

Smooth Polylines on Polygon Meshes

Georges-Pierre Bonneau¹ and Stefanie Hahmann²

¹ iMAGIS-GRAVIR, INRIA Rhône-Alpes,
655 Avenue de l'Europe, F-38330 Montbonnot (France)
Georges-Pierre.Bonneau@imag.fr

² Laboratoire LMC-IMAG,
BP. 53, F-38041 Grenoble (France)
Stefanie.Hahmann@imag.fr

Summary. Curves on surfaces can be very useful to visualize surface features at low graphical and memory cost. Curves on surfaces are also used for surface segmentation with possible applications to visualization, reconstruction and parameterization of complex surfaces. In this paper a simple and efficient algorithm for building smooth polylines on triangulated 2D-manifold polygonal meshes is introduced. The algorithm combines geometrical optimization with topological modifications in order to iteratively smooth an initial crude polyline. One key feature of this algorithm is that it relies solely on the geometry of the surface and the polyline. Another key feature is that during the smoothing the polylines always stay on the surface. Different smoothing criteria are proposed.

1 Introduction

Curves on surfaces have applications in many different areas. In visualization, their ability to reveal global surface features or local details at a very low graphical and memory cost has been used to help visualizing large data sets [10]. Curves on surfaces are also important in surface segmentation. The applications of surface segmentation span a large field of research from visualization, to reconstruction and parameterization. Irregular polygonal meshes can be reparameterized on a coarse mesh by first constructing a set of smooth polylines on a fine mesh that yield a segmentation of the mesh into three sided patches. The reconstruction of parametric surfaces from dense polygonal data can be based on the interactive design of a mesh of smooth polylines drawing the boundaries of the patches.

In this paper we present an algorithm for building smooth polylines (PL) on 2D manifold triangular meshes, for definition see section 2. An initial PL is first computed from a set of user specified mesh triangles. This is a crude (non smooth) approximation of the desired PL. Then an iterative execution of geometrical optimization steps and topological steps enables to smooth this initial PL on the surface. During this process the PL always lies exactly on the mesh. A key feature of the algorithm is that it relies solely on

the surface geometry and the PL geometry. It is independent of any surface parameterization or any external prescribed smooth curve. The algorithm has the following features:

- It is a purely geometry driven algorithm. This means that the smoothing criteria as well as the error measurements only depend on the intrinsic geometric properties of the surface or the PL.
- Between two user-specified or automatically computed points on the polygonal surface a smooth PL can be computed by using different smoothing filters.
 - We first introduce a “linear” filter which yields a local discrete geodesic. The visual effect is a global straightening of the initial PL.
 - Then several extensions are possible, we choose to implement a “cubic” smoothing filter. Here the global shape of the initial PL is preserved while small details are smoothed out.
- In any case smooth PL are obtained by iteratively moving the vertices of the PL on the mesh edges (geometrical modification) and by passing through mesh vertices if necessary (topological modification).
- The algorithm is easy to implement. However, the topological steps need to be well understood.

Related Works

There are many papers specifically dedicated to compute curves related to geodesics [11], [13], [7], [2], [12]. Geodesics or close-to-geodesics are not the issue of the present paper, rather we are computing smooth curves that need not to be geodesics.

Krishnamurthy and Levoy [9] build smooth PLs on a polygonal mesh using an external smooth 3D curve as a guide. The use of an external guiding curve may fail in high curvature surface regions ([9] figure 7). Also the PL does not exactly lie on the mesh, only the vertices of the PLs are constrained to lie on the mesh. In [8] Krishnamurthy modifies this method in order to avoid the use of an external guiding curve. But still the PLs don't lie exactly on the mesh, and the optimization relies on a set of user given external forces applied to a polynomial model of the PL, whereas our algorithm relies solely on the surface geometry and the PL geometry.

Eck et al. [5] construct a set of smooth PLs on a fine mesh, that segment the mesh into three sided patches. First a crude version of the PL is computed. The PLs are smoothed in the following manner: a special parameterization of the surface around one PL is used to map it onto a planar domain, then the PL is replaced by the inverse image by this map of a straight line in the parameter domain. Praun, Sweldens and Schröder [14] use the same method to straighten the PL, but with a different parameterization. While Eck et al. use a parameterization based on harmonic maps, Praun et al. use the parameterization introduced by Floater [6]. By contrast our algorithm does not

depend on any parameterization of the triangle mesh, and is much simpler to implement.

The paper is organized as follows. In section 2, the notations for a PL on a 2D manifold triangulation are given. Section 3 outlines the algorithm. Section 4 is dedicated to the computation of the initial PL. Section 5 describes the geometrical optimization that is used by the algorithm. Section 6 explains the topological steps that are required by the algorithm. Results are given in section 7. Finally section 8 concludes the paper and gives possible future works.

