

# Fast Shape-Preserving Method for Integrating Polygon into Two-and-Half Dimensional Triangulation

Bozhidar Stanchev<sup>1</sup>, Hristo Paraskevov<sup>1</sup>

<sup>1</sup>University of Shumen, Faculty of Mathematics and Informatics, 114 "Universitetska" str, Shumen, Bulgaria

**Abstract** – This paper presents an approach in integrating polygons into a triangulation. The motivation behind this work is to find a way to overcome the lack of appropriate shape-preserving methods for modifying 2.5D triangle meshes.

Widely used approaches for constructing Constrained Delone Triangulation (CDT) work in two steps: first constructing pure Delone triangulation, and next inserting the line segments one-by-one into it [1], [2], [3].

The presented method implements an effective mesh data structure and a walking-on-mesh approach allowing for fast polygon traversal looking for the intersected edges of the 2.5D mesh. Instead of reconnecting vertices or re-triangulating the affected mesh area, we introduce new mesh vertices and subdivide the mesh in order to integrate the polygon. The technique also examines and enhances the aspect ratios of the 2.5D triangles that are present in (and near) the partitioned area while maintaining the shape.

**Keywords** – constrained Delone triangulation, mesh data structure, mesh traversal, edge swapping, triangle aspect ratio.

## 1. Introduction

The motivation behind the Constrained Delone Triangulation (CDT) methods in general and the presented approach in particular is their applications in surface modeling, digital terrain modeling (DTM), CAD applications and other areas that require representation of the features of specific objects (such as the elevation contours from topographic maps, riversides, property lines, roadways, terrain excavations, etc.). The CDT approaches are to make sure that the triangulation topology incorporates certain segments, i.e. to provide a way to integrate into the triangulation the polygons used to model objects features into the triangulation.

Delone Triangulation. The pure Delone Triangulation (DT) [4], [5], [6] can be performed on any set of positions on the plane. Provided there are no coincident positions in the set or each coincidence is considered a single position, it produces a triangulation with the property that there are no other vertices inside of any triangle's circumcircle, thus, it is optimal in that it maximizes the smallest triangle angle.

Constrained Delone Triangulation. In practice, it is often necessary to guarantee topological connectivity, that is, it is preferable for the generated triangulation to provide a set of predetermined segments (links between positions). It is obvious that a pure DT cannot be obtained by triangulating a set of both positions and segments. Still, the aim is the same - to get an optimal triangulation that again maximizes the minimum triangle angle (such that improves aspect ratios of triangles as much as possible) and such triangulation is referred as Constrained Delone Triangulation (CDT). A more exact definition states that the CDT of a mixed set of positions and segments on the plane is such that the circumcircle of no triangle contains any other triangulation vertex in its interior that is visible from all three of the triangles' vertices. Two vertices of a triangulation are visible to one another if the line segment they form does not intersect any of the triangulation's edges.

DOI: 10.18421/TEM121-03

<https://doi.org/10.18421/TEM121-03>


**Corresponding author:** Hristo Paraskevov,  
University of Shumen, Faculty of Mathematics and Informatics, 114 "Universitetska" str, Shumen, Bulgaria  
Email: [h.paraskevov@shu.bg](mailto:h.paraskevov@shu.bg)

Received: 07 October 2022.

Revised: 07 December 2022.

Accepted: 13 January 2023.

Published: 27 February 2023.

 © 2023. Bozhidar Stanchev, Hristo Paraskevov; published by UIKTEN. This work is licensed under the Creative Commons Attribution-NonCommercial-NoDerivs 4.0 License.

The article is published with Open Access at <https://www.temjournal.com/>

## 2. Similar techniques

A variety of CDT algorithms is available. Their conceptual descriptions are provided here only in general terms.

The Divide-and-Conquer CDT methods [7] subdivide the input set of points and segments into smaller sub-sets. The triangulation of a small enough sub-set becomes trivial. However, in doing it, the complexity moves to how to combine the various triangulated sub-sets into a single triangulation, which is subsequently required. Apart from being resource-consuming, the merging is too difficult to implement which makes it the crucial part of such CDT methods.

