while, of course, keeping it manifold. This can be fully achieved by performing only edge swapping operations. This is a consequence of the following two propositions: 1) every triangulation of at least six distinguishable points has at least one swappable edge (proved by Hurtado et al. [11]) and 2) if there is at least one edge in a triangulation T1 which is swappable so that an edge of triangulation T2 is produced, then there is a sequence of edge swapping operations that transforms T1 into T2 (proved by Sleator et al. [14]). Hence, what remains to consider is when the triangulation has less than six distinguishable points. The good news is that the pure Delaunay triangulation is always covering the convex hull of the positions. Sleator et al. [14] already showed that for different triangulations of same convex polygon (no matter how many vertices it has) there always exists sequences of edge swapping operations to transform them into each other. This leads to the following lemma:

**Lemma 1.** *Having a convex triangulation, for every two vertices from it there exists a sequence of edge swapping operations transforming it to a triangulation where an edge connecting the two vertices is presented (which is a direct consequence of the above two propositions, first one proved by Hurtado et al. [11] and the second one - by Sleator et al. [14]).*

**C. Constrain-to-Segment algorithm.**

**Step 1:** Find all triangulation edges intersected by the segment.

In order to perform this step fast we need the triangulation data structure to store some additional information - for every triangle we store its three adjacent triangles. This way, once we find that the segment intersects a triangle edge, we already know its adjacent triangle along the intersected edge to go on with. In the adjacent triangle, besides the already intersected edge there are two more edges to be checked for intersection. Actually, this way we always have the direction for searching for the rest of the intersections and what is more, at the end we have the set of the found intersections ordered by the sequence in which the triangulation edges are intersected (means that any two successive intersections from the set are always on two edges of same triangle). Both the fact that we already know the adjacencies between the triangles and that we always know the search direction make this approach very fast with logarithmic complexity (see the work of Devillers et al. [23] for more information about the walking-on-triangulation approaches). Another circumstance that reduces the complexity even further is that our algorithm only needs to know which the edges intersected by the segment are, and doesn't care about the exact intersection positions.

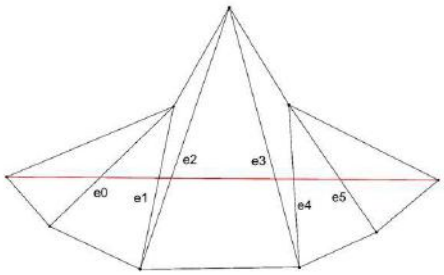


Figure 3: Set of ordered intersections between segment and triangulation edges.

**Step 2:** Edge swapping approach.

This step is the essence of the presented approach. It does not introduce any new triangulation elements (neither vertices nor edges). The only editing operation allowed here is edge swapping which does not add new edges but only changes the connectivity of the old ones. Let the ordered set of intersected by the segment edges be $E_k = \{e_0, e_1, \ldots, e_k\}$. The algorithm works as follows:

1. Find the next swappable edge in the set (figure 4., (2)) and swap it.

2. If all prior intersected edges are marked as 'done' (which means that they are already swapped), mark the lastly swapped again as 'done' (figure 4., (2)). Otherwise, keep the last swapped be 'not done' as it is still intersected by the edge (figure 4., (3) and (4)).

3. If not all prior intersected edges are already swapped (marked as 'done'), then collect the "outer" edges affected by the last swap.

   In other words, if the swapped edge is not intersected by the segment (after the swap), then it affects exactly two "outer" edges (figure 5). These are the two edges that form a triangle with the swapped edge on the opposite side to the segment (they are not visible from the segment and are separated from it by the swapped edge).

4. Also, if not all prior intersected edges are swapped (marked as 'done'), go backwards and swap all of them (figure 4., (5) and (6)).

