On The Robust Construction of Constrained Delaunay Tetrahedralizations

Feng Wang[1] and Luca Di Mare[2]

[1] Department of Mechanical Engineering, Imperial College London, UK
`feng.wang207@imperial.ac.uk`
[2] Department of Mechanical Engineering, Imperial College London, UK
`l.di.mare@imperial.ac.uk`

Summary. We present a simple and robust algorithm to construct the Constrained Delaunay Tetrahedralization(CDT) for a Piecewise Linear Complex(PLC). The algorithm is based on Si's CDT [1] algorithm, but we provide a new facet recovery algorithm, which is simple and easy to implement. The implementation and robustness issues of the CDT algorithm are discussed in detail and the program which implements the algorithm is tested with several examples. The results are compared with TetGen [2] and demonstrate the robustness and efficiency of the algorithm.

1 Introduction

One particular problem of Delaunay triangulation is preserving the integrity of boundaries. In two dimensions, recovering the boundary constraints of a Delaunay triangulation is successfully solved with no Steiner points and optimal complexity [3]. In three dimensions, the problem is far from solved. This research note will focus on one of the boundary recovery techniques for Delaunay Tetrahedralization, the Constrained Delaunay Tetrahedralization, which is firstly proposed by Shewchuk [4] and then further developed by Si [1].

In this research note, we discuss the robust implementation of a simple and efficient CDT algorithm. The algorithm is closely related to Si's algorithm [1] but we provide a simple and efficient facet recovery algorithm.

The reminder of this research note is organized as follows. The CDT algorithm is firstly overviewed in the next section. Then the implementation issues of the algorithm is discussed in detail in Section 3, Section 4 and Section 5. Finally we present the meshing results for several examples.

2 Overview of The CDT algorithm

The input is a 3D Planar Linear Complex(PLC) X [4]. X consists of a set of vertices, segments and facets. Each facet of X is represented as a union of triangular faces or so-called subfaces and each side of a subface here is considered as a segment. According to Si's work [1], *if D is the DT of X, the vertices of D are in general position, and D contains all the segments of X, then X has a unique CDT with no Steiner points.* The condition that vertices are in general position means that there are no five or more vertices on the same sphere. This condition can be achieved by using symbolic perturbation [4].

If the DT of the input PLC X_0 is DT_0, DT_0 might not respect segments and facets of X_0. Missing segments and facets will be recovered in the following three steps:

1. Insert Steiner points into DT_0 and transform DT_0 into DT_1, so that all the segments of X_0 appear as themselves or as a union of edges in DT_1. This step is called *segment recovery.*
2. Insert the Steiner points into X_0 and transform X_0 into X_1. As DT_1 is the DT of X_1 and all the segments of X_1 appear in DT_1, according to Si [1], the CDT of X_1 exists without no Steiner points.
3. Group missing subfaces into missing regions and recover them incrementally. This step needs no Steiner points and is called *facet recovery.*

3 Segment recovery

The general steps of the segment recovery algorithm is similar to the method recently proposed by Si [1]. Its robustness and implementation issues will be discussed in detail below.

Due to round-off errors, a Steiner point V_n might not lie exactly on the segment V_iV_j it tries to split.The history tag method [5] would fail to locate V_n, if such perturbation is ignored, because the success of this method relies on the principle that if a point lies inside a parent tetrahedron, then the point must be contained by one of its child tetrahedron. However, the jump and walk method [6] does not suffer from this problem, because its success relies on the integrity of DT_0. On another hand, since the number of tetrahedra stored by the history tag method is normally 8-10 times larger than the jump-and-walk method, this drawback becomes a prohibitive memory cost when a large surface grid is used. Hence the history tag method is not recommended.

If the missing segment V_iV_j is on the convex hull of DT_0, V_n might be perturbed outside of the convex hull. Under such conditions, both methods will fail. A simple solution is to use eight shadow points to form a large cube, which is big enough to contain the whole PLC.