The Sweep-Line CDTs [8], [9] use a sweep-line traveling from  $-\infty$  to  $+\infty$  (with respect to a chosen direction - usually there is no reason to use a direction other than one of the coordinate axes). First, the input set is sorted according to the chosen sweep-line movement direction. The triangulation below the sweep-line is always complete and as the advancing front moves, the triangulation is being gradually enriched by integrating the next encountered elements while making sure that the triangulation complies with CDT requirements.

The Incremental CDT techniques [10], [11], [12], [13] are popular and frequently utilized in practice since they are easier to implement. Again, new vertices and segments are gradually added to the triangulation while making sure that the CDT criteria are upheld. All known incremental CDTs proceed as follows when incorporating a segment: first, all triangle edges intersected by the segment are detected; next, the intersected ones are removed from the triangulation; and finally, the cavity (hole) that emerges is then re-triangulated and integrated back into the original triangulation.

Depending on how the points and segments are integrated, CDT approaches can be classified in another way. The Straightforward CDT methods construct the triangulation using both the positions and the segments simultaneously. The Post-processing CDT methods split the process into two steps - a first step that produces a pure DT of all points, including the ends of the segments; and a second step that "constrains" the triangulation by gradually integrating all of the segments into it.

Most of the Incremental CDTs are implemented as Post-processing CDTs. The algorithm of Shewchuk and Brown [14] inserts a segment into the CDT in time linear to the number of the necessary structural changes (i.e. to the number of the edges the segment intersects). First, all intersected edges are removed, thus, opening a cavity in the triangulation; next, the opening is re-triangulated and merged back.

The complexity is analyzed taking into account the separate operations (detection of the intersected edges, their removal, re-triangulation of the cavity). Merging of a sub-triangulation back into the whole triangulation rarely adds complexity as, in fact, most often the cavity re-triangulation itself is implemented as hole (cavity) filling directly in the overall triangulation. Other Post-processing CDT methods are presented by Agarwal et al. [2] and by Domiter [13]. Again, they remove the triangle edges intersected by the segment and re-triangulate the hollow that was opened (similarly to the Shewchuk and Brown's approach [14]).

Unlike the described Incremental CDTs, Stanchev and Paraskevov [3] proposed an algorithm that does not remove the triangle edges intersected by the segment. Instead, it locally modifies the triangulation by performing edge-swapping only. Segment integration using this approach is always possible. As demonstrated by Sleator et al. [16], the transition between two different triangulations of the same set of positions is possible through a series of edge-swapping operations.

Frequently, the input set of positions and segments include degenerate triangles or triangles with a poor aspect ratio in their CDT (with nearly collinear vertices). Even in the pure DT of a set of positions (the Delone condition, which states that no triangle's circumcircle contains any more vertices inside of it, is always satisfied), an irregular sampling still could lead to bad aspect ratio triangles. Constraining DT to CDT by integrating the constraints coming from the segments further degrades quality of the triangulation (i.e. further worsens aspect ratios of the triangles).

To improve quality of the constructed CDT (and to overcome the effect described above), there are triangulation techniques that introduce regularizing vertices in addition to the input positions and segments. It comes to the so-called mesh refinement methods. To make sure that the triangulation being built meets specific requirements, regulatory vertices are incorporated. The usual goal is to produce triangles with acceptable aspect ratios throughout the triangulation while keeping the number of triangles as small as possible.

Ruppert's initial research on CDT refinement [17] suggests an algorithm that produces a 2D CDT triangulation of a set of positions and segments while assuring that all triangle angles are between  $\alpha$  and  $\pi - 2\alpha$ , where  $\alpha$  can be between 0 and 20 degrees. The algorithm locally improves the CDT by introducing new vertices in order to eliminate the thin triangles. It splits triangles by introducing a new vertex at the triangle's circumcenter or at the triangle's edge midpoint. Each local improvement step involves both insertion of new vertices and re-triangulation.