2 Notations

The triangles in the mesh are denoted T_i , and the edges E_i . A *triangle strip* is defined by $n + 1$ adjacent triangles T_1, \dots, T_{n+1} in the mesh. Let E_1, \dots, E_n denote the common edges between adjacent triangles in the triangle strip. A polyline is defined as the combination of a strip and a set of n scalars $\alpha_1, \dots, \alpha_n$ that store the barycentric coordinate of its vertices V_1, \dots, V_n on the edges E_1, \dots, E_n , with respect to the orientation of the triangles T_1, \dots, T_n . A triangle strip can be efficiently stored in memory with one pointer to the first triangle, and n bits to encode the next edge in the strip. Figure 1 illustrates these notations.

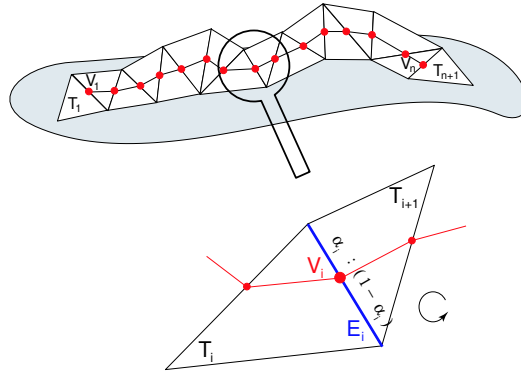


Fig. 1. Notations. Triangle strip, PL, barycentric coordinates of PL vertices.

3 Overview of the algorithm

This section gives an overview of the PL design and smoothing process. This process is summarized in fig. 2 and illustrated in fig. 4.

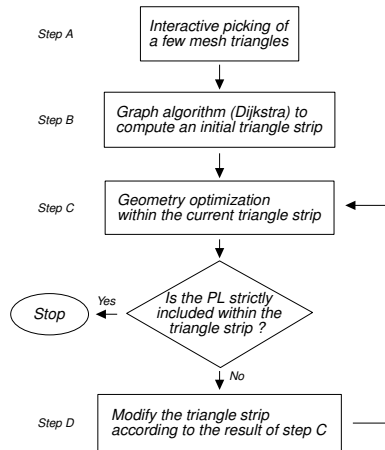


Fig. 2. Overview of the algorithm.

Step A: The user specifies a few triangles in the mesh, not necessarily adjacent. The order of the picked triangles is relevant: the initial triangle strip will interpolate these triangles in the given order. Note that these initial triangles can be interpolated by the final PL, if desired.

Step B: A Dijkstra algorithm ([4], [1]) is performed, that outputs a triangle strip, i.e. a sequence of adjacent triangles joining the triangles specified in step A. The output of step B is a triangle strip that gives a crude initial PL. Each vertex of this initial PL is chosen as the mid point of the edges between consecutive triangles in the strip (see fig. 4).

Steps C and D are looped until an exit criterion is fulfilled.

Step C: This is the geometrical part of the algorithm. The PL is optimized within the current triangle strip. The vertices of the PL are constraint to stay on the same edge during step C. Different criteria can be used for this geometrical optimization. The result of step C is a PL that can either be strictly included in the current triangle strip, or that can touch the boundary of this strip. If it is strictly included in the triangle strip, the algorithm stops and outputs the final PL.

Step D: This is a topological part of the algorithm. The output of step C is a PL in which some of the vertices have merged with the mesh vertices. This happens when the geometrical optimization performed in step C has pushed some vertices of the PL towards one of the two end points of the edge where these PL vertices lie on. This situation is illustrated in fig. 3. Around such vertices, a sequence of triangles of the current triangle strip is replaced by a

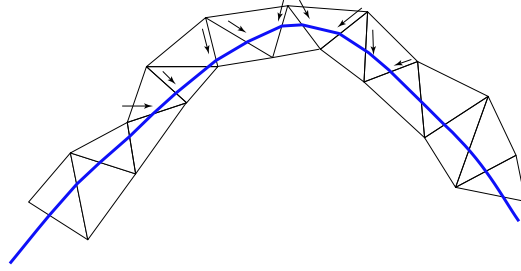


Fig. 3. *The geometrical optimization step C may push some of the PL vertices towards the mesh vertices. In such cases, the triangle strip has to be modified, which is the purpose of step D of the algorithm, see fig. 2.*

new sequence of triangles. The resulting new sequence of triangles is still a triangle strip and it still includes the PL.

Remark: It can happen that the polyline has one or several vertices merging with the mesh vertices. In this case, the exit criterion (see step C) is never fulfilled, since the optimal polyline is not strictly included in the triangle strip. As a consequence, there could be a cycle in steps C, D. In order to avoid such a behaviour, a general method would be to identify such cycles, and to stop the iteration then. In the current implementation the users fixes only a maximal number of iterations. In all examples we have tried, a maximal number of 100 iterations was fully sufficient.