   The prior non-swappable intersected edges (marked still as 'not done') form a consecutive subset right before the last swapped edge. This subset is processed backwards. The first processed not swapped edge now becomes swappable. Swapping it makes the next processed be swappable as well and so on until all subset edges are passed. This way all the edges from the subset will be swapped so they are not intersected by the segment anymore and will be marked as 'done'.

5. Repeat 1 to 4 until the triangulation is constrained to the segment (all intersected edges are swapped) (figure 4., (7), (8), (9) and (10)).

**Step 3:** Improve the local triangles aspect ratios.

Having collected all affected by the constraining "outer" edges (figure 6) we examine the triangles they share and improve the covered area again only through edge swapping. As a result, the local triangulation affected by the constraining is improved (only by improving the triangles aspect ratios).

To be able to perform this step we need a criteria whether the swapping of an edge will improve the aspect ratios of the paired triangles that share it after the swap. Having it, we iterate the edges. Every processed edge is checked whether swapping it (if it is swappable) will improve the aspect ratios of the shared triangles, and if so, the edge is swapped. Iterative process stops when there are no edges can be swapped any more so the triangles aspect ratios become better.

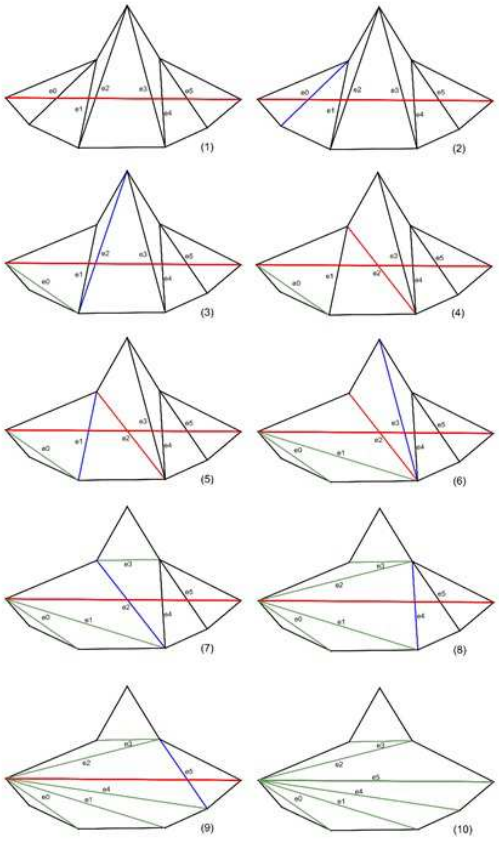


Figure 4: Illustration of the Constrain-to-Segment algorithm based on edge swapping operations only.

**D. Improving triangles aspect ratio of a pair of adjacent triangles.**

In order to improve the local triangulation of the area affected by constraining-to-segment, first of all we have to be able to estimate the aspect ratio of a triangle. The estimation that we use is the ratio between the diameter of the circumcircle around the triangle and the diameter of the inscribed circle in the triangle (figure 7).

Let denote the aspect ratio of a triangle $T$ with $R(T)$. Considering a pair of adjacent triangles $P = \{T_0, T_1\}$, first we check whether the common edge they share is swappable (figure 1) and if so, then there exists a derivative pair