A drawback of the proposed approach is that it does not naturally adapt to curved input (set of curves). Pav and Walkington [18] analyze the Ruppert's splitting approach and the possible improvements for the curved input cases and the algorithm they suggest places a protective ball around the acute corners. Boivin and Ollivier-Gooch [19] also propose an extension of Ruppert's refinement for curved input. Another improvement has been proposed by Miller et al. [15] that carries out gradual CDT construction by incrementally adding input points (and segments), "opening" the impacted local triangulation area, eliminating all triangles (edges), and re-triangulating the cavity that opens.

Note that both the 2D and 2.5D CDT methods operate on a planar domain. However, they are used for 3D modeling as well by working on a multi-domain defined by an initial (coarse) 3D mesh where each mesh face defines a planar domain to generate a 2.5D CDT. Thus, in the context of the adjacent domains (the mesh faces), all individual 2.5D CDTs naturally merge into an overall 3D mesh.

As already mentioned, shape preservation is problematic when integrating a line segment into a 2.5D CDT (which defines a 2.5D shape), since the known CDT methods avoid introducing new vertices by relying on changes in the mesh topology or local re-triangulation in order to do the integration, which could, however, greatly affect the shape. Instead of re-triangulating the affected 2.5D mesh area, the method described here subdivides the mesh while introducing new vertices in order to integrate line segment or polygon. Simultaneously, edge-swapping operations are performed to improve the aspect ratios of the locally affected 2.5D triangles provided the shape is preserved.

### 3. Shape-preserving approach for polygon integration

Presented is a 2.5D triangle mesh representation using simple data structure for effective mesh traversal suitable for consecutive searching for the edges intersected by the polygon. Having found a consecutive intersected edge, the triangle mesh is locally subdivided to integrate another polygon point into the triangulation. In doing so, the triangle aspect ratios in the immediate neighborhood of the subdivided triangles are examined and improved where possible using the edge-swapping approach. The search goes on until the entire polygon is integrated into the triangulation.

#### 3.1. Shape-preserving approach for polygon integration

For implementing an effective walking-on-mesh approach, it is necessary that the 2.5D mesh data structure keeps adjacency information between the mesh elements. For the purpose of polygon traversal and integration, the presented method uses a simple data structure.

All 2.5D mesh vertices are stored in vector. Each vertex is represented by three coordinates – two for the position in the 2D domain and one for the elevation. Along with that, the vertex stores an adjacency pointer to any of its adjacent triangles. In C++ terms:

```
struct TINVertex { double x, y; double elevation;
TINTriangle *triangle; };
```

Each triangle keeps adjacency pointers to the three triangles it shares edges with in addition to pointers to its three vertices (i.e. to its three adjacent-along-edge triangles). In C++ terms:

```
struct TINTriangle { TINVertex *vertex0,
*vertex1, *vertex2; TINTriangle *neighbor0,
*neighbor1, *neighbor2; };
```

The sequence of vertices in the triangle structure is important. We have assumed that each triangle must be clockwise relative to the 2D domain (of course, the rule could just as well be that all triangles must be counter-clockwise). The vertices order also determines the order of the edges (implicitly defined by the structure) - we assume that the  $i$ -th edge is defined by the  $i$ -th and  $((i+1) \bmod 3)$ -th vertices. Accordingly, the  $i$ -th edge corresponds to the  $i$ -th neighboring triangle (according to their order in the structure).

#### 3.2 Triangle mesh traversal. Walking-on-mesh along directed segment (directed polygon)

Traversing the 2.5D triangle mesh along a 2D polygon (which is a sequence of consecutively connected directed 2D segments) is performed segment-by-segment, i.e. the traversal algorithm is actually based on an approach for walking-on-mesh along a directed segment, and accordingly, the task itself is reduced to integrating a directed 2D segment into the 2.5D triangulation. For now, we will focus on the integration of a 2D segment. Extending the method to the 3D segment (respectively a 2.5D polygon) integration case is trivial, which will be shown further on.

Should be also noted that as for the walking-on-mesh approach, the point 2D position to its containing mesh triangle and the segment-mesh intersections are being calculated to the 2D mesh (that is, the walking actually works in the 2D domain).

