Finding a Triangular Mesh with a Constant Number of Different Edge Lengths

Shin-ichi Tanigawa* Naoki Katoh†

Abstract

We consider the problem of triangulating a simple polygon with a small number of different edge lengths using Steiner points. Given a parameter α, where $0 < \alpha < 1$, we shall present an algorithm for finding an almost uniform triangular mesh with $\frac{3\pi}{8\alpha^2} + o(\frac{1}{\alpha^2})$ different edge lengths such that every edge length is between $\underline{l}$ and $2\underline{l}$. We shall also give experimental results of this algorithm.

1 Introduction

The problem of triangulating a specified domain has various applications such as computer aided design or finite element methods, and has been extensively studied; see e.g. the survey article by Bern and Eppstein [3].

In the field of architecture, triangular mesh is often used for structure truss such as a dome to cover a huge space. In such a large scale structure, the lengths of members and angles between consecutive members incident to a joint are critical properties that determine the structural performance. Furthermore, a cost to realize such structure must be also evaluated. Construction costs heavily depends on the number of the different member lengths. From this standpoint, we are concerned with how to realize a triangular mesh with a limited number of different elements.

In this paper, we present a heuristic for constructing a triangular mesh which consists of almost the same edge lengths and no small angle using a constant number of different edge lengths.

The following notation will be used throughout. For two points x and y on the plane, let $d(x, y)$ denote their Euclidean distance. The minimum (non-zero) distance between two point sets X and Y is defined as $d(X, Y) = \min\{d(x, y) \mid x \in X, y \in Y, x \neq y\}$. The problem is defined as follows.

Input: A simple polygon P whose vertex set V_P is such that $\underline{l} \leq d(V_P, V_P) \leq 2\underline{l}$, and β is a parameter between $\sqrt{3}$ and 2 which controls the edge lengths of the approximated polygon.

Output: A simple polygon Q which approximates P, and triangulation of Q such that the edge lengths and the angles are uniform and the number of different edge lengths is constant.

Since it is impossible in general to find a triangulation for P such that the number of different edge lengths is constant, we first appropriately approximate P by a simple polygon Q so that all vertices of polygon P are moved to points of a square grid of width $\alpha\underline{l}$, where $0 < \alpha < 1$ and $\underline{l}$ is predetermined value. This implies the set of possible edge lengths produced becomes constant, if all edge lengths are bounded. Our algorithm then finds a triangulation T of Q such that (i) every edge of T is between $\underline{l}$ and $(2 + \sqrt{2}\alpha)\underline{l}$, and (ii) the number of different edge lengths is $\frac{3\pi}{8\alpha^2} + o(\frac{1}{\alpha^2})$, where α is a parameter between 0 and 1 which controls the number of different edge lengths. And then, the angles of a triangulation T are at least $\min\{\arccos\frac{\beta}{2}, \arcsin\left(\frac{1}{2+\sqrt{2}\alpha}\right)\}$ if the approximated polygon Q satisfies $\underline{l} \leq d(Q, Q) \leq \beta\underline{l}$, where $\sqrt{3} \leq \beta \leq 2$.

We use the notation G and V_G to denote square grid of width $\alpha\underline{l}$ and a set of grid points respectively.

2 Approximation of the boundary

In this section, we consider the problem of approximating the boundary by using points of a square grid of width $\alpha\underline{l}$. We need to introduce the function which measures the error between the original polygon and the approximated one. In this paper, we consider the minimization of Hausdorff distance defined as follows.

The Hausdorff distance between two polygons P and Q is given by

$$H(P, Q) = \max(h(P, Q), h(Q, P))$$

where

$$h(P, Q) = \max_{a \in P} \min_{b \in Q} d(a, b).$$

See [1] for more details about Hausdorff distance.

The problem is to find an approximated polygon Q such that Hausdorff distance with a given polygon P is minimum satisfying every vertex is in V_G and every edge length is between $\underline{l}$ and $\beta\underline{l}$. The algorithm of polygon approximation consists of three steps. In the first step we enumerate candidate grid points. Candidate grid

*Department of Architecture and Architectural Engineering, Kyoto University is.tanigawa@archi.kyoto-u.ac.jp

†Department of Architecture and Architectural Engineering, Kyoto University naoki@archi.kyoto-u.ac.jp

points are defined as $C = \{p \mid d(p, P) \leq \varepsilon, p \in V_G\}$, where ε will be determined later. Next we construct a network containing geometric information, and then we solve the bottleneck shortest path problem.