The efficiency of the segment recovery algorithm are found to be closely related to the following aspects:

Finding a reference point

A reference point can be found by walking through all the tetrahedra which intersect the missing segment, then the reference point is randomly selected from the point set of these tetrahedra [1].

Locating a Steiner point

To locate a Steiner point, the jump-and-walk algorithm can start from a tetrahedron which contains a vertex of the missing segment.

Identifying a segment

Identifying a segment is used to keep track of missing segments when Steiner points are inserted into DT_0. If a Steiner point remembers the segment it has split and each segment remembers its parent segment, and the adjacency graph of X_0 entities is maintained, identifying a segment is trivial to do. If one of the end point of an edge V_iV_j is a Steiner point, we can check all the segments that contains the Steiner point to see if V_iV_j is a segment, otherwise, we can easily check it using the adjacency graph of X_0.

4 Updating PLC

After segment recovery, the point set of X_0 is enriched by Steiner points, and X_0 is transformed into X_0'. In X_0', segments of X_0 appear as themselves or as a union of their child segments in DT_1. New subfaces need to be created in X_0' so as to transform X_0' into X_1, so that DT_1 is a DT of X_1. Based on Si's condition [1], facet recovery of X_1 can be undertaken with no Steiner points.

In X_0, as segments are sides of subfaces in X_0, a union of new subfaces in X_0' are spawned by incrementally inserting Steiner points into each subface σ_0 of X_0. As points are always inserted on sides of subfaces, point location can be done efficiently by letting a Steiner point remember on which side of σ_0 it is inserted on, so only a simple point-on-segment test is needed. This procedure is found to be very effective to avoid a possible robustness issue. As a Steiner point might be perturbed off the segment it tries to split, a naive point-on-segment might fail to tell on which side of σ_0 a Steiner point is located, especially if σ_0 is long and thin.

5 Facet recovery

Subfaces in some facets of X_1 may be missing in DT_1. All missing subfaces are recovered in this step. A facet is said to be recovered if all its subfaces appear in X_1.

Missing subfaces on the same facet can be grouped into missing regions if they are generated from the same subface in X_0 and at least one of their neighbors are recovered or generated from a different subface in X_0. A subface in X_0 can have several missing regions. Our definition of missing regions is similar to that of Shewchuk and Si [4, 1]. The primary difference is that only subfaces that are generated from the same subface, instead of the same facet, in X_0 can form a missing region Ω. Due to this difference, the properties of our missing regions are also slightly different:

- All subfaces of Ω belong to the same subface in X_0
- At lease one of the neighbors of each subface of Ω are either recovered or generated from a different subface in X_0
- The edges on $\partial\Omega$ are edges of DT_1
- All the internal edges of Ω are missing
- Ω is simply connected.

All subfaces of Ω belong to the same subface in X_0, as segment recovery only insert Steiner points on sides of subfaces in X_0, Ω must be simply connected[3]. Due to the above properties, identifying missing regions can be done robustly and the sizes of resulting missing regions are considerably smaller than that of Shewchuk and Si [4, 1].

A missing region Ω can be categorized into two types: Ω is said to be M-1 if its interior is not intersected by any edge in DT_1, otherwise it is M-2. According to their types, missing regions can be recovered in different ways.

If Ω is M-1, Ω actually represents a union of faces in DT_1 and then is re-meshed to match faces in DT_1. If Ω is M-2, its interior is intersected by at least one edge in DT_1. To recover Ω, tetrahedra in DT_1 which intersect Ω are removed first. Two cavities, C_1 and C_2, are then derived from removed tetrahedra and re-meshed separately to force subfaces in Ω to appear. A modified advancing front method is used to mesh the cavities and it resembles the gift-wrapping algorithm [4]. Since the size of Ω are small and it is simply connected, derived cavities can be meshed easily.

It is worth mentioning that as the cavity verification [1] is ignored, recovered subfaces in Ω might not be constrained Delaunay, and a complete proof of the termination of recovering M-2 missing region is our future work.