4 Computing an initial triangle strip.

The first step of the algorithm consists of computing the initial crude triangle strip. One could imagine different methods to define an initial triangle strip. One could for example draw a planar curve on the graphic window where the triangle mesh is displayed, and use this curve in order to pick triangles from the mesh. This could be implemented very efficiently using graphic hardware. Another possibility would be to draw a space curve, near the surface mesh, and to select triangles on the surface according to some projection mapping. In either case, the sequence of picked triangles would not be guaranteed to form a triangle strip. In regions of high surface curvature for example, the projection of the planar or the space curve could lead to sequences of triangle strips with "breaks" between them, see fig. 5. Thus it is necessary to define an algorithm that can compute a triangle strip between any two given triangles in the mesh.

We first build the graph in which the nodes are labelled with the mesh triangles, and edges connect nodes that correspond to adjacent triangles. This is an unoriented graph, the maximum degree of nodes is 3 since the triangles

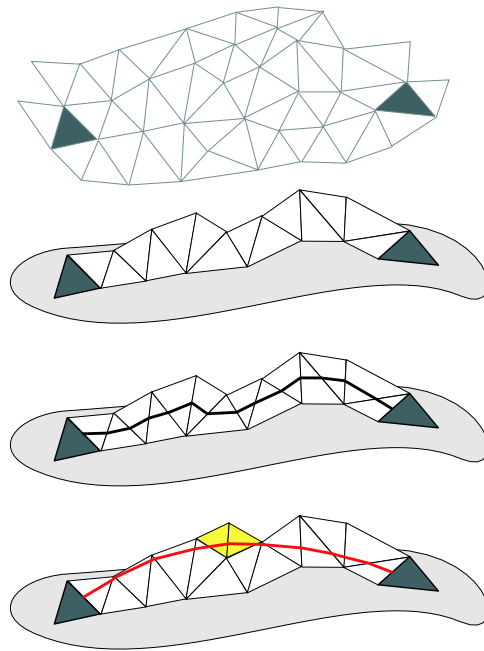


Fig. 4. Overview of the algorithm. (a) the user selects triangles in the mesh. (b) a initial triangle strip is computed. (c) an initial PL is build from the edge mid points of the triangle strip. (d) geometrical optimization and topological steps yield a smooth PL.

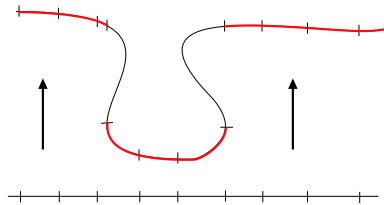


Fig. 5. Using an external curve in order to pick triangles on a polygonal mesh may fail in regions of high surface curvature. It may produce triangle strips with breaks.

have up to 3 neighbours. Fig. 6 shows this graph for a simple triangle mesh. In our current implementation, we assign weights to the edges of the graph. These weights are chosen equal to the distance between the barycenter of two adjacent triangles. Since the edges on this graph connect adjacent triangles, a path in this graph corresponds to a triangle strip on the mesh. Then a Dijkstra algorithm is applied to the graph in order to compute the shortest path between the nodes which correspond to the given pair of triangles. This path gives the desired triangle strip.

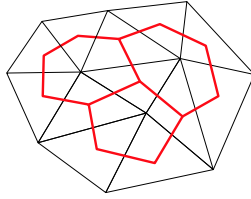


Fig. 6. *The triangulation and the dual graph is used for the computation of the initial triangle strip. All edges of this dual graph are weighted by the distance between the barycenter of the triangles.*

As explained in the overview, we apply this procedure to each pair of triangles (T_i, T_{i+1}) in a sequence of triangles T_1, \dots, T_n chosen by the user. Fig. 7 shows different triangle strips interpolating the first and last triangle. The top one only interpolates these two triangles. The middle one interpolates also one triangle below the ear, and the bottom one interpolates one triangle between the ear and the horn.

5 Geometry optimization of the polyline

The kernel of the algorithm consists of the iterative application of the geometrical optimization of the PL inside its current triangle strip followed by the topological modification of the strip. This section is dedicated to the geometrical optimization step, which is itself also an iterative process. It successively selects vertices of the PL according to some error metric, and it slides the position of this selected PL vertex along its mesh edge, in order to minimize the error. During this process all vertices are constrained to stay on their edge. This is why the triangle strip of the PL is fixed. This process is efficiently implemented using a heap data structure [3]. After the modification of one vertex, the new error in this vertex always vanishes, and the errors at the neighbouring vertices have to be updated. The process stops when the maximum error falls below a user-given threshold value. Note that, in case the user wants to interpolate some vertices, we can simply omit to add these vertices in the heap. Hence these vertices remain fixed in the PL.

Error criteria

In [9] the error criterion depends on an external space curve, or on some external forces applied to the PL. In [5], [14] the polygon is smoothed using some parameterization of the surface, and modifying the PL in the parameter domain. By contrast our error criterion doesn't depend on any parameterization, it relies solely on local geometrical properties.