Candidate grid points. For the computational efficiency, it is necessary to enumerate candidate grid points. From the following lemma, we set ε to $\sqrt{6}\underline{l}$.

Lemma 1 *Let $\tilde{Q}$ be the approximated polygon which minimizes Hausdorff distance. Then, $h(\tilde{Q}, P)$ is less than $\sqrt{6}\underline{l}$.*

Consider finding all the candidate points on a line $l : x = i\alpha\underline{l}$, where i is an integer, we take the Voronoi-diagram of P, $Vor(P)$. Suppose that a point p moves upward on l. Then the distance function between p and P with respect to the y-coordinate of p is unimodal in each Voronoi cell. Therefore, we can get candidate grid points in each Voronoi cell in time proportional to the number of candidate points in the Voronoi cell.

The number of Voronoi cells which intersect l is $O(|V_P|)$, and each of such Voronoi cell contains $O(\varepsilon/\alpha\underline{l})$ candidate grid points. Therefore, the number of candidate grid points on l is $O(\varepsilon|V_P|/\alpha\underline{l})$. Total number of lines to be considered is $O(|V_P|/\alpha)$ since horizontal width of the original polygon is $O(\underline{l}|V_P|)$. Therefore, total number of the candidate grid points is $O(\frac{\varepsilon|V_P|^2}{\alpha^2\underline{l}}) = O(\frac{|V_P|^2}{\alpha^2})$.

By more precise analysis, we get that the candidate grid points can be reduced to $O(\frac{|V_P|}{\alpha^2})$.

Constructing a network. We create a node in the network associated with every candidate point p. If points p and q satisfy $\underline{l} \leq d(p, q) \leq \beta\underline{l}$, we draw an edge from p to q. Fig.1 shows the edge weight of pq. Let p' and q' be the nearest points from p and q respectively, and let $p'a_1a_2\dots a_kq'$ be a polyline from p' to q' of P. The edge weight of pq is defined as the maximal value of $d(p, p')$, $d(q, q')$ and $d(a_i, pq)$ for all $i = 1, 2, \dots, k$, (see Fig.1).

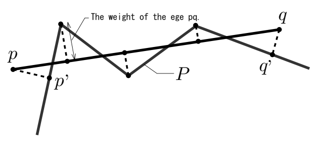

Figure 1: The edge weight of Hausdorff distance minimization problem between p and q

From Lemma1, there is at least one vertex of the optimal solution in the circle with radius $\sqrt{6}\underline{l} + \delta$ ($\delta > 0$) centered at a point of P, where δ can be arbitrarily small. Therefore, we choose an arbitrary vertex v of P and consider the set V_{start} of candidate points in the circle with radius $\sqrt{6}\underline{l}+\delta$ centered at v as starting nodes set of a network. We then create a copy of V_{start} as a set of terminal nodes. Then, the path from a start node to a terminal one corresponds to a sequence of edges of a polygon which approximates P. The optimal approximate polygon Q can be found by computing the bottleneck shortest path from a starting node to a terminal node on the network such that the maximum weight of the path is minimum. It can be obtained by a variant of the standard Dijkstra method.

Computational complexity for polygon approximation. The total number of nodes which associated with candidate points is $O(|V_P|/\alpha^2)$. Number of edges of the network incident to one node is $O(1/\alpha^2)$. Therefore, the total number of edges is $O(|V_P|/\alpha^4)$. For each starting node of V_{start}, it takes $O\left(\frac{|V_P|}{\alpha^4} + \frac{|V_P|}{\alpha^2}\log\frac{|V_P|}{\alpha}\right)$ time to get a shortest path. Therefore the total time to get the optimal solution is $O\left(\left(\frac{|V_P|}{\alpha^4}\right)\left(\frac{1}{\alpha^2} + \log\frac{|V_P|}{\alpha}\right)\right)$.

