Triangular Meshes Delaunay Conforming Filter

Release 0.01

Arnaud Gelas, Alexandre Gouaillard
and Sean Megason

August 5, 2008

Department of System Biology, Harvard Medical School,
Boston, MA 02115, USA

Abstract

The *Delaunay triangulation* is the triangulation of a set of points which maximizes the minimum angle of all angles of the triangles, and thus triangle aspect ratios. So converting a non Delaunay triangulation into a Delaunay triangulation, *Delaunay conforming*, improves the all triangle aspect ratios and avoid elongated triangles. Note that combining this filter with usual operations like smoothing will provide a better approximation and a better distribution a triangle aspect ratios. Here this document describes a new filter based on the *n*D 2-manifold mesh data structure available in itk: `itk::QuadEdgeMesh` [3] to produce (planar or surfacic) Delaunay triangulation from any non-Delaunay triangulation by edge flipping, by implementing the edge flipping method decribed in [1].

Contents

1 Introduction

In the plane Delaunay triangulation of a set of points $\mathcal{P}$ is the triangulation such that no point p of $\mathcal{P}$ is inside the circumcircle of any triangle of the triangulation. Such triangulations maximize the minimum angle of all angles of the triangles, and intend to avoid elongated triangles (see Figure 1).

Delaunay triangulations have been studied extensively in computational geometry first in 2D, generalized in higher dimensions (in 3D one Delaunay triangulation is a volumetric mesh where simplices are vertices, edges, triangles and tetraedrons), and for surface triangulation. Delaunay triangulations are nowadays used for many applications (simulation, finite-element analysis, visualization).

The filter presented here does not generate a Delaunay triangulation from a given set of points. Instead it creates a (planar or surfacic) Delaunay triangulation from a given nD 2-manifold non-Delaunay triangulation by edge flipping without modifying the point locations, *i.e.* only by changing the mesh connectivity, and under constraint of non exact geometry kernel.