We have to compute one error per vertex, and we have to find the new position of the vertex along its edge that minimizes the error. In or-

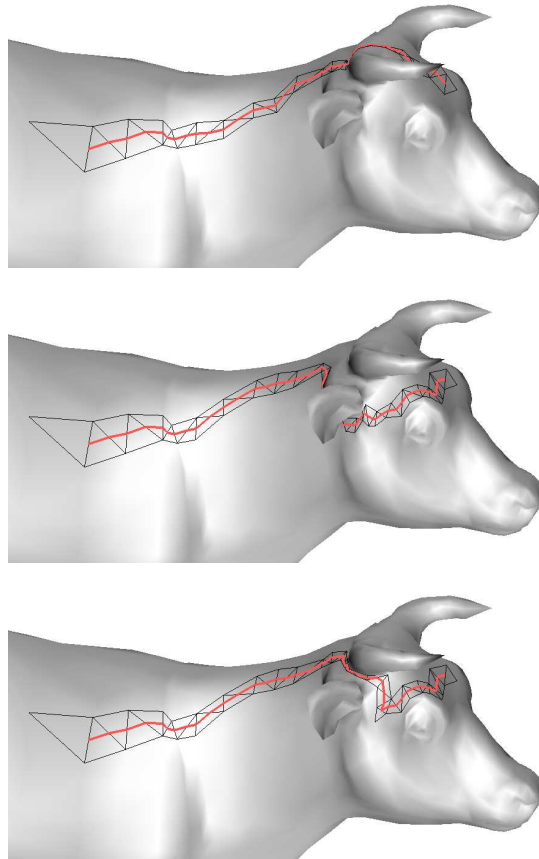


Fig. 7. *Different triangle strips interpolating the same first and last triangles. The top one only interpolates these two triangles. The middle one interpolates in addition one triangle below the ear, and the bottom one interpolates one triangle between the ear and the horn.*

der to do this the following method is applied. Let V_i denote the vertex that has to be optimized, and E_i the mesh edge on which V_i lies. First, a continuously defined curve C is computed from the neighbouring vertices $V_{i-k}, V_{i-k+1}, \dots, V_i, V_{i+1}, \dots, V_{i+k}$. Then V_i is replaced by the point on the edge E_i that is closest to the curve C . Note that the curve C doesn't need to be on the triangle mesh. The error in vertex V_i is chosen as the absolute value of the difference between the barycentric coordinate of V_i and the barycentric coordinate of the optimized point. In the current implementation, two different choices of C are offered to the user.

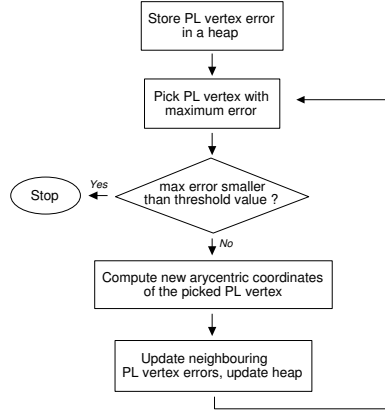


Fig. 8. Overview of the geometrical optimization step.

Linear smoothing filter

The first choice for C is simply the straight line between the vertices V_{i-1} and V_{i+1} . We will refer to this as the linear choice. This is illustrated in fig. 9. The effect of the linear choice for C is intuitive: it tends to straighten the PL. And in fact, this choice leads to PL that is a straight line if the mesh is planar, and that approximates geodesic paths for non planar meshes. Examples of results are given in section 7.

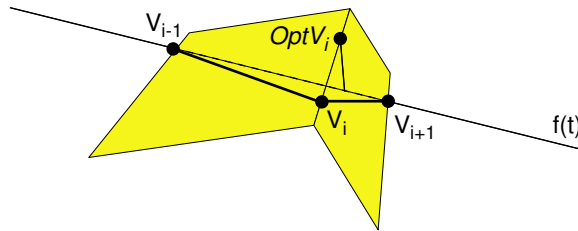


Fig. 9. The optimal position of V_i with the linear filter is the closest point on the edge to the line V_{i-1}, V_{i+1} .

Cubic smoothing filter

Straightened curves are not always desirable. Instead of straightening the PL, it could be useful to smooth it, while keeping its global shape. This is the case for example if the PL is closed. The linear choice for C would shrink the curve until it collapses to its centre of gravity. For all these reasons, a second choice for the curve C has been introduced: It is the unique cubic curve that interpolates $V_{i-2}, V_{i-1}, V_{i+1}, V_{i+2}$ (with uniform parameterization), see fig.

10. Since in this case C depends on a larger filter of points around V_i , it tends to smooth out the fine details, but doesn't change the overall shape of the PL. In fact if all points of the PL lie on a cubic curve, then the PL is not modified at all. In other words, this choice of C ensures that the scheme has a cubic precision.

