Modifying Delaunay Refined Two-Dimensional Triangular Meshes

Narcís Coll, Marité Guerrieri and J.Antoni Sellarès *

Abstract

We propose algorithms to modify a mesh of a PSLG through out the interactive addition/deletion of elements to/from the PSLG, keeping the quality of the mesh all along the process. Our algorithms achieve quality by deleting, moving or inserting Steiner points.

1 Introduction

There exist many works on the generation of quality meshes. *Delaunay refinement mesh generation* algorithms have taken place in this frame of investigations [6, 7, 5, 4]. Different kind of domains can be meshed, but a Planar Straight Line Graph (PSLG) is the main input to 2D refining algorithms. In all these works modification of the initial PSLG implies a regeneration of the whole mesh. These mesh generation algorithms do not allow us to modify the initial PSLG incrementally.

In keeping quality of a mesh two objectives are pursued. First, get *skinny* triangles, triangles without required quality, out of the mesh. Second, force segments of the PSLG into the mesh. Both goals are achieved by the addition of Steiner points, points that do not belong to the original mesh. In current Delaunay refinement algorithms, two kinds of Steiner points deal with the former goal, namely, circumcenters and off-centers. The later objective is carried out by the addition of midpoints on constrained segments to insert.

All these algorithms guarantee that the length of the edges of the triangulation are greater than the *minimum local feature size* (lfs_{min}) of the PSLG. This lfs_{min} is the shortest distance between two nonincident elements (points or segments) of the input PSLG.

Applications where a combination of dynamic modifications of a mesh and numerical methods like the Finite Element Method (FEM) is required have motivated this research. Apart from the quality requirement, these applications expected additional features to be provided by the algorithms, namely:

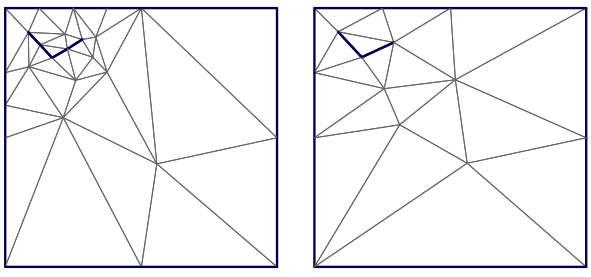
Progressively: The interactive modification of the initial PSLG are obtained without the regeneration of the whole mesh.

Locality: The changes applied to the mesh do not imply a propagation of these modifications to the whole mesh.

Optimality: The Steiner points added as a result of the modification of the mesh should be as few as possible.

Modification of quality meshes under insertion or deletion of PSLG elements required incremental algorithms. In order to obtain a new refined mesh adding the minimum number of Steiner points we have to consider that the current mesh is treated as an input PSLG by existing algorithms. Consequently, the lower bound of lengths of the edges of the new mesh will not be the lfs_{min} of the modified PSLG. This bound will be the lfs_{min} of the union of the current mesh and the modified PSLG.

Possible solutions to the excessive insertion of points could be the movement or deletion of Steiner points, or a different algorithm that split constrained segments. The movement of points, also, could resolve the problem of the generation of small triangles produced by a degradation in the distribution of points in successive updates of the mesh. Movement of points could assure better lfs_{min} bounds. Figure 1 shows the insertion of four consecutive segments before and after deleting Steiner points. In Figure 1(b) deleted Steiner points are shown to facilitate the understanding of the process.



(a) (b)

Figure 1: Insertion of four consecutive segments. Triangulation before and after deletion of Steiner points.

2 Quality zones

The main idea behind our proposed algorithms is that the quality of a mesh can be improved by means of the movement of Steiner points belonging to skinny triangles. This solution relies on the fact that it is possible to define a zone for a given Steiner vertex

*Departament d'Informàtica i Matemàtica Aplicada, Universitat de Girona, {coll,mariteg,sellares}@ima.udg.es. Work partially supported by grant TIN 2004-08065-C02-02.

where it can be placed ensuring that the quality is preserved.