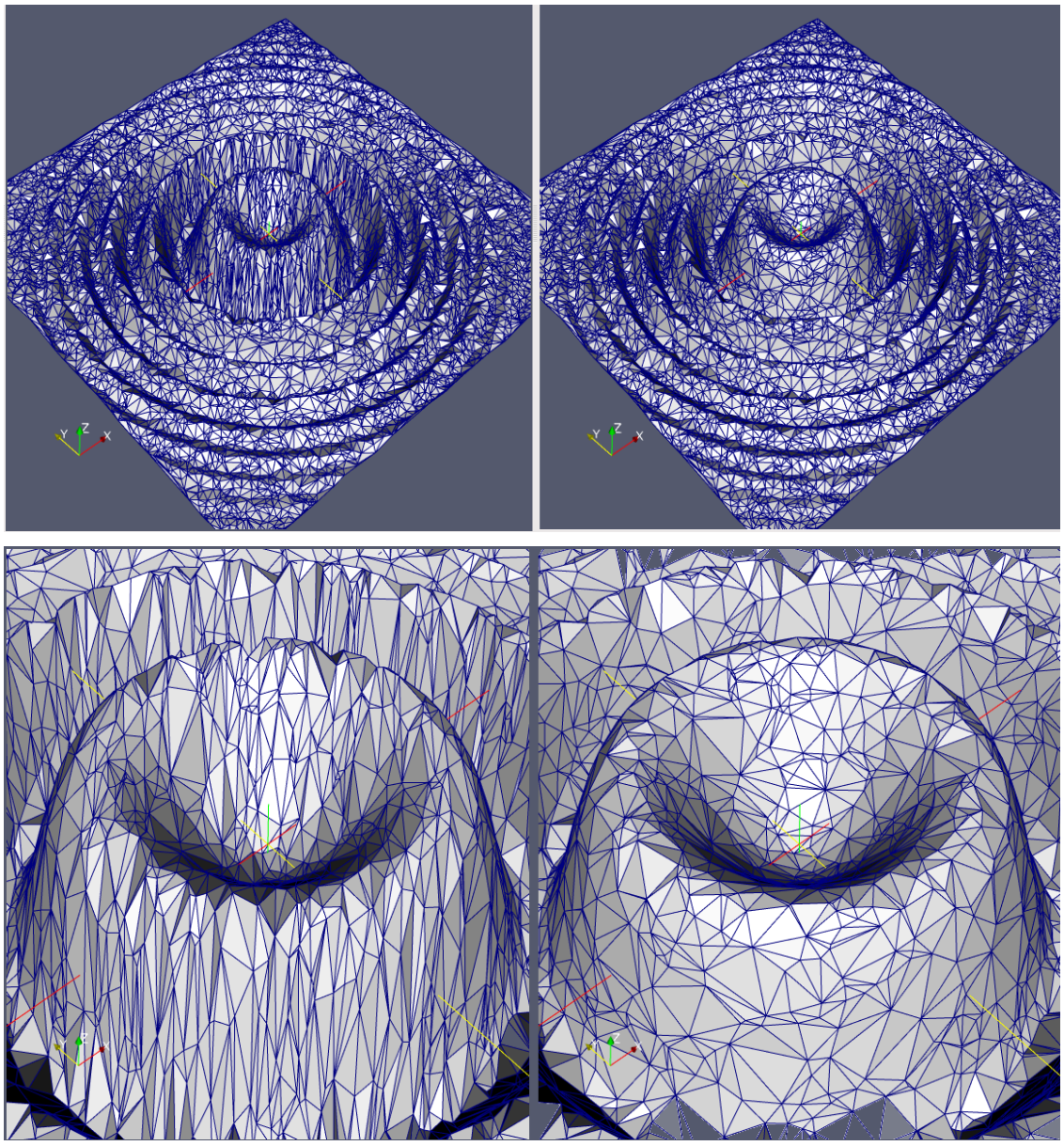

Figure 1: On the left a given non-Delaunay mesh $\mathcal{M}$, and on the right the resulting mesh after applying our filter.

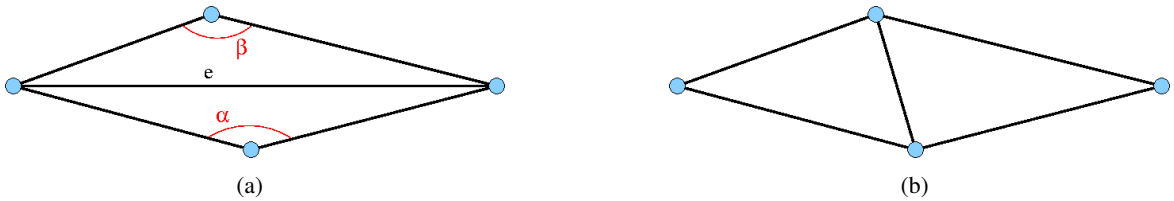

Figure 2: For a flippable edge e we compute its priority $p(e)$ as the sum of opposite angles at e minus π, *i.e.* $p = \alpha + \beta - \pi$ (a). If its priority is positive e must be flipped (b). The priority of the five edges displaid is then updated.

2 Principle

Following [1], we recall an *unflippable edge* an edge such that its opposing edge also exists in the mesh, a *constrained edge* is an edge which has been tagged by the user and which must not be modified[1], and thus a *flippable edge* an edge which is not unflippable and not constrained.

Given a triangular mesh $\mathcal{M}$, all flippable edges are pushed into a priority queue. The priority $p(e)$ of a given edge e is given by the sum of opposite angle (see Figure 2):

$$p(e) = \alpha + \beta - \pi \tag{1}$$

Then greedily the flippable edge e with the largest priority is flipped, and the priority of edges next to e is then updated. This process is reapeated until there is no more flippable edge with a positive priority in the priority queue.

3 Implementation

Based on the nD 2-manifold mesh data structure (`itk::QuadEdgeMesh` [3]), this filter is quite straightforward to implement. It only requires implementation of a method to compute the priority for a given edge (see Eq. 1), and a mutable priority queue container which allows to modify priority of any elements already inside the queue [2].

4 Usage

The use is really simple. The user only needs to provide a triangular mesh, and a list of constrained edges.

```
typedef double Coord;
const unsigned int Dimension = 3;
typedef itk::QuadEdgeMesh< Coord, Dimension > MeshType;

// Here we assume that mesh is given
...
```

[1] Constrained edge could be a boundary edge or a feature edge that the user wants to maintain.

```
typedef itk::DelaunayConform< MeshType, MeshType > DelaunayConformFilterType;
DelaunayConformFilterType::OutputEdgeCellListType list_of_constraints;

// push here all constraints
...

DelaunayConformFilterType::Pointer filter = DelaunayConformFilterType::New( );
filter->SetInput( mesh );
// if there is any constrained edges
filter->SetListOfConstrainedEdges( list_of_constraints );
filter->Update( );
```

5 Software Requirements

You need to have the following software installed:

- Insight Toolkit $\geq$ 3.7.0 (Revision $\geq$ 1.2737) compiled with USE_REVIEW ON.
- CMake $\geq$ 2.4

Acknowledgement

This work was funded by a grant from the NHGRI(P50HG004071-02) to found the Center for in toto genomic analysis of vertebrate development.

References

[1] Ramsay Dyer, Hao Zhang, and Torsten Möller. Delaunay mesh construction. In *SGP '07: Proceedings of the fifth Eurographics symposium on Geometry processing*, pages 273–282, Aire-la-Ville, Switzerland, Switzerland, 2007. Eurographics Association. (document), 2

[2] Arnaud Gelas, Alexandre Gouaillard, and Sean Megason. Mutable priority queue container. *Insight Journal*, January-June 2008. http://hdl.handle.net/1926/1395. 3

[3] Alexandre Gouaillard, Leonardo Florez-Valencia, and Eric Boix. itkQuadEdgeMesh: A discrete orientable 2-manifold data structure for image processing. *Insight Journal*, July-December 2006. http://hdl.handle.net/1926/306. (document), 3