Local Geodesic Parametrization: An Ant's Perspective

Lior Shapira
Tel Aviv University

Ariel Shamir
The Interdisciplinary Center

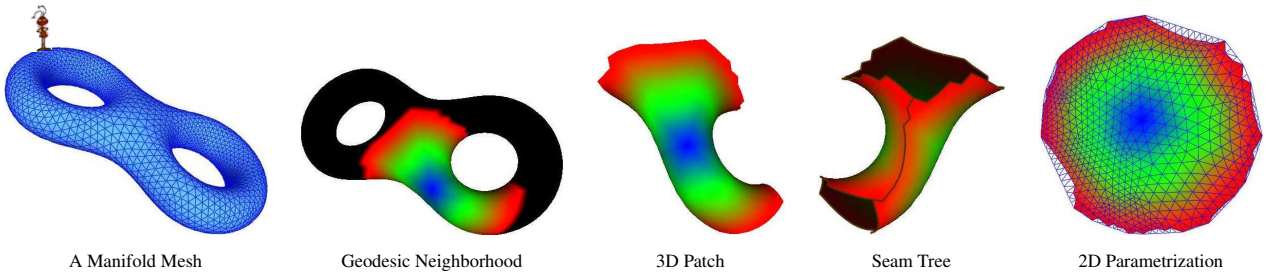


Figure 1: Overview of local geodesic parametrization.

Abstract

Two dimensional parameterizations of meshes is a dynamic field of research. Most works focus on parameterizing complete surfaces, attempting to satisfy various constraints on distances and angles and produce a 2D map with minimal errors. Except for developable surfaces no single map can be devoid of errors, and a parametrization produced for one purpose usually doesn't suit others.

This work presents a different viewpoint. We try and acquire the perspective of an ant living on the surface. The point on which it stands is the center of its world, and importance diminishes from there onward. Distances and angles measured relative to its position have higher importance than those measured elsewhere. Hence, the local parametrization of the geodesic neighborhood should convey this perspective by mostly preserving geodesic distances from the center. We present a method for producing such overlapping local-parametrization for each vertex on the mesh. Our method provides an accurate rendition of the local area of each vertex and can be used for several purposes, including clustering algorithms which focus on local areas of the surface within a certain window such as Mean Shift.

1 Introduction

Contrasted with the modern evidence that the Earth is round, the ancient belief that the Earth is flat is reasonable from the point of view of a person standing on it. Similarly, an Ant standing on a manifold surface in 3D has a particular perspective concerning the shape of the surface it lives on. The distances and angles to points which lie on the surface relative to the Ant's position create a perspective map of the neighborhood surface. We call this map local-geodesic parametrization and in this work we present a method to create such maps for any position on a manifold mesh.

A manifold is a topological space that is locally Euclidean. This means that around every point, there is a neighborhood that is topologically the same as the open unit ball in $\mathbb{R}^n$ (taken from [Weisstein 2004]). In $\mathbb{R}^3$ a manifold surface-mesh is locally homeomorphic to an open unit circle. The basic problem of parametrization is finding a map from 2D Euclidean space to the surface-mesh or to sub-parts of the mesh with minimum distortion of lengths and/or angles between points on the mesh.

Numerous solution have been proposed focusing on different aspects of the field, achieving results which suit different objectives, and using different methods. In several important papers Floater presented barycentric coordinates as a way to allow vertices in triangulations to be expressed as convex combinations of their neighbors [Floater 1995]. This was later enhanced by the mean value coordinates in [Floater 2003]. Such constraints can be solved as a sparse linear system of equations. Levy and Mallet presented a method to solve the same convex combination problem as a minimization problem [Levy and Mallet 1998]. Several papers such as Sheffer and Sturler [Sheffer and de Sturler 2000], and Zigelman et al. [Zigelman et al. 2002], concentrate on parametrization with free boundaries. Lee et al. [Lee et al. 2002] present parametrization with virtual boundaries. In this method a surface homeomorphic to a disc which is to be parameterized is surrounded with one to several layers of virtual vertices and edges. These virtual vertices lie on a convex boundary, allowing more flexibility in relaxation of the parameterization, achieving better results. Other works have focused on processing the mesh to make it suitable for parameterizations and improve the parametrization results. For example, by introducing seams to an arbitrary mesh, the mesh is modified to be homeomorphic to a disc and the parametrization distortion is reduced [Sheffer 2002] and [Erickson and Har-Peled 2002]. For a thorough review of the field the reader is referred to a recent survey by Floater and Hormann [Floater and Hormann n. d.].

