Incremental Delaunay Triangulation

Release 0.01

Stéphane Rigaud[1] and Alexandre Gouaillard[2]

July 26, 2012

[1]Image & Pervasive Access Lab, National Centre for Scientific Research (CNRS), Fusionopolis, Singapore
[2]Singapore Immunology Network, Agency for Science, Technology and Research (A*STAR), Biopolis, Singapore

Abstract

This document describes the implementation in ITK of the *Incremental Delaunay Triangulation* algorithm [1]. Using the *Straight Walk in Triangulation function* [4], the *exact discrete geometrical orientation predicate* [3], and the *itk::QuadEdgeMesh* API [2] of ITK , we propose a geometrically exact and robust implementation that, from a given 2-dimensional *itk::PointSet*, incrementally constructs the corresponding 2-dimensional Delaunay Triangulation as an *itk::QuadEdgeMesh*.

Latest version available at the Insight Journal [http://hdl.handle.net/10380/3372]
Distributed under Creative Commons Attribution License

Contents

1 Principle of Incremental Delaunay Triangulation

Taking a planar set of points $\mathcal{P}$ of n points embedded into a n-dimensions space, we construct a triangulation $DT(\mathcal{P})$, that respects the Delaunay criterion stating that no point of $\mathcal{P}$ should be inside of the circumference circle of any triangle of $DT(\mathcal{P})$.

Several algorithms exist to compute a Delaunay triangulation and, arguably, the most straightforward way of computing it is the Incremental Delaunay triangulation algorithm [1]. Let $\mathcal{P}$ a point set and $DT(\mathcal{P}_t)$ the Delaunay triangulation of $\mathcal{P}_t \subset \mathcal{P}$. We construct $DT(\mathcal{P}_{t+1})$ by adding a point p_t randomly taken from $\mathcal{P} \backslash \mathcal{P}_t$ into the $DT(\mathcal{P}_t)$. Then, the triangle t of $DT(\mathcal{P}_t)$ that embed the point p_t is located and subdivided into three new triangle t_1, t_2 and t_3, which share the same vertex p_t.

2 Implementation

2.1 Input & Output

This algorithm will generate a 2-manifold planar mesh of 1 component and 1 boundary, embedded into a n-dimensional space, but that will be parallel to the plan $(0,x,y)$. This output mesh will respect the Delaunay criterion. The filter is templated over 2-dimensions *itk::PointSet* and *itk::QuadEdgeMesh*, if a higher dimensional mesh or set of points is given, only the two first dimensions will be used in the process.

2.2 Inheritance

The algorithm is implemented as a filter class that takes an *itk::PointSet*, or any type that inherits from it, as input and generates an *itk::QuadEdgeMesh* as output. The current ITK classes do not allow PointSet to Mesh process (Fig1a). The closest existing class is *itk::MeshToMeshFilter* which manage the copy of the points, point data, cell, cell links and cell data that define an *itk::Mesh* object. But our algorithm only use the points and point data information to generate a triangulation. Therefore, in order to respect the ITK template implementation, we have extracted the points and point data copy process that was in *itk::MeshToMeshFilter* and added in intermediate new class *itk::PointSetToMeshFilter* and made *itk::MeshToMeshFilter* inherits from this new class (Fig. 1b). The points and point data copy from input to output that was implemented in *itk::MeshToMeshFilter* is now done in *itk::PointSetToMeshFilter*, while the cell, cell links and cell data copy from input to output is left to *itk::MeshToMeshFilter* to handle. Our filter, *itk::PointSetToDelaunayTriangulationFilter* will inherits from *itk::PointSetToMeshFilter*.

2.3 Initialisation

The Incremental algorithm is a step case algorithm which needs an initialisation step. We initialise $DT(\mathcal{P}_0)$ by creating a four points mesh which encloses all the points of $\mathcal{P}$. Those four points Ω_0, Ω_1, Ω_2 and Ω_3 are at the extremity of the coordinates space of $\mathcal{P}$ (Fig. 2a). This is to make sure that their edges will always respect the criterion and will not influence the triangulation. Once the algorithm will have converged, the points will be removed along with all edges connected to them (Fig. 2f).