Optimal grid layout. The method that we stated above produces a different solution depending on a position of the grid. If we slightly move the grid, $H(P, Q)$ may change. Suppose that the output polygon Q is already determined. This leads to the problem of finding an optimal grid layout which minimizes $H(P, Q)$, which can be treated as a matching problem between two polygons. An $O((m + n)^6 \log^2 mn)$ algorithm is known for optimal matching under rigid motion by Chew et al.in [5], where m, n are the members of vertices of two polygons. Even after an optimal grid layout is found, the optimal approximate polygon for P may be different from Q for this grid layout. Thus, we have to repeat this process until the solution converges. We do not have any idea currently about how both shape and grid layout can be optimized simultaneously.

3 Incremental Voronoi partition with square grid

In this section we present an algorithm for finding a uniform triangulation with a constant number of different edges. Our method is based on the incremental Voronoi partition used in [2, 6, 7].

Let Q be the approximate polygon, V_Q be a point set of Q, and S be a set of Steiner points set. Let p_i, for $i = 1, \dots, 4$, denote vertices of a grid square which contains a vertex p.

Algorithm

Step 1: Initialize $S := \emptyset$

Step 2: Compute the Voronoi diagram $\mathrm{Vor}(V_Q \cup S)$.

Step 3: Find a point p_i^* which is one of the vertices of the grid square containing a Voronoi vertex p of $\mathrm{Vor}(V_Q \cup S)$ and is in the interior of Q such that $d(p_i^*, V_Q \cup S)$ is maximum.

Step 4: If $d(p_i^*) \geq \underline{l}$, let $S := S \cup p_i^*$ and return to Step 2. Else go to Step 5.

Step 5: Output the constrained Delaunay triangulation for $V_Q \cup S$

We shall analyse the algorithm by following the ideas in [2, 6, 7]. Consider the Delaunay triangulation $\mathrm{DT}(V_Q \cup S)$. Let us classify a triangle $\triangle$ of $\mathrm{DT}(V_Q \cup S)$ as either *critical* or *non-critical* depending on whether the circumcenter of $\triangle$ is lies outside of the polygon Q or not. Critical triangles occur close to the boundary. Consider some edge e of $\mathrm{DT}(V_Q \cup S)$ on the boundary of Q. Edge e cuts off some part of the Voronoi diagram $\mathrm{Vor}(V_Q \cup S)$ that lies outside of Q. If that part contains Voronoi vertices then we define the *critical region*, $R(e)$, for e as the union of all the critical triangles that are dual to these vertices. Each critical triangle of $\mathrm{DT}(V_Q \cup S)$ belongs to a unique critical region.

Lemma 2 *No edge e of a non-critical triangle $\triangle$ of $DT(_Q \cup S)$ is longer than $(2 + \sqrt{2}\alpha)\underline{l}$. This upper bound is tight.*

Proof. Let r be the radius of circumcircle of $\triangle$. We assume that x_1 is nearest to the circumcenter x among four vertices x_i, for $i = 1, \ldots, 4$, of four vertices of a grid square containing x. Then we obtain $d(x_1, x) \leq \frac{1}{\sqrt{2}}\alpha\underline{l}$.

(1)When x_1 lies inside of a Voronoi cell of $\mathrm{Vor}(V_Q \cup S)$ whose generating point is one of three vertices of $\triangle$, let p denote the nearest point to x_1 of $\triangle$, we have $d(x_1, p) \leq \underline{l}$ from the terminal condition of algorithm. The triangle inequality implies

$$r \leq d(x, x_1) + d(x_1, p) \leq \left(1 + \frac{\alpha}{\sqrt{2}}\right)\underline{l}.$$

(2)When x_1 lies inside of a Voronoi cell whose generating point q is other than three vertices of $\triangle$, we get $r \leq \left(1 + \frac{\alpha}{\sqrt{2}}\right)\underline{l}$ in the same manner as above.

As all edges of $\triangle$ are less than $2r$, an edge of *non-critical triangle* is less than $(2 + \sqrt{2}\alpha)\underline{l}$. For $\alpha = \frac{1}{m\sqrt{2}}$, where m is an integer, we can realize the upper bound edge length. □

For a critical triangle, the lemma in [7] applies to our algorithm.