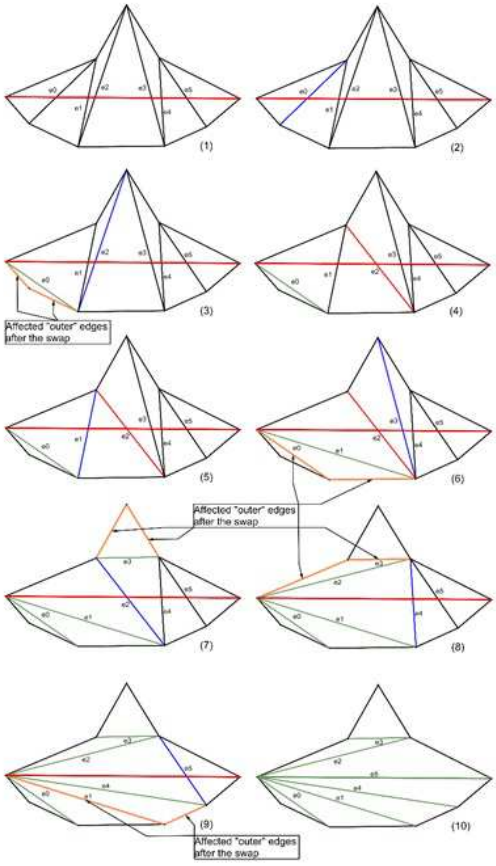


Figure 5: Collecting the "outer" edges affected by the constraining to segment.

of triangles $P^{'} = \{T_0^{'}, T_1^{'}\}$ for the same quadrilateral. The swap improves the aspect rations of the paired triangles if:

$$Max(R(T_0, R(T_1))) > Max(R(T_0^{'}, R(T_1^{'}))) \tag{1}$$

**Lemma 2.** *Having a subset of edges from a triangulation and provided that the local triangulation covered by the triangles sharing the edges from the subset has at least six distinguishable points, the local triangulation can be improved (i.e. its triangles aspect ratios can be improved) by a finite sequence of edge swapping operations.*

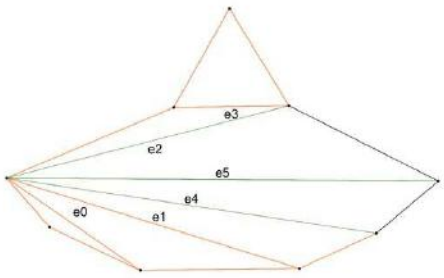


Figure 6: Collected "outer" edges affected by the constraining to segment.

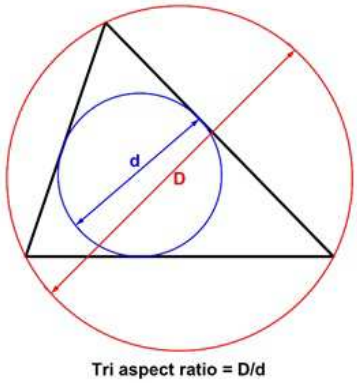


Figure 7: Triangle aspect ratio estimation.

It is again a direct consequence from the work of Hurtado et al. [11] - they proved that every triangulation of at least six distinguishable points has at least one swappable edge and from the work of Sleator et al. [14] who proved that if there is at least one edge in a triangulation T1 which is swappable so that an edge of triangulation T2 is produced, then there is a sequence of edge swapping operations that transforms T1 into T2. In our case: if we have at least one swappable edge witch improves the local triangulation, then there is a sequence of edge swapping operations that optimize it. Otherwise, the local triangulation is already optimal.

## 3 Conclusion

A method for constructing a Constrained Delaunay triangulation of a mixed set of both 3D position and 3D polygons is presented. A segment is incorporated into triangulation in expected $O(m)$ time at worst, where m is the number of the intersected edges by the segment. Further, we present an approach for improving the local triangulation affected by the constraining-to-segment.

## References

[1] Delaunay, B. Sur la sphère vide. Bulletin de l'Académie des Sciences de l'URSS, Classe des Sciences Mathématiques et Naturelles. 6: 793–800, 1934.

[2] Zalik, B., Kolingerova I. An incremental construction algorithm for Delaunay triangulation using the nearest-point paradigm, Geographical information science, vol. 17, 2, pp. 119–138, 2003.

[3] Guibas, L.J., Knuth, D.E., and Sharir, M. Randomized Incremental Construction of Delaunay and Voronoi diagrams. Algorithmica 7: 381–413, 1992.

[4] Guibas, L., and Stolfi, J. Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams,ACM Trans. Graphics, 4(2), pp. 75–123, 1985.