The *star*, $\mathcal{S}_q$, of a vertex q of a mesh consists of all the triangles that contain q. All edges of triangles in $\mathcal{S}_q$ that are disjoint from q form the *link*, $\mathcal{L}_q$, of q. The *quality zone* $\mathcal{Z}_{ab,t,\alpha}$ of an edge ab of a triangle t for an angle α, that controls the quality of the triangles, is defined by the domain of points p external to the circumcircle of the triangle adjacent to t by ab and satisfying $\widehat{apb} \geq \alpha$, $\widehat{pab} \geq \alpha$ and $\widehat{abp} \geq \alpha$. These conditions assure that the triangle abp is Delaunay and non-skinny. See Figure 2 for an example of this defined zone. The *Quality zone* for a given vertex q is defined by $\mathcal{Q}_{q,\alpha} = \bigcap \mathcal{Z}_{e,t_e,\alpha}$, $\forall e \in \mathcal{L}_q$ and $t_e \in \mathcal{S}_q$. The computation of $\mathcal{Q}_{q,\alpha}$ can be achieved by means of a sweep algorithm. Since the mean number triangles of $\mathcal{S}_q$ is bounded by six, then it can be inferred that the mean cost of the computation of the quality zone of q is constant.

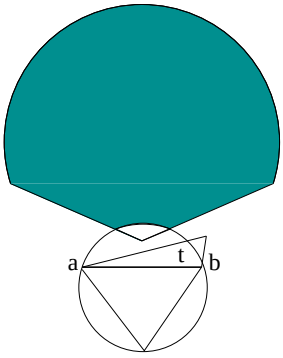


Figure 2: Quality zone of edge ab

3 Basic operations

Our main goal is the design of an algorithm that maintains a Delaunay refined mesh after the progressive insertion of elements from a PSLG. We also work under the demand of achieving local modifications of the mesh and the challenge of adding as less Steiner points as possible.

Movement and deletion of Steiner points are the *basic operations* designed to carry out these objectives. To face the appearance of skinny triangles, caused by the insertion of a new element from a PSLG, the algorithms developed try first to destroy each skinny triangle by moving or deleting its Steiner vertices, adding additional Steiner points, circumcenter or midpoints, if basic operations can not be applied. The algorithm herein described is based in a modification of an incremental Delaunay one. As it will be seen all these operations have a constant cost. The use of an incremental algorithm as well as movement and deletion of points allow us to modify a mesh in a local and progressive way, adding a lesser number of Steiner points than whether existing Delaunay refinement algorithms are used.

It has to be taken into account that points belonging to the previous mesh as well as Steiner points inserted by the modification in process are target points to be treated by our algorithm. Previous circumcenters or midpoints could be moved as a result of applying a basic operation, therefore, it is necessary to give them a new denomination. Points to be treated by our algorithm can basically belong to two main groups. One group is formed by Steiner points that are on a segment of the PSLG, whose movement is restricted to the segment. A second group is formed by those points to which their restriction of movement comes from the triangulation itself. We have named the first group *restricted vertices*, and the second group *free vertices*.

3.1 Moving free vertices

The key concept regarding the movement of a free vertex p is to substitute this vertex for another point q interior to the quality zone of p. If $\mathcal{Q}_{p,\alpha}$ is empty the vertex p can not be moved. The problem of the election of a point q that maximizes the minimum angle of $\mathcal{S}_q$ has been studied by [1] and others, but they do not include the Delaunay property as a restriction as it is required in our problem. During the sweep process used to compute the quality zone, we currently choose the midpoint of the first segment contained in the quality zone. Choosing the point into the quality zone that maximizes the smallest angle is left as a future work.

The quality zone ensures that if the vertex p is placed inside it the new $\mathcal{S}_p$ is constituted by *non-skinny* triangles. The quality zone, also, produces a Delaunay triangulation of the $\mathcal{L}_p$ with respect to its exterior triangulation. But this zone does not guarantee that triangles belonging to this $\mathcal{S}_p$ fulfil Delaunay property among them. For this reason the vertex p has to be deleted and reinserted in the quality zone using an Incremental Delaunay algorithm.

Consequently, the steps to determine whether a free vertex p can be moved are: Determine $\mathcal{Q}_{p,\alpha}$, determine point q in $\mathcal{Q}_{p,\alpha}$, delete p, and finally, insert q.

3.2 Deleting free vertices

Devillers in [2] proposed an algorithm to delete a point from a Delaunay triangulation. Basically, his algorithm retriangulates $\mathcal{L}_p$ by determining *Delaunay ears* using the concept of *power* of p. We need to obtain not just a Delaunay triangulation, but a Delaunay refined one. Consequently we verify whether the Delaunay ear is skinny or not, stopping the process when a skinny one was found. The algorithm may then be stated as:

1. Obtain a priority queue in ascending order of the power of ears from the $\mathcal{L}_p$
2. Repeat until three ears remain in the queue or a skinny triangle is found
   (a) Take the first ear from the queue and flip

the diagonal to form a triangle
(b) If the triangle is not skinny
(i) Modify ears previous to and next to the treated ear
(ii) Compute the power for these two new ears
(iii) Update the priority queue

3.3 Moving restricted vertices

The movement of those restricted vertices is constrained over its correspondent subsegment. This kind of vertices can be present on a boundary subsegment or on a non boundary subsegment of a PSLG. In both cases the same algorithm to determine the movement of these vertices is applied.