Provided a 2D position falls on a 2D triangle, its elevation on the 2.5D mesh is found by intersecting the vertical ray through the position with the 2.5D triangle's plane (i.e. extending to the walking-on-2.5D-mesh case is quite natural).

Essentially, integrating a directed segment  $\overline{(P_0 P_1)}$  into a 2.5D triangle mesh consists of the following: For the segment starting point  $P_0$  is searched the 2D mesh triangle that contains it. Due to lack of a proper seed triangle at that point, the search starts from a random mesh triangle  $S$  and from a random point  $P_s$  within  $S$  (see figure 1). In terms of the method, the random  $S$  and  $P_s$  are respectively the seed triangle and seed point to start the search with. Empirically we concluded that choosing  $P_s$  to be the center of the circle inscribed in the initial seed  $S$  gives good results (though, no obstacles to be on a seed's vertex or edge).

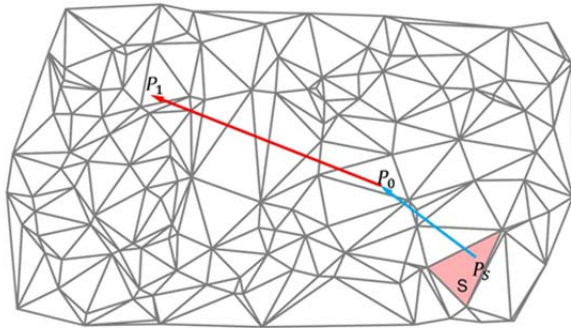


Figure 1. Integrating directed segment in triangulation - seed search settings

Next, the 2D edge of the seed triangle intersected by the newly formed seed segment  $\overline{(P_s P_0)}$  is searched for. There are two possibilities - such intersection  $I$  exists (figure 2, A) or does not (figure 2, B).

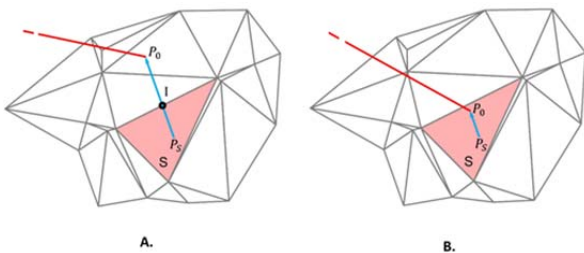


Figure 2. Intersection search between the seed segment and seed triangle - two possibilities

If no intersection is found (case B), it means that  $P_0$  is inside the seed triangle (that is, the containing triangle we are looking for has been already found). When an intersection  $I$  is found (case A), then it becomes the next seed point ( $P_s := I$ ), and the neighbor triangle  $N$  along the intersected edge becomes next seed triangle ( $S := N$ ).

In this case the new seed point is certainly on an edge of the new seed triangle. This edge is transitional from the previous seed triangle to the new one) and therefore it is excluded from further search for intersections (figure 3).

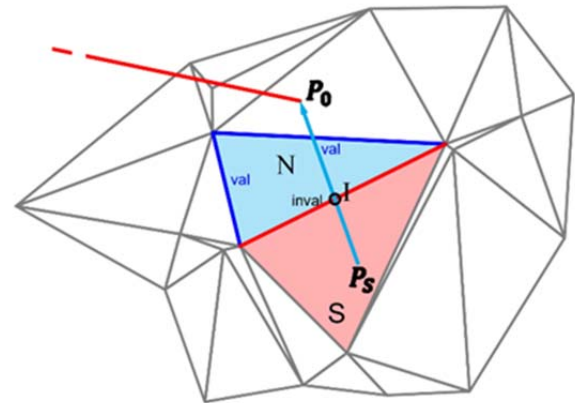


Figure 3. Marching along the intersected edge ( $N$  becomes the next seed triangle,  $I$  becomes the next seed point, marked with "val" are the two edges of  $N$  allowed for the next intersection check)

A specific processing requires the case when the intersection  $I$  is at a vertex of the seed triangle. Again, the intersection  $I$  becomes the next seed point but there is no certainty about which triangle to choose as next seed triangle. To handle this case, all triangles of the immediate neighborhood of the intersected vertex are used (all triangles the vertex is part of). Only their edges that form the outer envelope of this immediate neighborhood are allowed for the next intersection check, whereby the edges of the entry (previous) seed triangle are being excluded (see figure 4).

