Walk In A Triangulation : Straight Walk

Release 0.01

Stéphane Rigaud[1] and Alexandre Gouaillard[2]

February 14, 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 *Straight Walk in a Triangulation* algorithm proposed by Devillers *et al.* [1]. Using the *exact discrete geometrical orientation predicate* [3], and the *itk::QuadEdgeMesh* API [2] of ITK , we propose an efficient implementation that locates a point in a triangulated mesh structure. This paper is accompanied with the source code and examples that should provide enough details for users. This work has for principal intended an exact and robust implementation of a Delaunay triangulation / Voronoi tesselation in ITK, which will be presented in a separate paper, once done :).

Contents

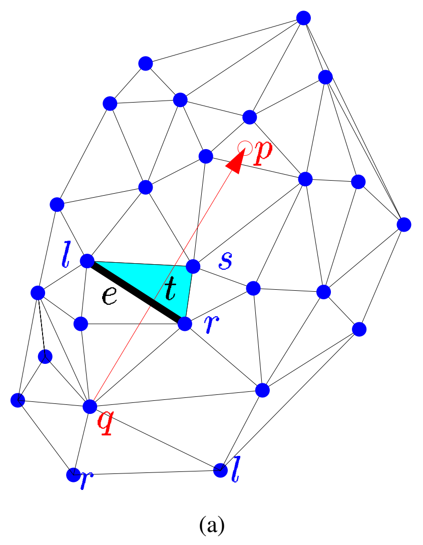
1 Principle of Walking in a Triangulation **1**

2 Implementation **2**

3 Usage **3**

1 Principle of Walking in a Triangulation

Taking a triangulated mesh 2-manifold planar structure $\mathcal{T}$ of n vertices embedded into a n-dimensions space, and supposing that $\mathcal{T}$ has only one component and only one boundary, we try to find the triangle t of $\mathcal{T}$ that contain a given point p. One of the simplest and most effective strategy is the Straight Walk, which consists in, given a specific or random starting point q, going through all the triangles along the direction vector $\overrightarrow{qp}$,



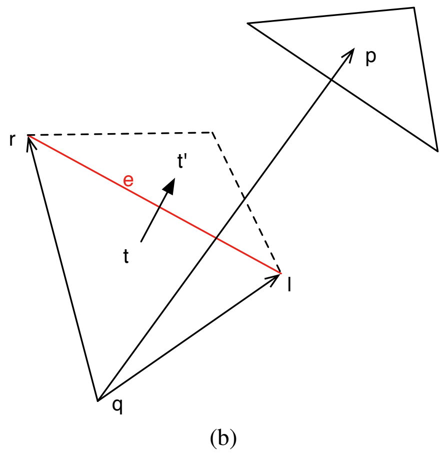


Figure 1: Straight Walk in a Triangulation Algorithm

using adjacency relation between triangles until the triangle that contains the point p is reached (Fig 1a). This algorithm is especially used in Delaunay triangulation as it can be proved that it actually converges to the right triangle in a $O(|qp|\sqrt{n})$ complexity, with n the number of vertices of $\mathcal{T}$, and $|qp|$ the distance between q and p. The path followed by the Straight Walk is determined by an orientation predicate (simple geometric question) that gives us the direction to follow in the triangulation.

2 Implementation

The algorithm is implemented as a functor templated over the mesh type and the output type defined in *itk::QEMeshFunctionBase*. It take in arguments a triangular mesh (*itk::QuadEdgeMesh*), a point coordinate (*itk::QuadEdgeMesh::PointType*) and optionally a starting triangle (*itk::QuadEdgeMesh::CellIdentifier*). It returns a vector of index (*itk::VectorContainer*) that contains the list of the visited triangles' identifiers. The last element of the vector is corresponding to the index of the triangle containing the input point.

From the given initial triangle t, we randomly determine q, one of the vertices of t. We rotate around q until the current triangle incident to q is intersecting with the $\overrightarrow{qp}$ vector. Once t is intersecting with $\overrightarrow{qp}$, we test on witch edge e the vector $\overrightarrow{qp}$ is going out of t using the orientation predicate (Eq. 1). We move to the neighbour triangle of t that share the edge e and test again, in the new triangle, which edge is crossed by the $\overrightarrow{qp}$. The walk stops when no edge crossing $\overrightarrow{qp}$ is found (Fig 1b).

$$orientation(\alpha,\beta,\gamma) = sign\left(\begin{vmatrix}\beta_x-\gamma_x & \alpha_x-\gamma_x\\ \beta_y-\gamma_y & \alpha_y-\gamma_y\end{vmatrix}\right) \tag{1}$$

For robustness and exactness in the algorithm, the predicate is done by using the ITK implementation of an exact discrete geometrical predicate [3] based on the work of Shewchuk [4]. For more details, please refer to the scientific article *Walk in a Triangulation* [1].

3 Usage

An example `WalkInTriangulation.cxx` is provided with the sources and is used for the tests. The functor is templated on 2 dimensions *itk::QuadEdgeMesh*, if a higher dimensions mesh is given, only the two first dimension will be used in the process. The initialisation triangle is optional, if not specified the algorithm will start at the first triangle in the cell container.

```
itk::WalkInTriangulation< TQuadEdgeMesh,
                          itk::VectorContainer<unsigned int, TCellIdentifier> >
( TQuadEdgeMesh *, TPointType &, TCellIdentifier & )
```

References

[1] O. Devillers, S. Pion, and M. Teillaud. Walking in a triangulation. In *Proceedings of the seventeenth annual symposium on Computational geometry*, pages 106–114. ACM, 2001. (document), 2

[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] Jonathan Richard Shewchuk. Adaptive Precision Floating-Point Arithmetic and Fast Robust Geometric Predicates. *Discrete & Computational Geometry*, 18(3):305–363, Oct 1997. 2