Distributed under Creative Commons Attribution License

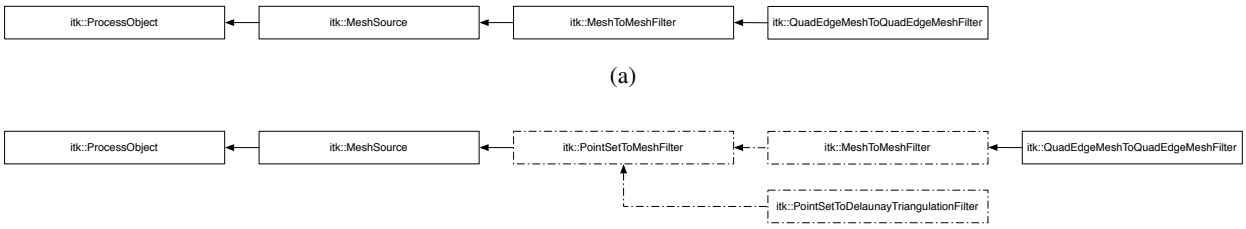

Figure 1: Inheritance diagram. (a) Current inheritance branch in ITK. (b) Modified inheritance branch for our implementation, with the modified classes in dash. The data structure of *itk::PointSet* (points and point data) copy process that was managed in *itk::MeshToMeshFilter* has been extracted and put into a new class *itk::PointSetToMeshFilter* from which our filter *itk::PointSetToDelaunayTriangulation* inherits. The new *itk::MeshToMeshFilter*, that inherits from *itk::PointSetToMeshFilter*, now managed the copy of the rest of the *itk::Mesh* data structure (cells, cell links and cell data).

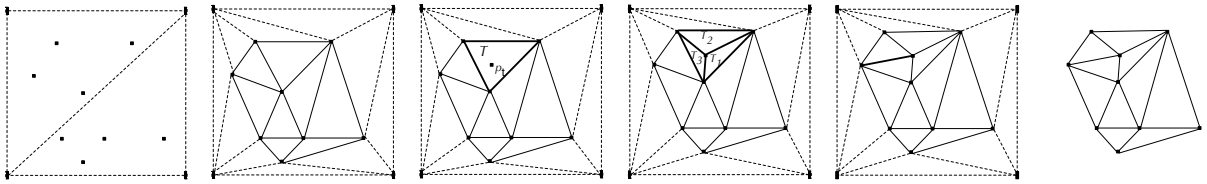

Figure 2: Incremental algorithm iteration. (a) Initialisation step. (b) $DT(\mathcal{P}_t)$. (c) Add a point p_t to $DT(\mathcal{P}_t)$. (d) Create three new triangles T_1, T_2 and T_3. (d) Flip illegal edge in order to obtain $DT(\mathcal{P}_{t+1})$. (e) When all point are processed, remove temporary points from the initialisation step. (f) Final $DT(\mathcal{P})$.

2.4 Main algorithm

Adding a point

At each step of the algorithm, we add a new point p_t to the triangulation (Fig. 2c). First we locate the triangle $T(p_i,p_j,p_k)$ of the current triangulation $DT(\mathcal{P}_t)$ the point p_t is going to affect. This is done using the *itk::WalkInTriangulationFunction* [4]. The triangle T is removed and replaced by the three triangles $T_1(p_i,p_j,p_t)$, $T_2(p_j,p_k,p_t)$ and $T_3(p_k,p_i,p_t)$ (Fig. 2d).

Delaunay Criterion Check

The Delaunay criterion is then checked for the newly created triangle T_1, T_2 and T_3 (Fig. 2e). It uses the *itk::PointInCircleGeometricalPredicateFunctor* [3] and verifies, for the given triangle and point, the emptiness of the circumference circle for the adjacent face and opposite to the given point. If the face is not Delaunay conform, we *flip* the diagonal edge of the quadrilater formed by the triangle and its adjacent triangle using the *itk::QuadEdgeMeshFlipEdgeEulerOperator*. Because the *flip* can affect the validity of other local edge, the verification is recursively called on the two new triangles created from the edge flipping.

Distributed under Creative Commons Attribution License

3 Usage

An example `DelaunayIncremental.cxx` is provided with the sources and is used for the tests. Some points set generator are provided for testing.

```
typedef itk::PointSet< PixelType, Dimension > TPointSet;
typedef itk::QuadEdgeMesh< PixelType, Dimension > TQuadEdgeMesh;
typedef itk::PointSetToQuadEdgeMeshFilter< TPointSet, TQuadEdgeMesh > MyFilter;

TPointSet::Pointer pointset = TPointSet::New();
TQuadEdgeMesh::Pointer triangulation = TQuadEdgeMesh::New();

MyFilter::Pointer myFilter = MyFilter::New();
myFilter->SetInput( pointset );
triangulation = myFilter->GetOutput();
myFilter->Update();
```

4 Validation

The validation of the filter output is made using the *itk::DelaunayConformingQuadEdgeMeshFilter*, by quantifying how many edge flip was necessary to make the filter output mesh Delaunay conform. The number of edge flip done by the *itk::DelaunayConformingQuadEdgeMeshFilter* is expected to be equal to zero.