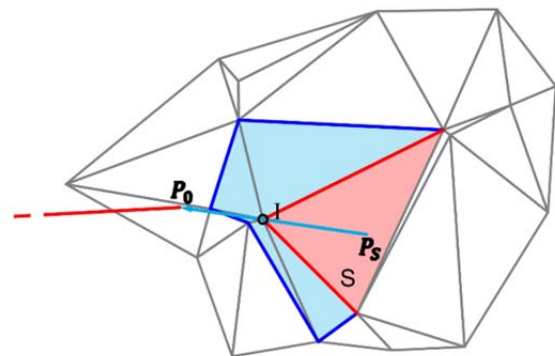


Figure 4. Special case when the intersection  $I$  is at a vertex of the seed triangle

The described stepping from a triangle to its neighboring triangle while following a directed segment is the walking-on-mesh approach in question, key to the method. It is introduced here in the context of the search for the seed triangle that contains the starting point the segment to integrate.

Meanwhile the 2.5D triangle mesh remains unchanged.

Once the seed triangle containing  $P_0$  is found, the same walking-on-mesh approach is used in segment  $(P_0 P_1)$  integration procedure.

Can be easily noticed that the described walking-on-mesh approach requires the triangle mesh to be convex with respect to the 2D domain and segment  $P_0 P_1$  entirely within it. An extension of this approach overcoming these limitations (i.e. one working on randomly positioned segment towards a random concave mesh) is being developed with the intention to be presented in a separate work.

### 3.3. Segment integration

As already shown, the walking-on-mesh along a directed segment consecutively finds intersections with triangle edges. These intersections plus the segment's start and end positions are the points that will be integrated into the triangulation. While integrating into the 2.5D triangle mesh, for each of point-to-integrate needs to have an elevation set (since the walking-on-mesh alone works in the 2D domain).

For each point-to-integrate is known the triangle that contains it (i.e. known are: the seed triangle containing the 2D segment starting position, the triangle whose edge is intersected, and the triangle containing the 2D segment end position when reached). As already mentioned above, having a 2D position inside a 2D triangle, its elevation on the 2.5D mesh can be found by intersecting the vertical ray through the position with the 2.5D triangle's plane. Finding an elevation "projected" onto a 2.5D grid form is not the subject of this work, so we will not go into detail.

The 3D positions are being integrated sequentially as they are found during walking-on-mesh along the segment. Each point-to-integrate can either be fully inside a triangle (can happen for the segment start and end positions) or on a triangle edge (but not at a triangle vertex), that is, the triangle mesh is being subdivided either by inner position or by edge position (see figure 5).

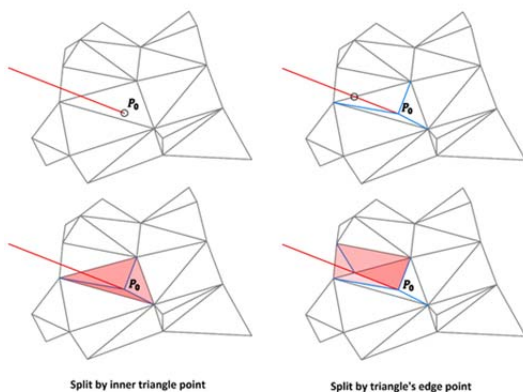


Figure 5. Subdividing the 2.5D triangle mesh approach

Two observations can be made:

The first is that when integrating a 2D segment, all newly introduced vertices and edges conform to the original 2.5D triangle mesh, thus, the mesh shape remains untouched which means that the described integration method is shape-preserving.

The second observation is that the way of introducing new mesh elements does not guarantee that the resulting triangular mesh satisfies the CDT condition of maximizing the minimum triangle angle whenever possible (i.e. more formally, each triangle's circumcircle should not have any other mesh vertex in its interior that is simultaneously visible from any of the triangle's three vertices).