In fact, most previous papers concentrate on minimizing the angle, length or area distortions on a global scale. In contrast, in this work we are interested in minimizing the distortion with a bias towards a specific position (the Ant's viewpoint) and locally in its neighborhood. This gives raise to a different type of parametrization which can be seen as a perspective geodesic map.

Figure 1 gives an overview of our method. The algorithm starts from a vertex which is the central position of the map. Using a front marching method a surface patch with a given geodesic distance is built around the vertex. If the patch is not homeomorphic to a disk, seams are introduced in order to have only one boundary loop. Next, the vertices of the boundary loop are placed on a 2D circle according to their angle and then moved to their true geodesic distance from the center. Filler vertices and triangles are added to create a new convex boundary. Lastly, mean-value coordinates are used to calculate the parametrization inside the patch.

The algorithm is described in details in Section 2. Next, in Section 3 we discuss several implementation issues. We give an example for the use of such maps for geodesic mean shift in Section 4, and conclude in Section 5.

2 The Algorithm

2.1 Building the Neighborhood Patch

The first step in the algorithm is to define the geodesic neighborhood of the vertex. This is done using a front marching algorithm similar to the one used in [Kimmel and Sethian 1998]. The algorithm starts from the vertex and expands along a breadth first front. At all times the algorithm keeps a set of vertices for which geodesic distance has been found (*fixed*), a set of vertices which are on the advancing front (*close*) and a set of vertices whose distance is yet unknown (*far*). The front advances until it reaches the geodesic distance defined as maximum radius (Figure 2(a)–(g)). At this stage, all fixed vertices are added to the local geodesic neighborhood. Note that if the front meets itself, it merges to produce a new unified front. This will mean that there is a need to cut seams in the patch in later stages (Figure 2(d)–(f)).

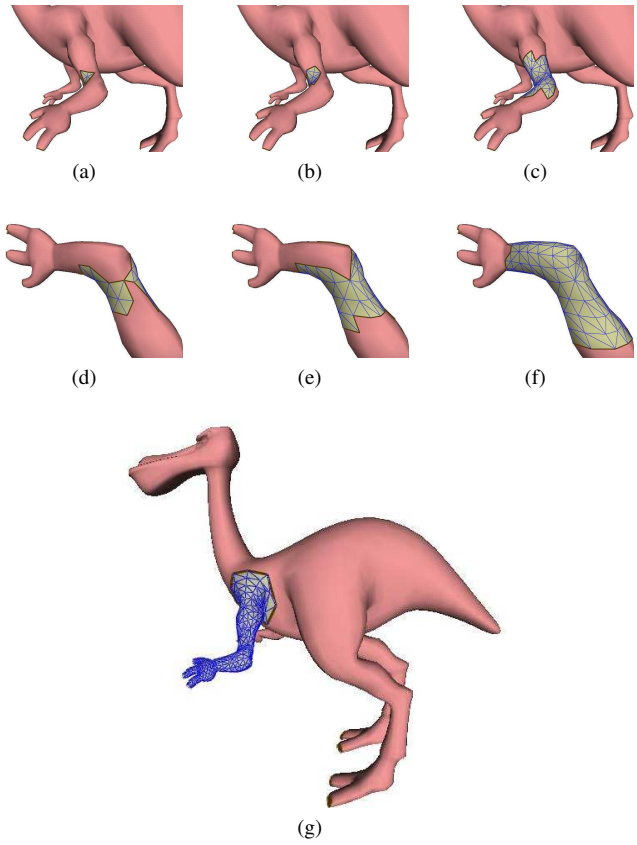


Figure 2: Building the geodesic neighborhood using the front marching method.

The output of this step is a collection of vertices and edges which comprise a vertex's local geodesic neighborhood within a given radius. Even though the mesh is manifold, the computed geodesic neighborhood will not, in general, be homeomorphic to a disk. This is the case when the neighborhood radius is large or the mesh is complex (e.g. fingers of a hand). To create a patch which is homeomorphic to a disk, we wish to ensure that it includes only one boundary loop. This boundary loop will become the boundary of the patch's local geodesic map, and define the patch's shape.

Based on [Sheffer 2002] we attempt to connect the different boundary loops using two steps. First, we build a set of essential vertices which must be in the seams. This set includes all the vertices sitting on the different boundary loops, but also specific vertices which were marked during the front marching algorithm. These are called *maxima vertices* and are vertices where the front merged with itself (see Figure 2(d)). Next, we assign weights to all edges in the patch. Boundary edges are assigned a small weight of $\epsilon > 0$ and other edges are given weight relative to the distance from the center vertex. We use all-pairs-shortest path to find the shortest paths between the subset of vertices. In the second step, a minimum cost spanning tree of these paths is created. This tree connects all the vertices we marked as essential in the previous step creating the seams needed to cut the patch (Figure 3). It will include all the boundary loops, and by including maxima vertices, this process guarantees that the seams occur at 'natural' boundaries of the patch, i.e. along maxima ridges of the geodesic distance from the center.