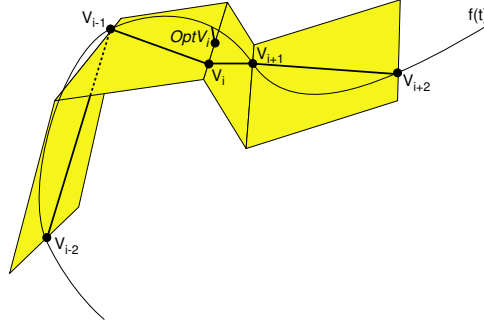


Fig. 10. *The optimal position of V_i with the cubic filter is the closest point on the edge to the unique cubic polynomial interpolating $V_{i-2}, V_{i-1}, V_{i+1}, V_{i+2}$.*

6 Topological steps

The output of the geometrical optimization step is a PL in which some of the vertices have merged with the mesh vertices. This situation is illustrated in fig. 3. The limit case is shown in fig. 11: the PL has smooth parts strictly included in the current triangle strip, and non-smooth parts that touch the boundary of the strip.

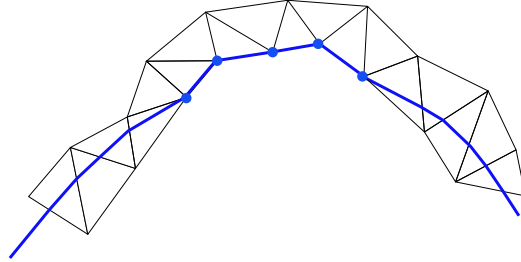


Fig. 11. *The results of the geometrical step is a PL which has smooth parts strictly included in the triangle strip, and non smooth parts along the boundary of the strip.*

The triangle strip has then to be modified, which is a topological step. This topological modification has been implemented in the following manner.

First we select the mesh vertices which have merged with the PL vertices. These vertices are highlighted in red in fig. 12. Then we find the edges of the triangle strip that are adjacent to these vertices. These edges are highlighted in red in fig. 12.

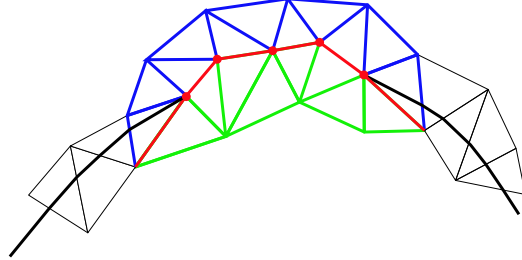


Fig. 12. *Topological step. The red points show the PL vertices that have merged with the mesh vertices. The red edges show the triangle strip edges that are adjacent to these vertices. The green triangles are included in the new triangle strip. They replace all but the first and the last blue triangles. (see colorplate)*

Let T_1, \dots, T_k denote the whole triangle strip, and T_i, \dots, T_{i+n} denote the sequence of triangles adjacent to the selected (red) edges, that are part of the strip. T_i, \dots, T_{i+n} are highlighted in blue in fig. 12. Let $\tilde{T}_1, \dots, \tilde{T}_m$ be the sequence of triangles adjacent to the selected edges, but not part of the strip. The new triangle strip is now $T_1, \dots, T_i, \tilde{T}_1, \dots, \tilde{T}_m, T_{i+n}, \dots, T_k$. Note that the new triangle strip still includes the PL. The geometry of the PL is not modified, but its topology is changed: the vertices that have merged with the mesh vertices are now encoded relative to the edges of the new triangle strip. In particular, if the barycentric coordinate of the PL vertices was zero (resp. one) relative to the old triangle strip, it is equal to one (resp. zero) with the new triangle strip.

Fig. 17-right shows an example of topological steps. The PL after geometrical optimization is in black, the blue PL shows the portion of the current triangle strip that has to be removed and the red PL shows the new portion of the triangle strip.

Loops in triangle strip

An issue in this topological modification step is the possibility of creating loops in the triangle strip. This happens whenever the triangle strip has a high curvature, see fig. 13. This is a classical problem in Computer Aided Geometric Design, that is related to the computation of offset curves. To avoid this problem in the algorithm, a further modification of the triangle

strip is conducted, that consists in searching for possible loops and deleting them.

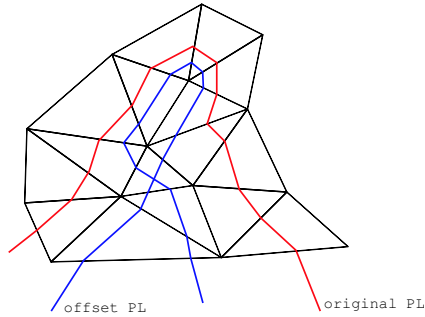


Fig. 13. *The topological modification can lead to loops in the triangle strip. These loops are deleted in an additional step.*

7 Results

Most of the time spent for building smooth PL is due to the user’s choice of the input triangles that the PL will interpolate. Once this is done, building the initial triangle strip takes theoretically $O(n \log m)$ where n is the number of mesh vertices and m the number of mesh edges. Practically it took less than 0.5s on a 1 GHz PIII even for the ball joint mesh that has 137062 vertices. The time spent for the iterative optimization process generally depends on the shape of the initial PL. Practically, all optimizations took less than 5s for all the examples of this paper.