### 3.4. Improving mesh quality while integrating segments

To overcome this shortcoming, our method uses the approach for improving the triangles aspect ratios in mesh area affected by modification presented by Stanchev and Paraskevov [3]. At each segment integration step (that is, each time the triangle mesh is being subdivided to integrate a new vertex, as described above), each newly introduced triangle edge except the one following the segment itself is checked whether edge-swapping it will improve the mesh quality and if so the swapping is performed. In our case, the edge-swapping criterion checks whether the paired 2.5D triangles sharing the edge will have better aspect ratios (in terms of their 2.5D geometry) and, simultaneously, whether the angle between their 2.5D planes will be better (means bigger) after the swap. Figure 6 shows an example of how local improvement of the triangles during segment integration improves mesh quality.

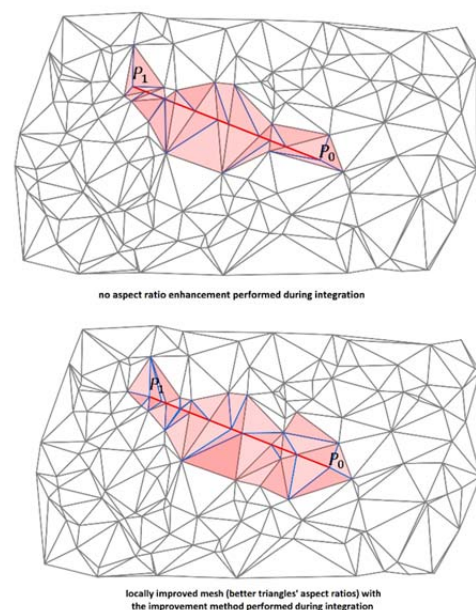


Figure 6. Improving mesh quality using the approach for local improvement of triangles aspect ratios during segment integration

Due to the use of edge-swapping operations, improving the triangles aspect ratios may affect the 2.5D mesh shape. The edge-swapping criterion parameters and their setup are key for obtaining the desired result. In our case, besides the condition for improving triangles aspect ratios, an additional criterion condition is that the swapping of an edge should not produce a sharper mesh edge. An extra control over the edge-swapping would be the adding of an ability to "lock" selected mesh edges to prevent them from swapping. This would make it possible to specify features on the 2.5D mesh by fixing selected mesh edges (i.e. by setting edge constraints). We are doing research in this direction as well and intend to share the possible results in a separate work.

### 3.5. Polygon integration into 2.5D triangle mesh

An oriented polygon is a set of consecutively linked directed segments. In walking-on-mesh terms, the polygon orientation is only necessary to set the traversing direction, the choice of which is not essential for the result of the polygon integration. The difference in the resulting mesh for differently oriented traversal is with precision up to swapped mesh edges, and if triangles aspect ratios are not being improved during the polygon integration, this difference will not affect the 2.5D mesh shape at all.

Polygon integration can be thought of as successive integration of the polygon's directed segments where the end of a segment is the beginning of the next one. That is why the search for the seed triangle is done only once for the starting polygon point only. As shown above, once the integration of a directed segment is completed, the current (running) seed triangle contains the end of the segment which is the beginning of the next one (i.e. the seed triangle to start the integration of the next segment is already known). This significantly speeds up the polygon integration by minimizing the walking-on-mesh processing.

The overall process naturally integrates either random 2D polygons (closed, open and/or self-intersecting, etc.) while preserving the 2.5D mesh shape (see figure 7 and figure 8) or 2.5D polygons (difference from integrating 2D polygon is that a 2.5D polygon comes with defined elevations). Along with that, the method keeps the triangles aspect ratios in the resulting constrained triangle mesh (thus, quality of the result) as good as possible.