References

[1] O. Devillers. Improved incremental randomized delaunay triangulation. In *Proceedings of the fourteenth annual symposium on Computational geometry*, pages 106–115. ACM, 1998. (document), 1

[2] A. Gouaillard, L. Florez-Valencia, and E. Boix. Itkquadedgemesh: A discrete orientable 2-manifold data structure for image processing. *Insight Journal*, Sep 2006. (document)

[3] B. Moreau and A. Gouaillard. Exact geometrical predicate: Point in circle. *Insight Journal*, Nov 2011. (document), 2.4

[4] S. Rigaud and A. Gouaillard. Walking in a triangulation: Straight walk. *Insight Journal*, Feb 2012. (document), 2.4

Distributed under Creative Commons Attribution License

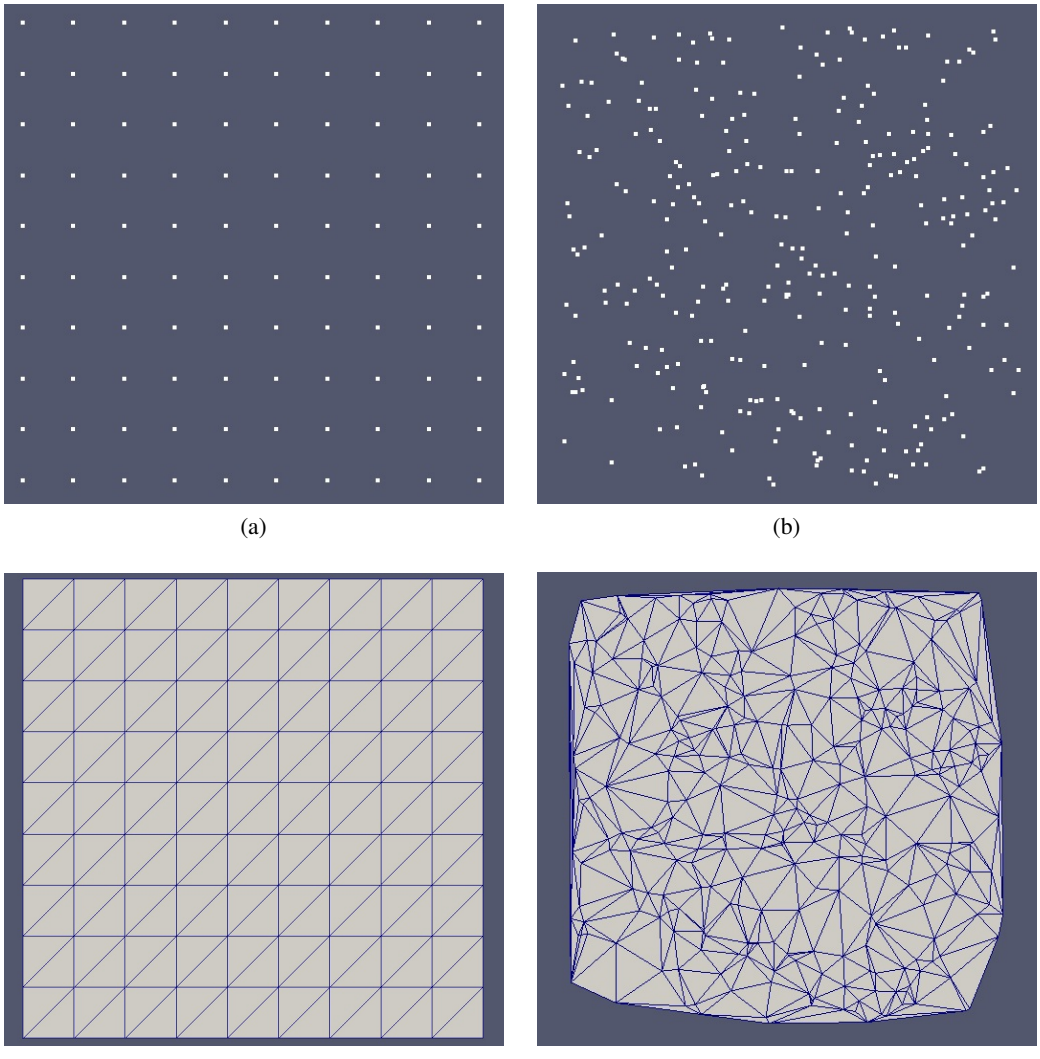

Figure 3: Paraview displays of results. (a) Regular pointset. (b) Randomly generated pointset. (c) Corresponding Delaunay Triangulation of pointset (a). (d) Corresponding Delaunay Triangulation of pointset (b).

Distributed under Creative Commons Attribution License