Studying the properties of polygon meshes built from Delaunay triangulations

Sergio Salinas[1], Nancy Hitschfeld-Kahler[2], and Hang Si[3]

1,2 Computer Science Department, University of Chile
ssalinas@dcc.uchile.cl,nancy@dcc.uchile.cl
3 Weierstrass Institute for Applied Analysis and Stochastics
si@wias-berlin.de

Abstract

This paper shows a preliminary study on a new kind of polygon meshes obtained from Delaunay triangulations. To generate each polygon, the algorithm starts by generating a Delaunay triangulation of a point set; second it builds polygons (simple or not) from terminal-edge regions, third it transforms each non simple polygon into simple ones, convex or not convex. Both types are permitted. We analyse which kind of non simple polygons appear and show the algorithm to divide them into simple ones. Some empirical properties of the resulting meshes are shown.

1 Introduction

Meshes based on triangles and quadrilaterals are common in simulations using Finite Element Method (FEM). The problem is that polygons(elements) in FEM need to obey specific quality criteria, such as to avoid too large obtuse angles or too small angles, to have sides of graded length (aspect ratio criteria), etc. To fulfill these criteria, sometimes the insertion of a large number of points and elements is required in order to properly model a domain, increasing the time needed to make the simulation. New methods as Virtual Element Method (VEM) can use any polygon as basic cell. So, (i) the domain geometry can be fitted using less elements than if only triangles and quadrilaterals are used, (ii) the required point density distribution is just the required by the simulation problem, and (iii) it should not be necessary to improve further the quality of the elements. Currently, it is a research topic how far the VEM can allow the simulation of more complex problems, both in 2D and 3D, with respect to FEM [5].

In this paper we propose an algorithm to generate meshes with polygons of arbitrary shape (convex and non-convex) based on the label system described in [2]. These meshes might contain non-simple polygons, so we also propose an algorithm to repair them using the maximum area criterion per polygon. At the end, we do an experimentation to show the properties of these polygons, and compare these meshes with Voronoi meshes.

Our motivation is the generation of meshes that adapt to a geometric domain using polygons of any shape, but respecting the required point density. To do this, we propose to use the concept of terminal-edge region. Our main research questions are: Can terminal-edge regions be adapted to be used as good basic cells for polygon numerical methods? Do these kind of meshes need less polygons to model the same problem than polygon meshes based in the Voroni diagrmam?

This paper is organized as follows: Section 2 introduces the basic concepts to understand the algorithm; in Section 3 there is a brief description of the algorithm divided in 3 main phases. Section 4 shows a preliminary experimental evaluation of the generated meshes on random points, some properties of the polygons and a comparison with Voronoi based meshes. Section 5 shows the ongoing work.

Extended abstract submitted to the 37th European Workshop on Computational Geometry, St. Petersburg, Russia, April 7–9, 2021. This abstract describes an ongoing work that when it is finished it will be sent to a journal or conference.

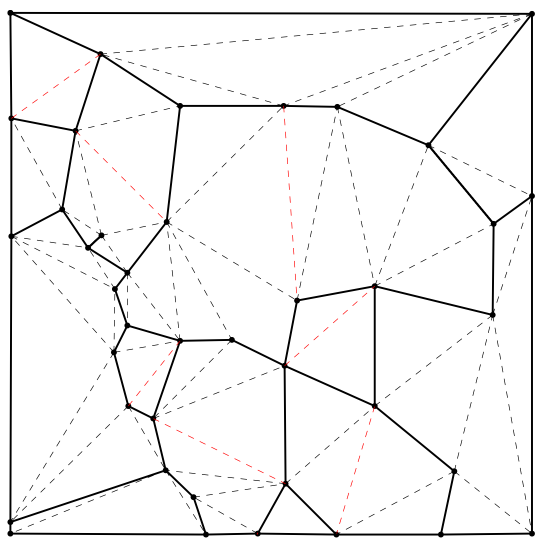


Figure 1 Terminal-edge regions: Solid lines are frontier-edges, dashed black lines are internal-edges and red dashed edges are terminal-edges.