The steps to determine when a restricted vertex p can be moved are the followings:

1. Determine adjacent triangles to p
2. For each edge e opposite to p
   (a) Determine quality zone, z
   (b) Calculate segment $i_e = z \cap s$, where s is the subsegment that contains vertex p
3. Determine $i = \cap i_e$
4. If i is not empty
   (a) Determine a point q on i
   (b) Retriangulate: change the vertex p by q

3.4 Deleting restricted vertices

Deletion of a restricted vertex depends on whether it belongs or not to a boundary subsegment of the PSLG. The presence of a restricted vertex on a non-boundary subsegment implies the application of the deletion algorithm explained in the section 3.2 to the two sides of the subsegment independently. In this way the point is deleted only if each side independently fulfils the point deletion verification, and then the region in each side is retriangulated individually. The same steps from the algorithm of section 3.2 can be carried out without changes, with the only extra consideration that a restricted vertex which encroached some of the polygon points can not be deleted. In case of a boundary subsegment the process detailed above is applied only to the interior side.

3.5 Expanding deletion of vertices

Once a vertex has been deleted from the mesh other vertices are susceptible of deletion. Each time a vertex is deleted all its adjacent Steiner vertices are added to a queue to be deleted, producing in this way several iterations. The iteration process ends when any of the vertices in the queue can not be deleted.

4 Modifying a 2D Delaunay refined mesh

Modification of a Delaunay refined mesh will mean to insert/delete elements to/from a PSLG. Using the basic operations we design algorithms for insertion and deletion of points, segments, polygonal lines, polygonal holes, as well as progressive polygonal lines insertions.

After the insertion or deletion of an element of the PSLG, a process of refinement is called that carries the quality through the mesh.

4.1 Inserting an element

To insert a point p of the PSLG, we have modified the incremental Delaunay algorithm from [3]. A list of midpoints belonging to segments that could not be flipped, and a list of the generated skinny triangles is passed to the refinement process.

We used a recursive algorithm to insert a segment of the PSLG into the mesh. The process starts inserting its endpoints, then if the segment does not appear as an edge in the mesh a midpoint is added and a Delaunay process is triggered. The process of adding midpoints continues over half segments that are not yet into the triangulation.

Inserting a polygonal line into the mesh will mean to insert several segments of the PSLG one after the other. Inserting a polygon means to insert a closed polygonal line.

To obtain a polygonal hole inserted into the mesh we start inserting the polygon without any refinement, and then we delete the internal triangles.

4.1.1 Inserting a polygonal line progressively

To insert a polygonal line progressively we add interactively a set of points $p_0, \cdots, p_n$. Next we describe the basic process we follow. For each point p_i we insert the segment $p_{i-1}p_i$. If point p_i is collinear with p_{i-2} and p_{i-1} then p_{i-1} is considered a Steiner point and a deletion point process is started with this point.

4.2 Deleting an element

To delete an element of the PSLG (point, segment, polygonal line ...) the free and restricted vertices of the element are considered as Steiner points and then a deletion process is run for them.

4.3 Refinement process

It is a process applied after the insertion or deletion of an element of the PSLG. It receives as input two lists. The list of points to insert, initially containing only midpoints, and the list of skinny triangles to remove, produced by insertion of an element. The output of the process is a mesh with desired quality. The process maintains the two lists and finishes when the two list are empty. Priority is established on midpoints. To remove an skinny triangle, we first check its Steiner vertices for deletion, then we check its Steiner vertices for movement, and finally we add circumcenters to the list of points. This order of treatment of skinny triangles is important in order to obtain a reduction in the number of vertices. To insert a midpoint we apply our

algorithm to insert a point into the mesh. To insert a circumcenter we follow rules from Ruppert's algorithm: circumcenter is not inserted if it is encroached over a constrained edge, in this case the midpoint of this edge is added to the list of points.

5 Results

In Figure 3 we show the insertion of a hole into the PSLG. Figure 3(a) is obtained by applying the incremental algorithm with movement and deletion of Steiner vertices and Figure 3(b) without movement or deletion of Steiner vertices. In the first case less triangles have been generated.