[5] Shewchuk, J. R. Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator, in Applied Computational Geometry: Towards Geometric Engineering, Lecture Notes in Computer Science, Volume 1148, pp 203–222, Springer-Verlag, Berlin, 1996.

[6] Chew, L. P. Constrained Delaunay triangulations, Proceedings of the 3rd annual symposium on Computational geometry, ACM Press, Waterloo, Ontario, Canada, pp. 215–222, 1987.

[7] Fortune, S. A sweep line algorithm for voronoi diagrams, Algorithmica, vol. 2, pp. 153–174, 1987.

[8] Shewchuk, J. R., Sweep Algorithms for Constructing Higher-Dimensional Constrained Delaunay Triangulations, Proceedings of the Sixteenth Annual Symposium on Computational Geometry (Hong Kong), pp. 350–359, Association for Computing Machinery, 2000.

[9] Anglada, M. V., An improved incremental algorithm for constructing restricted Delaunay triangulations, Computers and Graphics, vol. 21, pp. 215–223, 1997.

[10] Domiter, V., Constrained Delaunay triangulation using plane subdivision. in Proceedings of the 8th Central European Seminar on Computer Graphics, pp. 105–110, 2004.

[11] Hurtado, F., Noy, M. and Urrutia, J., Flipping edges on triangulations. Proceedings of Twelfth ACM Annual Symposium on Computational Geometry, pp. 214–223, 1996.

[12] Aichholzer, O., Aurenhammer, F., Krasser, H., Brass, P., Pseudotriangulations from surfaces and a novel type of edge flip. SIAM Journal on Computing 32 (6), pp. 1621–1653, 2003.

[13] Hanke, S., Ottmann, T., Schuierer, S., The Edge-Flipping Distance of Triangulations. Journal of Universal Computer Science, 2, pp. 570–579, 1996.

[14] Sleator, D. D., Tarjan, R., and Thurston, W., Rotation distance, triangulations, and hyperbolic geometry, Proceedings of the 18th Annual ACM Symposium on Theory of Computing, pp. 122–135, 1986.

[15] Cai, J. Y., Hirsch, M., Rotation Distance, Triangulations of Planar Surfaces and Hyperbolic Geometry. Proceedings of ISAAC' 94 (5th International Symposium on Symbolic and Algebraic Computation), Beijing, pp. 172–180. 1994.

[16] Shewchuk, J.R. and Brown, B.C., Fast segment insertion and incremental construction of constrained Delaunay triangulations. Computational Geometry, 48(8), pp. 554–574, 2015.

[17] Agarwal, P.K., Arge, L. and Yi, K., I/O-Efficient Construction of Constrained Delaunay Triangulations. ESA'05 Proceedings of the 13th annual European conference on Algorithms, pp. 355–366, 2005.

[18] Ruppert, J., A Delaunay refinement algorithm for quality 2-dimensional mesh generation. Journal of Algorithms, vol. 18, no. 3, pp. 548–585, 1995.

[19] Pav, S.E. and Walkington, N. J., Delaunay Refinement by Corner Lopping. Proceedings, 14th International Meshing Roundtable, Springer-Verlag, pp. 165-182, 2005.

[20] Miller, G.L., Pav, S.E., Walkington, N.J. An incremental Delaunay meshing algorithm. Technical Report 02-CNA-023, Center for Nonlinear Analysis, Carnegie Mellon University, Pittsburgh, 2002.

[21] Maur, P. Delaunay triangulation in 3D. Technical Report, Department of Computer Science and Engineering, University of West Bohemia, 2002.

[22] Shewchuk, J.R. Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangu-lator. Applied Computational Geometry: Towards Geometric Engineering, Lecture Notes in Computer Science, vol. 1148, pp. 203–222. Springer-Verlag, 1996.

[23] Devillers, O., Pion, S. and Teillaud, M. Walking in a Triangulation. International Journal of Foundations of Computer Science 13, pp. 181–199, 2002.