2 Preliminaries

The original algorithm is based in two important concepts, Longest-edge propagation path (Lepp) introduced in [3] and terminal-edge regions defined in [1]. We will use these terminal-edge regions as initial polygons.

▶ **Definition 2.1. Longest-edge propagation path** [3]: For any triangle t_0 in any triangulation Ω, the Lepp(t_0) is the ordered list of all the triangles t_0 , t_1, t_2, ..., t_{l-1}, t_l, such that t_i is the neighbor triangle of t_{i-1} by the longest-edge of t_{i-1}, for $i = 1, 2, \ldots, l$. The longest-edge shared by t_{l-1} and t_l is a terminal-edge and t_{l-1} and t_l are terminal-triangles.

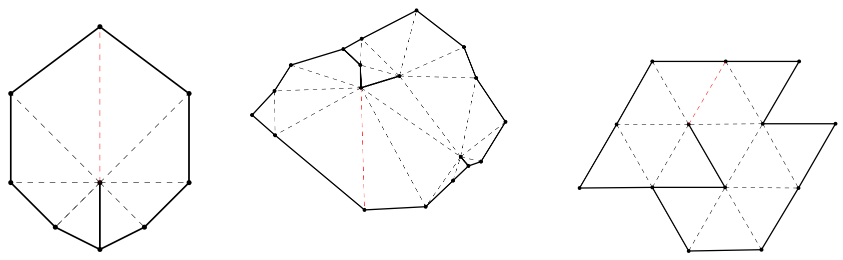
▶ **Definition 2.2. Terminal-edge region:** A *terminal-edge region* [1] R is a region formed by the union of all triangles t_i such that Lepp(t_i) has the same terminal-edge. In case the terminal-edge is a boundary-edge the region will be called *boundary terminal-edge region*. For an illustration, see Figure 1, each solid line delimits a terminal-edge region.

Given an edge $e \in \Omega$ and two triangles t_1, t_2 that share e, we can label e as:

- Terminal-edge [3], if e is the longest-edge of t_1 and t_2.
- Frontier-edge, if e is neither the longest-edge of t_1 nor t_2.
- Internal-edge, if e is the longest edge of t_1 but not of t_2 or vice-versa.

For the simplicity, in the case e is a domain boundary edge, e will be considered a frontier-edge too (see Figure 1). In [1, 2] it was proven that terminal-edge regions cover the whole domain without overlapping. Non-simple polygons appear when a frontier-edge is shared by two triangles belonging to the same terminal-edge region. We call this kind of frontier edge a **Barrier-edge**.

▶ **Definition 2.3. Barrier-edge tip:** A barrier-edge tip in a terminal-edge region R is a barrier-edge endpoint shared by no other barrier-edge. See Figure 2.



(a) One barrier-edge and one tip (b) Four barrier-edges and two tips (c) Two barrier-edges and one tip

Figure 2 Examples of non-simple polygons with barrier-edges. Black lines are frontier-edges, dashed black lines are internal-edges and red edges are terminal-edges.

▶ **Lemma 2.4.** *Each vertex of a terminal-edge region is endpoint of a frontier-edge.*

Proof by contradiction: Let v be a vertex of a terminal-edge region R generated by the terminal-edge e and T the set of the triangles that contains v. Let assume that v is not an endpoint of a frontier-edge as shown in Figure 3.

As all triangles in T are part of R, they share their longest-edge, around v, v interior point. Due to T is finite, there should exist a triangle t_0 (see Figure 3) that shares their longest-edge with two triangles of R in order to maintain v interior in R. This is not possible because a triangle has just one edge labeled as its longest-edge. This contradicts our assumption, so v has to be endpoint of som at least one frontier-edge.

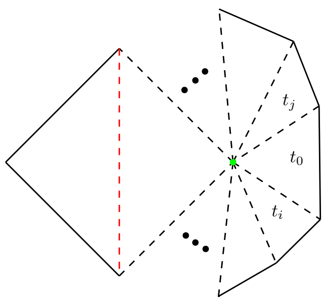


Figure 3 Green vertex is not part of a frontier-edge.