Lemma 3 *No edge f of a critical triangle in $R(e)$ is longer than e.*

Theorem 4 *Let $T^{'}$ denote the triangulation obtained by the algorithm. Then, the length of every edge of $T^{'}$ is between $\underline{l}$ and $(2 + \sqrt{2}\alpha)\underline{l}$.*

Next observation is for an angle of $\mathrm{DT}(V_Q \cup S)$. In the same way of analysis in [4, 8, 9], we can prove the following lemmas.

Lemma 5 *No angle of a non-critical triangle $\triangle$ is less than* $\arccos\left(\frac{1}{2+\sqrt{2}\alpha}\right)$.

Proof. Let r be the radius of circumcircle $\triangle$, and let e be an edge of $\triangle$, whose length is denoted by $d(e)$, and let θ be the angle opposite to e. It is a well-known geometric fact that $\theta = \arcsin\frac{d(e)}{2r}$. From Theorem 4, we have $d(e) \geq \underline{l}$. As remarked in Lemma 2, $r \leq \left(1 + \frac{\alpha}{\sqrt{2}}\right)\underline{l}$. Therefore, we obtain $\theta \geq \arcsin\left(\frac{1}{2+\sqrt{2}\alpha}\right)$. □

Lemma 6 *No angle of a critical triangle of $\triangle$ is less than* $\arccos\frac{\beta}{2}$.

Theorem 7 *If the vertex set of Q satisfies $\underline{l} \leq d(Q, Q) \leq \beta\underline{l}$, where $\beta \leq 2$, the angles obtained by the algorithm are at least* $\min\{\arccos\frac{\beta}{2}, \arcsin\left(\frac{1}{2+\sqrt{2}\alpha}\right)\}$.

Computational complexity for triangulation. First we count the number of Steiner points n. Let $r^* = \left(1 + \frac{\alpha}{\sqrt{2}}\right)\underline{l}$. The $n + |V_Q|$ circles with radius r centered at the points in $V_Q \cup S$ completely cover the polygon Q. Then we get the upper bound for n,

$$n < \frac{Area(Q)}{\pi\left\{\left(1 + \frac{\alpha}{\sqrt{2}}\right)\underline{l}\right\}^2} - |V_Q|.$$

Steps2∼4, of the algorithm has the same structure as incremental Voronoi partition. Therefore, a runtime for triangulation is $O((n + |V_Q|)^2 \log(n + |V_Q|))$, (see [2]).

4 Experimental Results

We have implemented the proposed algorithm. We shall show experimental results obtained by applying the algorithm for the convex polygon, given by $x^2 + (y/1.5)^2 = 400^2$ (see Fig.2(a)). We have tested the case of $\underline{l} = 40$, $\beta = 2.0$. The relationship between α and the error of boundary approximation is given in Table1. The table shows a clean trade-off between the number of edge lengths and the closeness of the approximation of the boundary. Results for $\alpha = 0.4$ and $\alpha = 0.2$ are shown in Fig.2(b) and (c). In Fig.2(c), it is hard to recognise the difference between P and Q.

5 Conclusion and future work

We have presented an algorithm for generating a triangular mesh with a constant number of different edge lengths. In previous works [7, 10], the triangular mesh, having $O(|V_P|)$ different edge lengths, is presented. In this paper, we obtain a triangular mesh such that the number of different edge lengths is $\frac{3\pi}{8\alpha^2} + o(\frac{1}{\alpha^2})$ with a boundary gap depending on α. This implies that we can

Table 1: Results for a convex polygon. Change with respect to α, for $\underline{l} = 40$.

α	♯ of lengths upper-bound	♯ of lengths	Max length upper-bound	Max length	Min length lower-bound	Min length	♯ of angles	Hausdorff distance
0.1	153	84	85.7	79.7	40.0	40.2	432	1.3
0.2	50	29	91.3	79.2	40.0	43.3	181	2.7
0.4	18	11	102.6	86.2	40.0	45.3	50	6.0
0.6	10	6	113.9	86.5	40.0	48.0	12	9.3