Most of the meshes shown in this paper are courtesy Cyberware and can be downloaded at <http://www.cyberware.com>. Fig. 14 and fig. 15 show a typical example where discrete geodesics can not be used, since the PL is closed. The mesh (a ball joint) has 137062 vertices and 274120 faces. The initial crude PL (left in fig. 14 and 15) interpolates 30 user specified triangles. The images on the right in fig. 14 and 15 show the result of our “cubic” filter. Notice in fig. 15 how the global shape of the initial PL is preserved, while small details are smoothed out.

Fig. 16 (a,b,c) show three PLs obtained using the “linear” filter on the initial PLs shown in fig. 7. This example illustrates the fact that our iterative algorithm prevents the PL from jumping over obstacles. It is not able to find shortest paths. The user can precisely choose where the PL should lie: if he/she wants the PL to traverse between the ear and the horn, he/she only

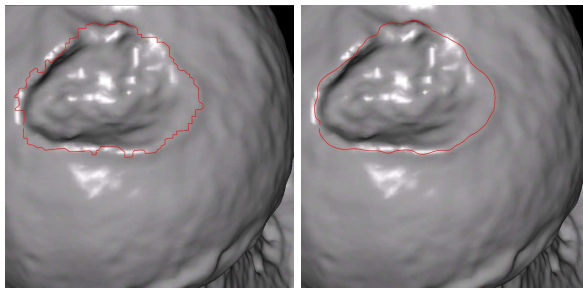


Fig. 14. Ball joint mesh, 137062 vertices, 274120 faces. The initial PL shown left interpolates 30 user specified triangles. The smooth PL shown right is obtained using our cubic smoothing filter. This is a typical example where discrete geodesics can not be used.

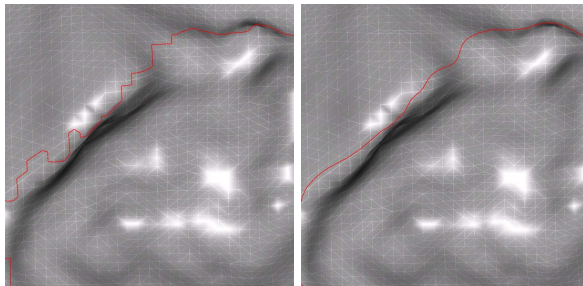


Fig. 15. Local zoom on the top left corner of fig. 14. The mesh edges are shown in white.

has to select one triangle in this area during the initial triangle strip construction, and the PL will remain in this area throughout the optimization process.

Fig. 17-left shows the PL at different stage of the optimization process. The initial PL is in red, and the final PL is in blue. Fig. 17-right illustrates the topological steps of the algorithm.

Fig. 18 illustrate a segmentation. It shows top a set of initial PLs, and bottom the final smooth PLs using the linear filter. Such a curve net can be useful for building a mesh parameterization.

8 Conclusion

In this paper a very simple and efficient algorithm for building smooth PLs on 2D-manifold triangulations has been introduced. By contrast with prior works, this algorithm is based purely on intrinsic geometric properties of the mesh and the PL, and in particular it does not rely on any parameterization.

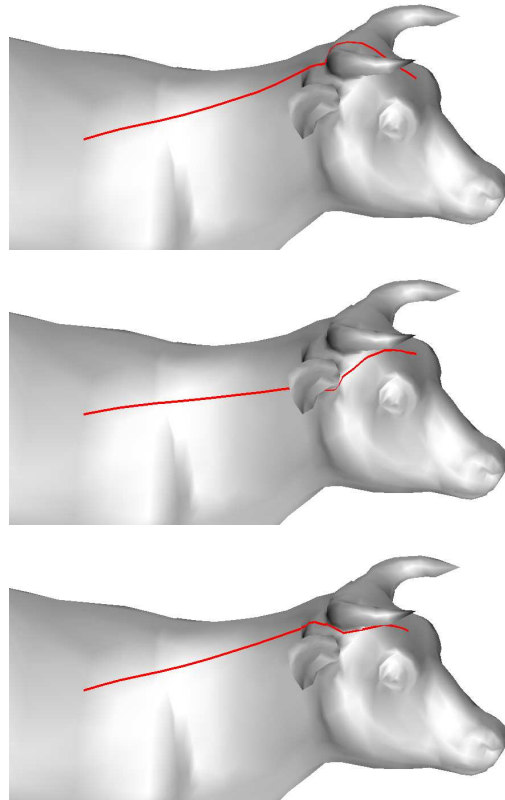


Fig. 16. *The three smooth PL are obtained using the linear filter with the initial PL shown in fig. 7. Our algorithms prevents the PL from jumping over obstacles.*

Two different smoothing filters were proposed, one that tends to straighten the PL, and the other that removes small details while preserving the overall shape of the PL.