To create a disk-like shape, the patch is cut along the branches of the seam tree. Vertices from which several branches in the seam tree grow are duplicated and maintain a reference to their original vertex. The resulting patch has only one boundary loop, hence it is homeomorphic to a disc and can be parameterized.

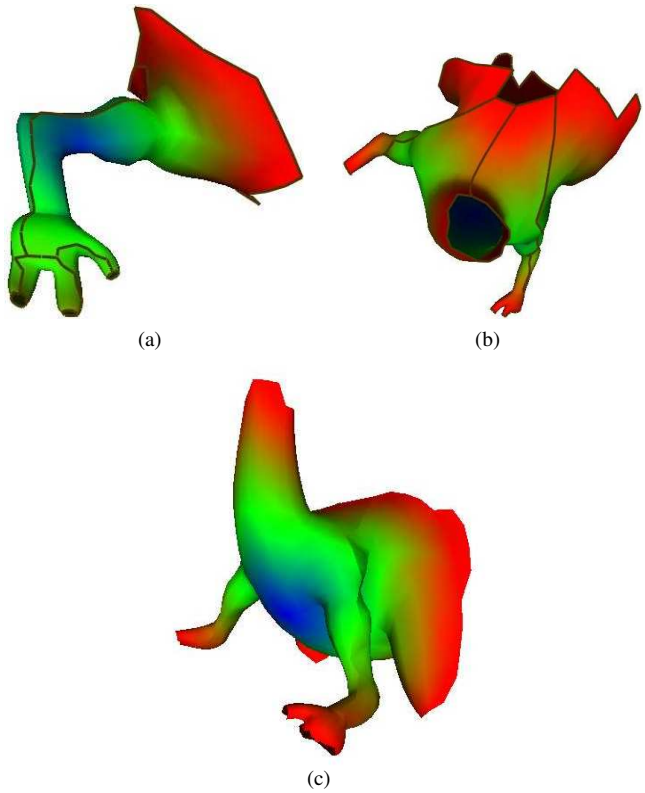
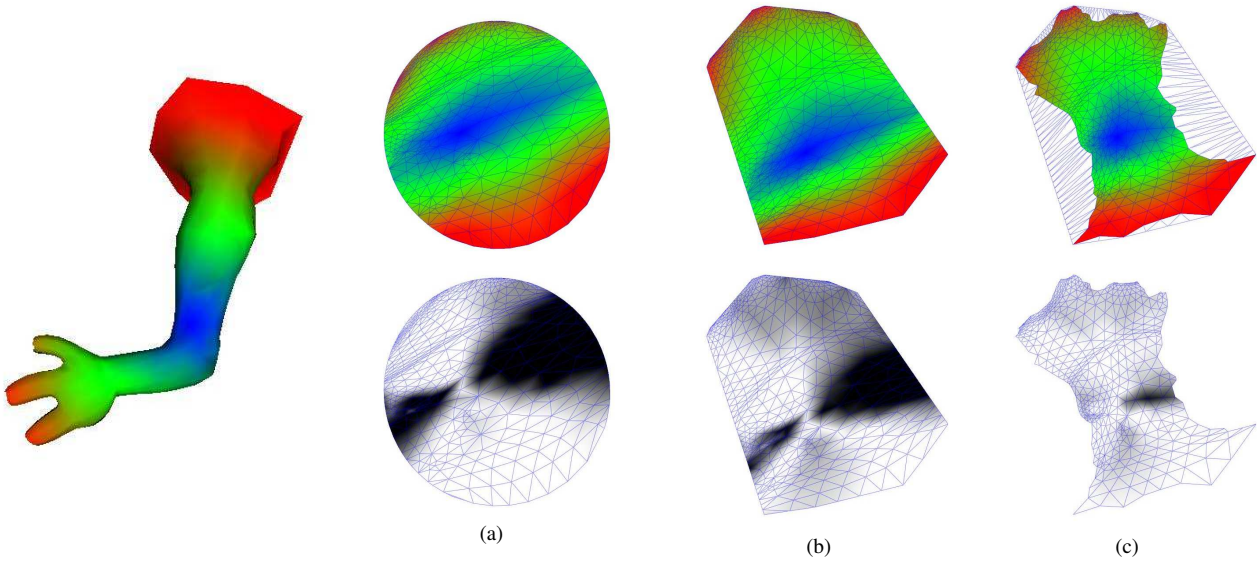


Figure 3: Examples of seam-trees cuts: (a) for the geodesic neighborhood of Figure 2, and (b) and for larger geodesic neighborhood defined in (c). The color represents the geodesic distance from the center (from blue to red).