6 Experimental Results

The algorithm has been implemented in our program *Mauggrt*. Its performance will be demonstrated by two examples, Statue[4] (25386 vertices and 50780 faces) and CORE[5] (920270 vertices and 1840540 faces). Their CDTs are shown in Figure 1.

Table 1 shows the runtime statistics of Mauggrt and its comparison with Tetgen [6]. The comparison shows that Mauggrt presents comparable efficiency to Tetgen but require slightly more Steiner points. Table 2 shows statistics of missing regions and cavities in facet recovery. We can see that the sizes of missing regions and cavities are all relatively small values.

[3]Otherwise, it means segment recovery inserts Steiner points inside subfaces in X_0 or subfaces in X_0 have holes, which obvious will not happen.

[4]Statue is available from http://www-roc.inria.fr/gamma/download/

[5]CORE is a piece of structure from a three-shaft aero-engine.

[6]Tetgen version is 1.4.3. Both program are compiled in a Linux workstation with the default option of Tetgen, that is -O0 for geometry predicates and -g -Wall for other .cxx files.

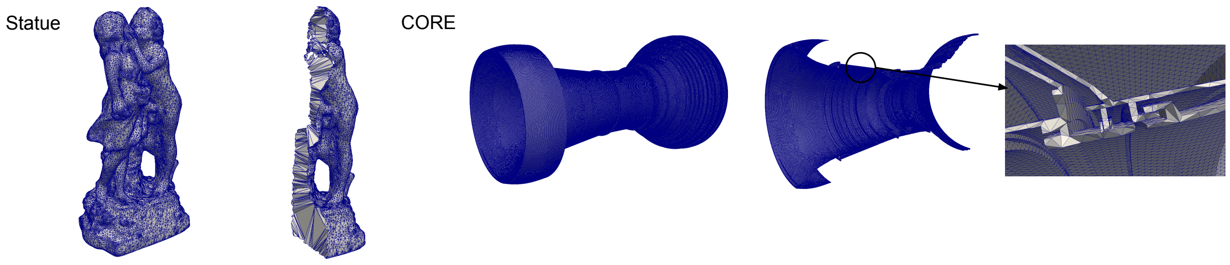

Fig. 1. CDT of Statue and Core

Table 1. Runtime statistics of Mauggrt and comparison with TetGen.

	No. of Steiner point		Segment recovery(s)		Update PLC(s)		Facet recovery(s)		Total(s)	
	Mauggrt	Tetgen	Mauggrt	Tetgen	Mauggrt	Tetgen	Mauggrt	Tetgen	Mauggrt	Tetgen
Statue	4951	3720	0.76	0.86	0.09	N/A	0.55	0.24	1.40	1.10
CORE	385024	3000534	48.85	44.92	4.32	N/A	44.96	38.33	98.13	83.25

Table 2. Statistics of missing regions and cavities in facet recovery

	No.of missing faces	No. of missing regions	No. of M-1	Max. size of M-1	No. of M-2	Max. size of M-2	Max. size of cavities
Statue	99	98	1	2	97	1	5
Core	145704	74396	608	4	73788	13	40

Acknowledgements

The authors gratefully acknowledge Rolls-Royce plc. for sponsoring this work and granting permission for publication.

References

1. Hang Si, K. Gartner. 3D Boundary Recovery by Constrained Delaunay Tetrahedralization(2011). Int.J.Nume.Meth.Eng. 85,1341-1364.
2. http://tetgen.berlios.de/
3. http://www.cs.cmu.edu/ quake/triangle.html
4. Jonathan Richard Shewchuk. Constrained Delaunay Tetrahedralizations and Provably Good Boundary Recovery(2002). 11th IMR, pp 193-204.
5. Leonidas J. Guibas, Donald E. Knuth, and M. Sharir. Randomized Incremental Construction of Delaunay and Vorono Diagrams (1992). Algorithmica 7(4):381-413.
6. L. Devroye, E. Mucke and B. Zhu. A note on point location in Delaunay triangulations of random points(1998). Algorithmica, 22:477-482.