Abstract

Meshes extracted from images or resulting from sampling of level sets often exhibit high-frequency "jagged" artefacts. One very popular method to remove that high frequency noise on the surface is to smooth the surface. One of the most common methods of smoothing a surface is Laplacian smoothing, where the position of a point is modified depending on the local connectivity and position of the neighboring points. This paper will describe the implementation in ITK of such a smoothing filter for triangular meshes. Although based on Laplacian smoothing, this filter goes one step further than existing Laplacian filter in VTK for example, by providing the possibility to use internally a Delaunay conforming filter to ensure that the shape factor of the triangles is optimal. Delaunay conforming improves both the quality of the mesh and the quality of the smoothing. Results illustrating this statement on some meshes are provided.

Contents

1 Introduction 1
2 Description 2
3 Implementation Details 2
4 Usage 3
5 Results 3

1 Introduction

Extracting surface mesh models from given volume images or level sets is a common step in many applications, such as visualization, intervention planning and quantification. In many cases surface mesh smoothing is required.
In VTK, another NAMIC toolkit, vtkSmoothPolyDataFilter also implements a Laplacian smoothing. Our implementation is for the QuadEdgeMesh [4] data structure. Our filter does not support feature edges at this time like the VTK filter does, e.g. all edges are filtered equally. However, it implements a lot of different weights to smooth the mesh depending on different quality factors of the local surface. One unique feature that improves greatly the quality of the resulting smooth surface is the ability to use internally a Delaunay Conforming Filter [2]. The Delaunay conforming filter ensure that the triangles have the best form factor possible, and result in optimum configuration for laplacian smoothing.

### 2 Description

itkQuadEdgeMeshSmoothing is a filter that adjusts point coordinates using Laplacian-like smoothing. The effect is to “relax” the mesh. Note that this filter operates only on itkQuadEdgeMesh(es) [4].

The algorithm proceeds as follows. For each point \( p^n_i \) at a given iteration \( n \), a topological analysis (analysis of the one-ring \( N(i) \)) is performed to determine which points \( p^n_j \) are connected to \( p^n_i \), and which cells are connected to the same point. Note that if no Delaunay conforming is used, the topology is constant and \( p^n_j = p_j \). Next, an iteration phase begins over all Points \( p^n_i \). The new coordinates of the point \( p^{n+1}_i \) are modified according to a weighted average of the connected points \( p^n_j \):

\[
p^{n+1}_i = p^n_i + m\cdot\text{RelaxationFactor} \cdot \frac{\sum_{j \in N(i)} \alpha_{ij} \cdot (p^n_j - p^n_i)}{\sum_{j \in N(i)} \alpha_{ij}}
\]  

(A relaxation factor, \( m\cdot\text{RelaxationFactor} \) in \([0,1]\), is available to control the amount of displacement). The weights \( \alpha_{ij} \) can be computed using different formula described shortly in the following section and more in details in [1].

Warning: The Laplacian operation reduces high frequency information in the geometry of the mesh. With excessive smoothing important details may be lost, and the surface may shrink towards the centroid.

### 3 Implementation Details

There is one main class, \texttt{itk::QuadEdgeMeshSmoothing}, that does all the job. It is templated over the Input and Output Mesh Types.

\( m\cdot\text{RelaxationFactor} \) is given by the user, and is in \([0,1]\)

\( m\cdot\text{NumberOfIterations} \) is given by the user

\( m\cdot\text{DelaunayConforming} \) is a boolean that lets the user optimize the triangle aspect ratio in the same time as smoothing, it leads to a better mesh in terms of valence, and triangle aspect ratio. Note that the resulting mesh is then an intrinsic Delaunay triangulation of the surface. Note that this operation can be quite expensive since it is applied at the end of each iteration.