In future works we will investigate other kind of filters, with larger support, and we will use the results of this paper as a tool for segmentation and visualization purposes.

Acknowledgments

This work was partially supported by the European Community 5-th framework program, with the Research Training Network MINGLE (Multiresolution IN Geometric modELing, HPRN-1999-00117). Special thanks to Alex Yvart for his Dijkstra code. Models are courtesy Cyberware.

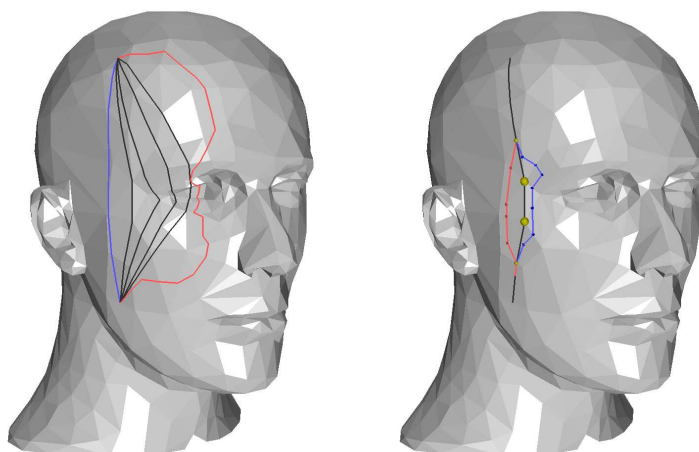


Fig. 17. *Iterative smoothing process (left). The topological modifications (right). Details are given in section 6 (see colorplate).*

References

1. AHO, A., AND ULLMAN, J. *Data structures and algorithms*. Addison-Wesley, 1979.
2. ALEKSANDROV, A. D., AND ZALGALLER, V. A. Intrinsic geometry of surfaces. *Translation of Mathematical Monographs 15* (1967).
3. CORMEN, T., LEISERSON, C., AND RIVEST, R. *Introduction to Algorithms*. MIT Press, 1990.
4. DIJKSTRA, E. A note on two problems in connection with graphs. *Numer. Math 1* (1959), 269–271.
5. ECK, M., DEROSE, T., DUCHAMP, T., HOPPE, H., LOUNSBERY, T., AND STUETZLE, W. Multiresolution analysis of arbitrary meshes. In *Proceedings of SIGGRAPH 1995* (1995), Computer Graphics Proceedings, Annual Conference Series, ACM, ACM Press / ACM SIGGRAPH, pp. 173–182.
6. FLOATER, M. Parameterization and smooth approximation of surface triangulations. *CAGD 14* (1997), 231–250.
7. KIMMEL, R., AND SEITHAN, J. Fast marching method on triangulated domains. *Proceedings of the National Academy of Science 95* (1998).
8. KRISHNAMURTHY, V. *Fitting Smooth Surfaces to Dense Polygon Meshes*. PhD thesis, Stanford University, 2000.
9. KRISHNAMURTHY, V., AND LEVOY, M. Fitting smooth surfaces to dense polygon meshes. In *Proceedings of SIGGRAPH 1996* (1996), Computer Graphics Proceedings, Annual Conference Series, ACM, ACM Press / ACM SIGGRAPH, pp. 312–324.
10. MA, K.-L., AND INTERRANTE, V. Extracting feature lines from 3d unstructured grids. In *Proceedings of Visualization 1997* (1997), IEEE, pp. 285–292.
11. MITCHELL, J., AND MOUNT, D. The discrete geodesic problem. *SIAM J. Comput. 16* (1987), 647–668.