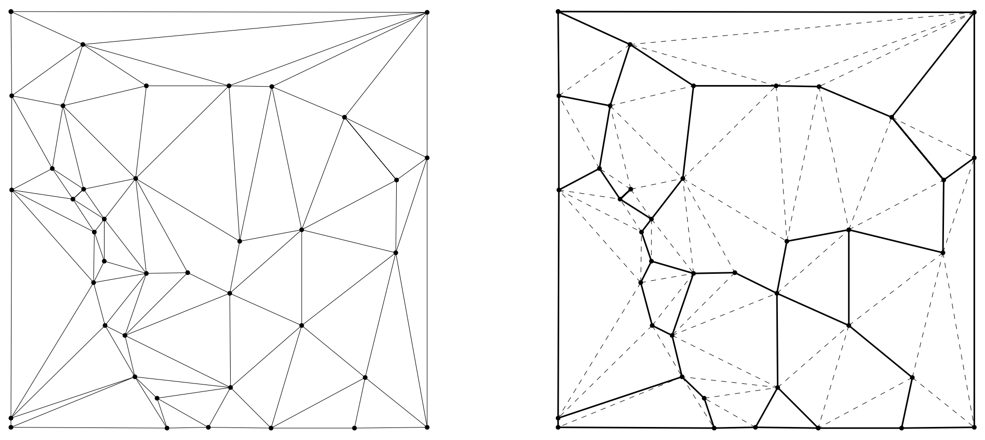
Lemma 2.4 is very important since it means that the initial points used to represent the geometric domain and the ones inserted to fulfill points density requirements are maintained after the polygon mesh is generated. Also, it allows to use barrier-edges to splits polygons by just adding one edge of the triangulation to them.

3 The algorithm

The following algorithm can be applied to any triangulation but because of their well-known properties we decided to study first the kind of polygons obtained from Delaunay triangulations. The algorithm is divided in 3 main phases. In the first one it labels each edge of the triangulation Ω as terminal-edge, internal-edge or frontier-edge. In the second one it selects a triangle with a frontier edge and uses this as starting triangle to build the polygon border, and in the third phase it repairs polygons with barrier-edges by splitting them.

3.1 Label Phase

The label phase was proposed in [2]. Tt consists in two cycles: the first cycle marks each longest-edge in each triangle and the second, labels which edge is not the longest-edge of both triangles (frontier-edges). See Figure 4. To Label all edges has a cost of $O(n)$, with n the number of triangles.



(a) Delaunay Triangulation

(b) Triangulation after applying the label phase

Figure 4 Label phase: the solid lines show the frontier-edges and dashed lines the rest.

3.2 Travel phase: Polygon building

In this phase polygons are recognized and represented as a closed polyline. For any non-visited terminal-edge region, the algorithm selects a triangle as a seed and travels through adjacent triangles storing each frontier-edge until each polygon is built.

By lemma 2.4 there are no interior vertices; all triangles have at least one vertex which is end-point of a frontier-edge. So the algorithm travels through all triangles in a terminal-edge region without getting stuck in a infinite loop. To build all closed polylines has a cost of $O(n)$. This phase is shown in Figure 4.

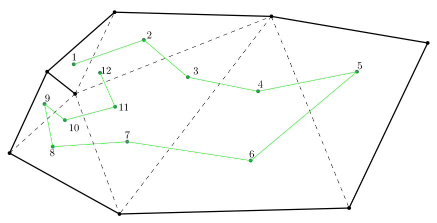


Figure 5 Travel of the algorithm inside a terminal-edge region. Note that triangles with no frontier-edge are visited 3 times, with 1 frontier-edge 2 times and with 2 frontier-edges 1 time.