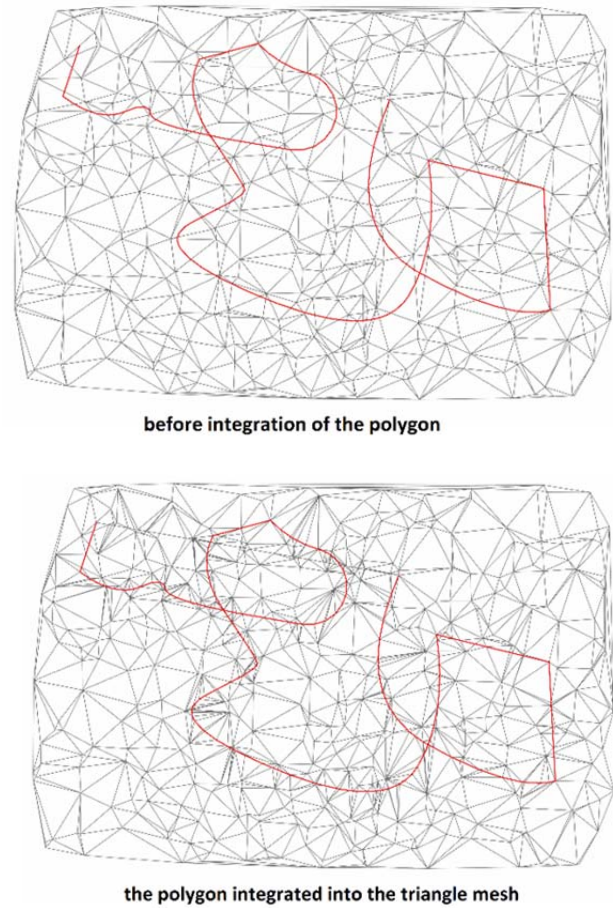


Figure 7. An example of how the method integrates a 2D polygon consisting of 91 segments into a triangle mesh

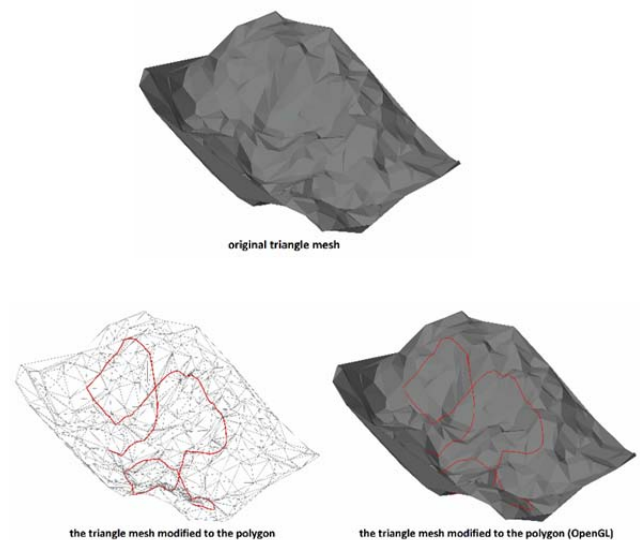


Figure 8. Same example in a 3D view of how the method preserves the 2.5D mesh shape and improves the triangles aspect ratios during the integration

#### 4. Conclusion

A shape-preserving method for integrating polygons into a 2.5D triangle mesh is presented. Integrated in this way, the polygon follows the original mesh's shape quite precisely and acts as a clear area separator. This ability is crucial for delimiting areas of modification and ensuring that adjacent areas are not affected.

The presented technique implements an effective mesh data structure and a walking-on-mesh approach that allows fast mesh traversal along the polygon being integrated. Instead of re-triangulating the affected mesh area, the method enriches the mesh by introducing new mesh elements to integrate the polygon. Additionally, it maintains the shape while maintaining the best aspect ratios for the 2.5D triangles in the affected area which results in a good quality triangle mesh. It also means an optimal number of triangles that is still sufficient for quality surface modeling which is crucial for the conciseness of the mesh representation, as well as the efficiency of any post-processing and analysis.

#### Acknowledgements

The article is partially funded by the project RD-08-147/02.03.2022.