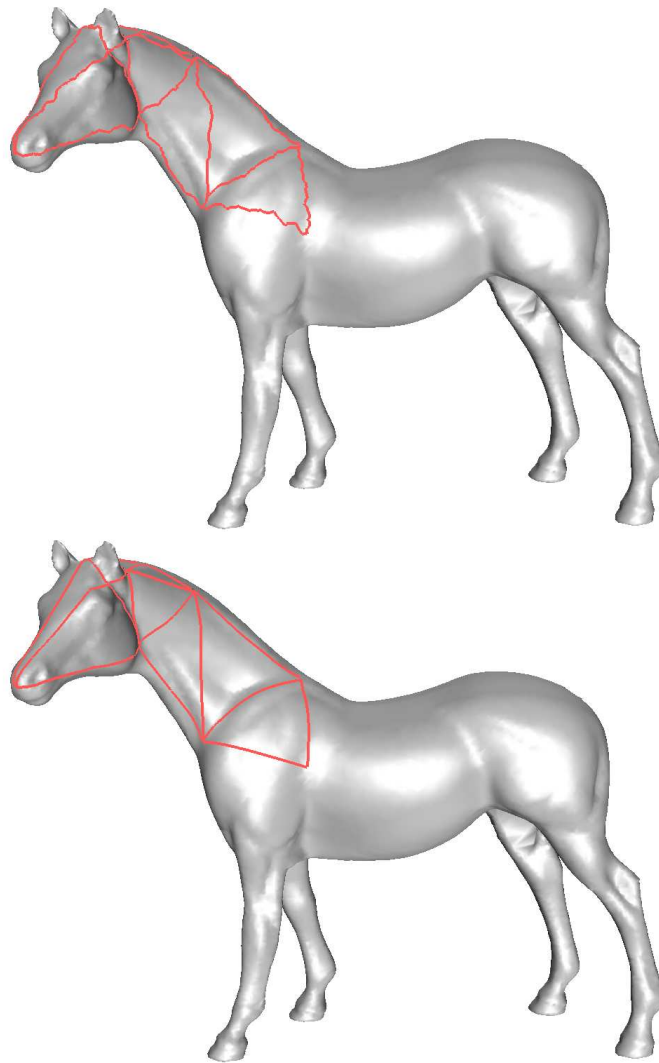
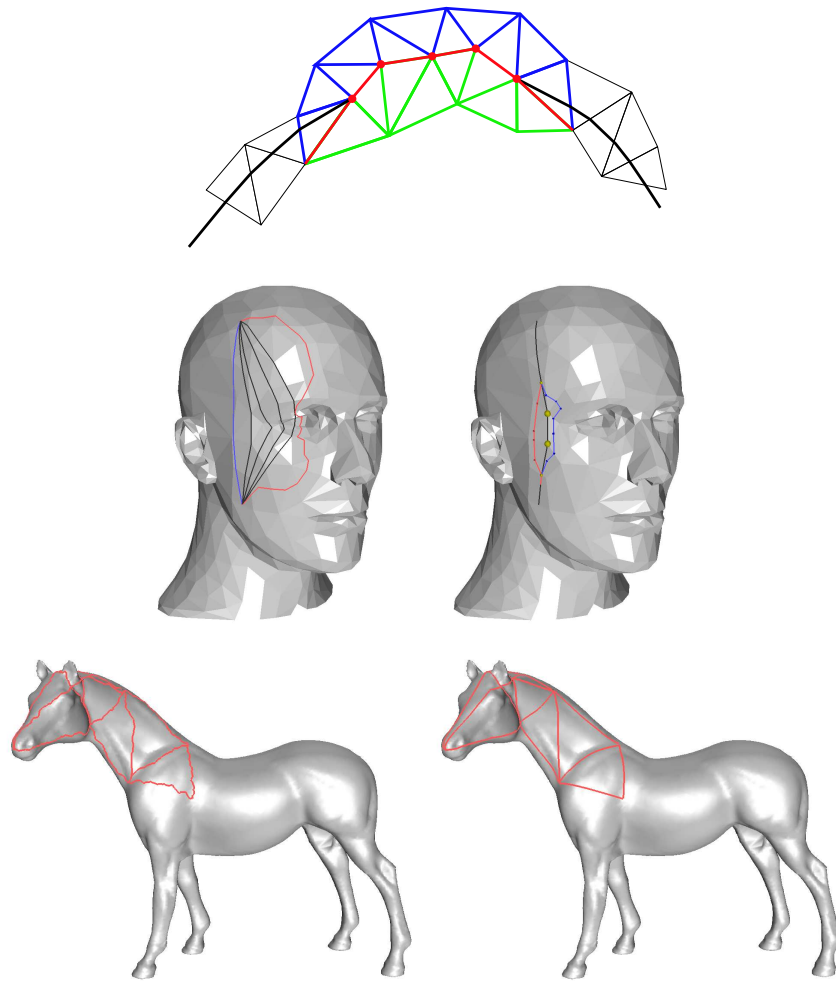


Fig. 18. The top image shows the layout of 19 initial PL connecting 10 user specified triangles. The bottom image shows the final smooth PL using the linear filter. Such a net of smooth PL may be used for surface parameterization.

12. PHAM-TRONG, V. *Détermination géométrique de chemins géodésiques sur des surfaces de subdivision*. PhD thesis, Université Joseph Fourier, Grenoble, 2001.
13. POLTHIER, K., AND SCHMIES, M. Straightest geodesics on polyhedral surfaces. In *Mathematical Visualization, 1998*. In H.C. Hege, K. Polthiers (eds.), Springer, 1998.
14. PRAUN, E., SWELDENS, W., AND SCHRÖDER, P. Consistent mesh parameterizations. *Computer Graphics Proceedings (SIGGRAPH 01)* (2001), 179–184.



Top: Topological step. The red points show the PL vertices that have merged with the mesh vertices. The red edges show the triangle strip edges that are adjacent to these vertices. The green triangles are included in the new triangle strip. They replace all but the first and the last blue triangles.

Middle: Iterative smoothing process (left). The topological modifications (right). Details are given in section 6.

Bottom: The top image shows the layout of 19 initial PL connecting 10 user specified triangles. The bottom image shows the final smooth PL using the linear filter. Such a net of smooth PL may be used for surface parameterization.