\( \alpha_{ij} \) are spring coefficients taken from \texttt{itk::QuadEdgeMeshParamMatrixCoefficients}. Explanation about their respective advantages can be found in [1].
1. **Graph theory based:** class itk::OnesMatrixCoefficients[6,7]

\[ D_{ij} = 1 \]  

(2)

2. **Chord Length:** class itk::InverseEuclideanDistanceMatrixCoefficients

\[ D_{ij} = \frac{1}{\|p_i - p_j\|^2} \]  

(3)

3. **Discrete Conformal:** class itk::ConformalMatrixCoefficients[5]

\[ D_{ij} = \cot \alpha_{ij} + \cot \beta_{ij} \]  

(4)

4. **Discrete Authalic:** class itk::AuthalicMatrixCoefficients[3]

\[ D_{ij} = \cot \gamma_{ij} + \cot \delta_{ij} \]  

(5)

5. **Intrinsic:** class itk::HarmonicMatrixCoefficients[3]

\[ D_{ij} = \mu \cdot \frac{\cot \gamma_{ij} + \cot \delta_{ij}}{\|p_i - p_j\|^2} + (1 - \mu) \cdot \cot \alpha_{ij} + \cot \beta_{ij} \]  

(6)

4 **Usage**

typedef double Coord;
const unsigned int Dimension = 3;

typedef itk::QuadEdgeMesh< Coord, Dimension > MeshType;

...

// Note that any other coefficients could have been used
itk::OnesMatrixCoefficients< MeshType > coeff0;

typedef itk::QuadEdgeMeshSmoothing< MeshType, MeshType > SmoothingType;
SmoothingType::Pointer filter = SmoothingType::New();
filter->SetInput( input );
filter->SetNumberOfIterations( 10 );
filter->SetRelaxationFactor( 0.5 );
filter->SetDelaunayConforming( true );
filter->SetCoefficientsMethod( &coeff0 );
filter->Update();

5 **Results**

We first ran the algorithm on a relatively simple dataset already included in ITK which represent a tooth (/Insight/Testing/Data/Input/geusZeroSurface01.vtk). We ran it using only 5 iterations, once with and once
Figure 1: Illustration of the smoothing filter output with or without Delaunay conforming. (a) the original surface. One can see the degenerated points. (b) simply smoothed surface. The smoothing operator was performing less well on the degenerated points. (c) smoothing and Delaunay conforming. The sampling of the surface is much more regular, the triangle aspect ratio improved, and the geometry correctly smoothed.

without Delaunay conforming. This dataset is perfect example of what you obtain after a surface extraction algorithm followed by a decimation. Output of surface extraction algorithm have close to regular connectivity, and have good triangle quality factor and usually do not require Delaunay conforming. However, they are tremendously over-sampled, as the sampling is done on the regular grid of the original image. Users hence almost always decimate the mesh, in order to capture the same geometry with the minimum vertices or triangle budget. The decimation algorithms are unfortunately oblivious to connectivity, and result in some very high valence points in the area of low curvature (attraction point for the merging algorithm). The geometry is well captured, up to the artifacts caused by the surface extraction algorithm. However lots of vertices are connected to way more than 6 edges and that is known to cause numerical instabilities for mesh processing algorithms that use the local connectivity graph as a support, like laplacian smoothing. We can see on figure 1 that using the Delaunay conforming during the smoothing improves the connectivity and the triangle aspect ratio.

For the second dataset we used a pelvis model, courtesy of of the Visualization Research Group, University of Magdeburg (http://isgwww.cs.uni-magdeburg.de/rbade/meshsmoothing/data/index.html) This dataset is the output of a surface extraction algorithm and need serious smoothing. Figures 2 show the great improvement in the geometry. No Delaunay conforming as been used here. There was no decimation after extracting the surface and that is why, as explained above, we did not need to apply Delaunay conforming.

Acknowledgment

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

For the second dataset we used a pelvis model, courtesy of of the Visualization Research Group, University of Magdeburg (http://isgwww.cs.uni-magdeburg.de/rbade/meshsmoothing/data/index.html).
References


Figure 2: Illustration of the smoothing. (a) original surface. Usual ”staircase” artifact is visible. (b) smoothed surface.