#### References

- [1]. Anglada, M.V. (1997, March). An Improved Incremental Algorithm for Constructing Restricted Delaunay Triangulations. *Computers and Graphics* 21(2), 215–223.
- [2]. Agarwal, P.K., Arge, L., & Yi, K. (2005). I/O-Efficient Construction of Constrained Delaunay Triangulations. In *Algorithms–ESA 2005: Algorithms—ESA 2005, Proceedings of the 13th Annual European Symposium, October 3-6, 2005. Proceedings, 13*, 355-366. Springer Berlin Heidelberg.
- [3]. Stanchev, B. & Paraskevov, H. (2020). Constraining Triangulation to Line Segments: A Fast Method for Constructing Constrained Delone Triangulation. *Mathematics and its Applications: Annals of the Academy of Romanian Scientists*, 12 (1-2). <https://doi.org/10.56082/annalsarscimath.2020.1-2.164>
- [4]. Delone, B.(1934). Sur la sphère vide. *Bulletin de l'Académie des Sciences de l'URSS, Classe des Sciences Mathématiques et Naturelles*. 6, 793–800.
- [5]. Guibas, L., & Stolfi, J. (1985). Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams. *ACM transactions on graphics (TOG)*, 4(2), 75–123. doi: 10.1145/282918.282923
- [6]. Shewchuk, J. R. (1996). Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator. In *Applied Computational Geometry: Towards Geometric Engineering, Lecture Notes in Computer Science, 1148*, 203-222, Springer-Verlag, Berlin.
- [7]. Chew, L. P. (1987). Constrained Delaunay triangulations. In *Proceedings of the 3rd annual symposium on Computational geometry*, ACM Press, Waterloo, Ontario, Canada, 215-222.
- [8]. Fortune, S. (1987). A sweep line algorithm for voronoi diagrams. *Algorithmica*, 2, 153-174.
- [9]. Shewchuk, J. R. (2000). Sweep Algorithms for Constructing Higher-Dimensional Constrained Delaunay Triangulations. In *Proceedings of the Sixteenth Annual Symposium on Computational Geometry*.
- [10]. Zalik, B., & Kolingerova, I. (2003). An incremental construction algorithm for Delaunay triangulation using the nearest-point paradigm, *International Journal of Geographical Information Science*, 17(2), 119-138. <https://doi.org/10.1080/713811749>
- [11]. Guibas, L.J., Knuth, D.E., & Sharir, M. (1992). Randomized Incremental Construction of Delaunay and Voronoi diagrams. *Algorithmica*, 7(1), 381-413.
- [12]. Anglada, M. V. (1997). An improved incremental algorithm for constructing restricted Delaunay triangulations. *Computers & Graphics*, 21, 215-223.
- [13]. Domiter, V. (2004). Constrained Delaunay triangulation using plane subdivision. In *Proceedings of the 8th Central European Seminar on Computer Graphics*, 105–110.
- [14]. Shewchuk, J.R. & Brown, B.C. (2015). Fast segment insertion and incremental construction of constrained Delaunay triangulations. *Computational Geometry*, 48(8), 554-574.
- [15]. Miller, G.L., Pav, S.E., & Walkington, N.J. (2002). *An incremental Delaunay meshing algorithm*. Technical Report 02-CNA-023, Center for Nonlinear Analysis, Carnegie Mellon University, Pittsburgh.
- [16]. Sleator, D. D., Tarjan, R., & Thurston, W. (1986). Rotation distance, triangulations, and hyperbolic geometry. In *Proceedings of the 18th Annual ACM Symposium on Theory of Computing*, 122-135.
- [17]. Ruppert, J. (1995). A Delaunay refinement algorithm for quality 2-dimensional mesh generation. *Journal of Algorithms*, 18(3), 548–585.
- [18]. Pav, S.E. & Walkington, N. J. (2005). Delaunay Refinement by Corner Lopping. In *Proceedings, 14th International Meshing Roundtable*, Springer-Verlag, 165-182.
- [19]. Boivin, C., & Ollivier-Gooch, C.F. (2002). Guaranteed-Quality Triangular Mesh Generation for Domains with Curved Boundaries. *International Journal for Numerical Methods in Engineering*, 55(10), 1185–1213.