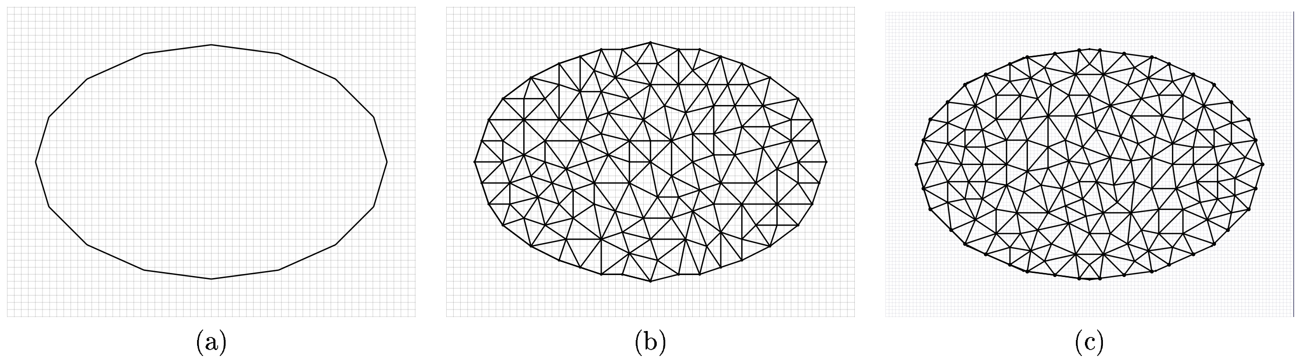

Figure 2: (a)Initial boundary. (b)Approximated boundary and its triangular mesh, where $\alpha = 0.4$. (c)Approximated boundary and its triangular mesh, where $\alpha = 0.2$.

construct a triangular mesh with a constant number of different edge lengths, depending on a permissible error of the boundary for a designer. This feature is very useful in the case of application to the architecture.

We have also implemented the polygon approximation algorithm which minimizes the area of the symmetric difference. Details will be reported at the conference.

We are interested in the following problems:

- Optimal grid layout
- Extension to curved surfaces
- Characterization of curved surfaces realized by a constant number of edge lengths

Triangular mesh on curved surfaces with $O(\frac{1}{\alpha^3})$ different edge lengths will be obtained by using cubic grid. However, it is difficult to find an approximation of the initial boundary.

References

[1] H. Alt and L. J.Guibas. Discrete geometric shapes: Matching, interplation, and approximation. In *Handbook of Computational Geometry*, pages 121–153. Elsevier Science Publishers, 1999.

[2] F. Aurenhammer, N. Katoh, H. Kojima, M. Ohsaki, and Y.-F. Xu. Approximating uniform triangular meshes in polygons. In *Theoretical Computer Science*, pages 879–895, 2002.

[3] M. Bern and D. Eppstein. Mesh generation and optimal triangulation. In *Computing in Euclidean Geometry, Edited by Ding-Zhu Du and Frank Hwang, World Scientific, Lecture Notes Series on Computing – Vol. 1.* World Scientific.

[4] L. P. Chew. Guaranteed-quality mesh generation for curved surfaces. In *Annual ACM Symposium on Computational Geometry*, pages 274–280, 1993.

[5] L. P. Chew, M. T. Goodrich, D. P. Huttenlocher, K. Kedem, J. M. Kleinberg, and D. Kravets. Geometric pattern matching under Euclidean motion. *Computational Geometry: Theory and Applications*, 7:113–124, 1997.

[6] N. Katoh, H. Kojima, and R. Taniguchi. Approximating uniform triangular meshes on spheres. In *Japanese Conference on Discrete and Computational Geometry*, pages 192–204. LNCS, 2000.

[7] N. Katoh, M. Ohsaki, and Y. Xu. A uniform triangle mesh generation of curved surfaces. In *Proc. of JCDCG 2004*. Springer-Verlag, 2004.

[8] J. Ruppert. A Delaunay refinement algorithm for quality 2-dimensional mesh generation. *Journal of Algorithms*, 18:548–585, 1995.

[9] J. R. Shewchuk. Delaunay refinement algorithms for triangular mesh generation. *Computational Geometry: Theory and Applications*, 22:21–74, 2002.

[10] Y. Xu, W. Dai, N. Katoh, and M. Ohsaki. Triangulating a convex polygon with less number of non-standard bars (extended abstract). to appear in *Proc. of COCOON 2005. The Eleventh International Computing and Combinatorics Conference*, 2005.