3.3 Non-simple polygon reparation

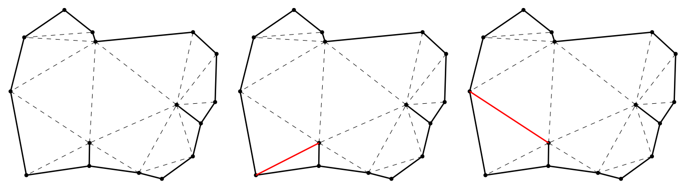
To repair non-simple polygons, the algorithm uses the barrier-edges and internal-edges of the triangulation to split them into simple ones. So, we avoid to check intersections between inserted edges and the polygon boundary. In order to decide which internal-edge is the best, we use the maximum area criterion because the resulting polygons are the largest ones as possible. Let A be the area of the non-simple polygon and b the number of its barrier-edges tips. Let A_1 and A_2 be the area of the two polygons generated after the split. We define:

▶ **Definition 3.1.** Optimal Area (A_{opt}): The largest area that each polygon after the split could have. It is calculated as the division between A and number of polygons needed to eliminate all barrier-edges tips $(b + 1)$

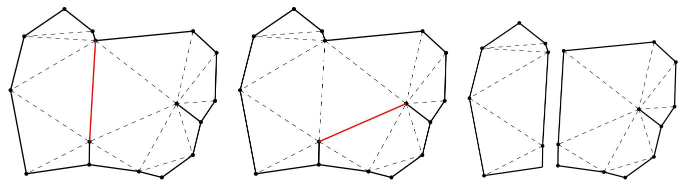
▶ **Definition 3.2.** Optimal Area difference (ΔA): This value is used to estimate how close is the algorithm to find the right split. It is calculated as:

$$\Delta A = |min(A_1, A_2) - A_{opt})|, \;\; A_{opt} = \frac{A}{b+1}$$

Given a polygon P to split and a barrier-edge tip v. The algorithm computes the optimal area difference ΔA of each internal-edge incidents to v in clockwise order as shown in Figure 6. The internal-edge to split the polygon is chosen when A gets an inflection point in its value.



(a) $A = 30$, $A_{opt} = 10$ (b) $A_1 = 3$, $A_2 = 27$, $\Delta A_{act} = 7$ (c) $A_1 = 5$, $A_2 = 25$, $\Delta A_{act} = 5$



(d) $A_1 = 11$, $A_2 = 19$, $\Delta A_{act} = 1$ (e) $A_1 = 22$, $A_2 = 8$, $\Delta A_{act} = 2$ (f) $A_1 = 11$, $A_2 = 19$, $\Delta A_{act} = 1$

Figure 6 Optimal split for a polygon with 2 barrier-edges. The value Optimal Area difference ΔA_{act} decreases from the steps b) to e) and increases in the step e), so the optimal area has been bypassed and the red edge in d) is chosen to split the polygon.

This split is a recursive function that is called for each polygon with barrier-edges tips until barrier-edges tips are no longer present. Let k be the number of triangles in a polygon with b barrier-edge tips and; both to calculate ΔA and split a polygon has cost $O(k)$. The algorithm has a complexity cost of $O(k \cdot b)$

In the worst case scenario, all polygons have barrier-edges; the time complexity is $O(n \cdot B)$, with n the number of triangles of the initial triangulation and B the number of barrier-edges tips in the mesh. But, as we shown in the next section, this case rarely happens when a Delaunay triangulation is used as initial mesh.