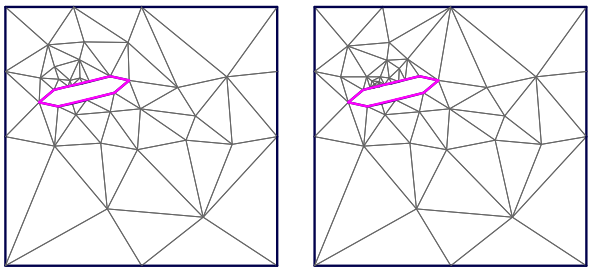


Figure 3: Insertion of a hole applying incremental algorithm.

Figure 4 shows a sequence of six steps during the insertion of a polygonal line using our algorithm. From Figure 4(a) to 4(c) a deletion operation of a vertex has been carried out. From Figure 4(d) to 4(e) a vertex has been moved. In these triangulations the Steiner points deleted are shown to facilitate the understanding of the process.

In Figure 5 the progressive insertion of the polygonal presented in the previous example is achieved using the same algorithm but without moving or deleting Steiner vertices. It can be seen that in this case we obtain more triangles.

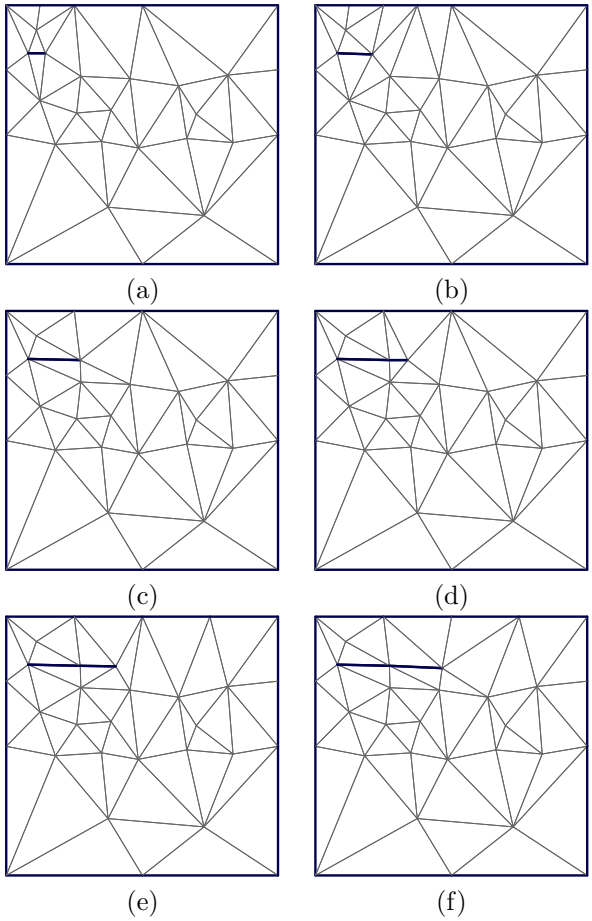


Figure 4: A sequence showing a progressive polygonal line insertion

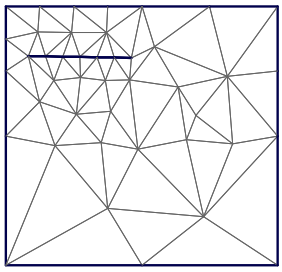


Figure 5: A progressive polygonal line insertion without moving or deleting Steiner vertices.

References

[1] N. Amenta, M. Bern and D. Epstein. Optimal point placement for mesh smoothing. *In SODA:ACM-SIAM Symposium on Discrete ALgorithms*, 1997.

[2] O. Devillers. On deletion in Delaunay triangulation. *15th Annual ACM Symposium on Computational Geometry*, 181–188, 1999.

[3] L. Guibas, D. Knuth and M. Shair. Randomized incremental construction of Delaunay and Voronoi diagrams. *Algorithmica*, 7:381–413, 1992.

[4] S. Har-Peled and A. Üngör. A Time-Optimal Delaunay Refinement Algorithm in Two Dimensions. *21st Annual ACM Symposium on Computational Geometry (SoCG)*, 228–229, 2005.

[5] S.-E. Pav. Delaunay Refinement Algorithms. *Department of Mathematical Sciences - Carnegie Mellon University - PhD thesis*, 2003.

[6] J. Ruppert. A Delaunay Refinement Algorithm for Quality 2-Dimensional Mesh Generation. *Journal of Algorithms*, 18:3:548–585, 1995.

[7] J.-R. Shewchuk. Delaunay Refinement Mesh Generation. *School of Computer Science - Carnegie Mellon University - PhD thesis*, 1997.