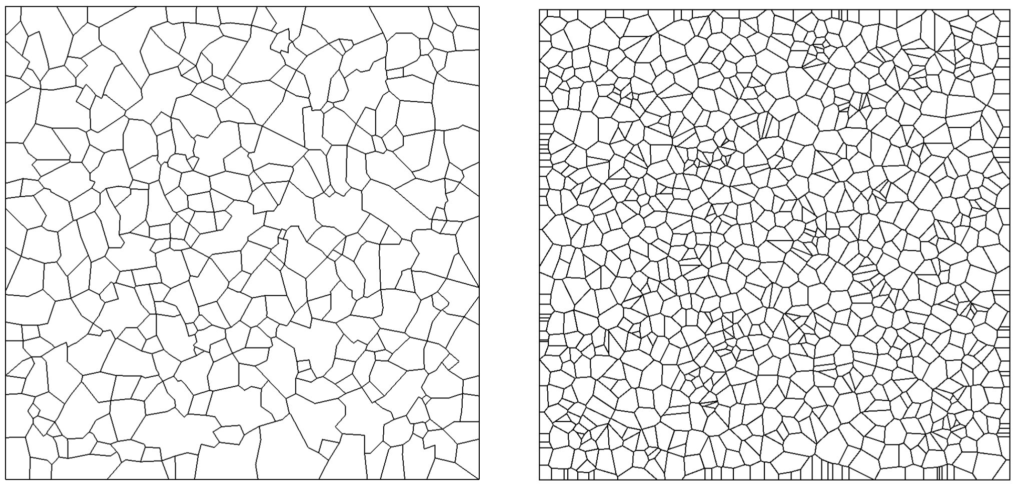

(a) Polygon Mesh (b) Voronoi diagram

Figure 7 **(a)** Polygon mesh generated by 1000 random points. **(b)** Voronoi diagram of the same 1000 random points generated with Detri2QT [4].

4 Experimentation

Sites	Number Triangles	Number Regions	Terminal-edge Regions	Sites per poly	Triangles per poly	Edges per poly	Total bet†	Max bet† in polygon
10	14	4	4	2.50	3.50	5.50	0	0
10^2	189	30	29	3.33	6.30	8.30	1	1
10^3	1869	307	290	3.26	6.09	8.09	17	2
10^4	18835	2886	2724	3.47	6.53	8.53	164	3
10^5	188876	28698	27045	3.48	6.58	8.58	1716	4
10^6	1888472	286479	269253	3.49	6.59	8.59	17877	5

Table 1 Preliminary empirical evaluation. [1]barrier-edge tips

The input geometry can be defined by the convex hull of random point sets or using a PSLG specification. The Delaunay triangulations were generated using the Detri2 library [4]. Statistical data of the polygon meshes generated from random points are summarized in Table 1. We can observe that after 10^4 input points (sites), the number of triangles per polygon is in average 6.5 and the number of edges per polygon is 8.5. The number of barrier-edges is less than 1% of the number of sites, so the reparation phase just add $\approx 0.5\%$ of polygons to the mesh. If we compare these polygon meshes with the meshes generated from the Voronoi diagram, the Voronoi meshes contains 3 times more polygons than the our meshes. In average, each Voronoi region is formed by 6 edges and the our polygons by 8 edges. Figure 7 shows at the left, the polygon meshes generated by the proposed algorithm and at the right, the mesh based on the Voronoi Diagram.

5 Conclusions and Ongoing work

We have presented a preliminary formalization and statistical evaluation of a new kind of polygon meshes based on terminal-edge regions in order to explore the geometric properties of the generated polygon cells. We have observed that this kind of meshes contains three times less polygons that the standard polygon meshes based on the Voronoi diagram. It is still an open question if these meshes will be as useful as Voronoi meshes to solve numerical problems using the VEM. Our ongoing work is to complete the theoretical formulation and validate this kind of meshes in real applications.

References

1 R. Alonso, J. Ojeda, N. Hitschfeld, C. Hervías, and L.E. Campusano. Delaunay based algorithm for finding polygonal voids in planar point sets. *Astronomy and Computing*, 22:48 – 62, 2018.

2 José Ojeda, R. Alonso, and N. Hitschfeld-Kahler. A new algorithm for finding polygonal voids in delaunay triangulations and its parallelization. In *The 34th European Workshop on Computational Geometry, EuroCG*, pages 56:1–6, 2018.

3 M. C. Rivara. New longest-edge algorithms for the refinement and/or improvement of unstructured triangulations. *Int. Jour. for Num. Meth. in Eng.*, 40:3313–3324, 1997.

4 Hang Si. An introduction to unstructured mesh generation methods and softwares for scientific computing. Course, 7 2019. 2019 International Summer School in Beihang University.

5 Peter Wriggers, Fadi Aldakheel, and Blaž Hudobivnik. Application of the virtual element method in mechanics. Technical report, Report number: ISSN 2196-3789. Leibniz Universität Hannover, 01 2019.