2.2 Parametrization

Once the patch has a single boundary, we can order the vertices on this boundary to form a single loop. We place the vertices in order on a 2D circle, whose initial radius is the geodesic neighborhood radius. We calculate the loop length and assign an *importance* measure to each vertex based on the ratio of the lengths of its adjacent boundary edges to the total length. The angle between the



(a) (b) (c)

Figure 4: Different parametrization alternatives: (a) Placing the patch boundary loop on a circle. (b) Pulling the boundary vertices to their true geodesic distance and back to the convex hull. (c) Pulling the boundary vertices to their true geodesic distance and adding filler vertices and triangles. The top row shows a color mapping of the true geodesic distance from the center and the bottom row shows the difference between the true geodesic distance and the Euclidean distance inside the patch, where darker means larger error.

vertices is chosen relative to the importance of each vertex.

After this stage we have several options for defining the boundary of the patch to be parameterized. We can keep the boundary vertices on the initial circle (which is convex) and fill the rest of the patch using a barycentric method. This method is quick and simple but fairly inaccurate in terms of preserving the geodesic distance (Figure 4(a)).

If we want to preserve the shape of the patch as defined by its boundary, we pull towards the center all the vertices whose distance to the center is smaller than the neighborhood-radius, and position them in their true distance. This forms a non-convex shape, which is harder to parameterize. This can be solved in two possible ways. The first alternative is to calculate the convex hull of the shape, and pull the vertices which do not lie on the convex hull back to the hull. This achieves better results than if we parameterize the circle, by conforming more naturally to the patch shape (Figure 4(b)). This solution is useful if we require a convex map (e.g. for mean shift, see Section 4). However, using this method still introduces some errors in the preservation of geodesic distances.

The most accurate method in term of geodesic distance preservation involves computing the convex hull as above, but instead of pulling the vertices back to the hull, add 'filler' triangles and vertices to complete the shape to a convex shape much like in [Lee et al. 2002]. This will create a convex patch where the original boundary vertices are positioned in the true geodesic distance. Hence, by keeping the natural shape of the patch and adding 'filler' vertices we add more flexibility to the patch, and the resulting parametrization preserves geodesic distances much better than the other two solutions (Figure 4(c)).

Now that we have a convex boundary of the patch we calculate the parameterizations of the triangles inside the patch by using simple mean value coordinates [Floater 2003]. In this method each vertex is expressed as a convex combination of its neighbors. This improves over Floater's earlier work [Floater 1995] utilizing the mean value theorem for Harmonic functions. In practice, the method takes into account both the angles around the vertices and the edge lengths. This method produced good results, although other parameterizations techniques may be applied at this stage. Post processing techniques might also be used to relax the parameterizations and spread the geodesic distance error more evenly across the patch.

Figure 4 shows the different alternatives for parameterizing the patch. We measure the error using the difference between geodesic distance from the center vertex (as calculated by the front marching algorithm) and the Euclidean distance from the center vertex (distance calculated between the vertices on the 2D parametrization).

3 Implementation

3.1 Timing

The computation time for parametrization is dependent on the size of the patch (radius of geodesic neighborhood), the complexity of the mesh (number of seams etc.) and more. However, as can be seen in the following table, in most cases the whole process takes a few second to compute. Most of the computation time is taken up by the iterative solution which finds the coordinates of vertices inside the patch. This code was not optimized and furthermore, this part is similar in any parametrization method. This means that the added computation for creating the correct geodesic boundary is very small.

Mesh	# Vertices	Param. Type	# Seams	Seconds
dino-pet	500	circle	4	3.2
dino-pet	500	filled	4	4.4
dino-pet	650	circle	4	4.1
dino-pet	650	filled	4	5.4
dino-pet	800	circle	4	7.0
dino-pet	800	filled	4	9.6
horse	100	circle	0	1.4
horse	100	filled	0	2.0
horse	1000	circle	0	6.7
horse	1000	filled	0	7.5
eight	800	circle	1	4.6
eight	800	filled	1	5.7
eight	1300	circle	1	9.0
eight	1300	filled	1	9.9

3.2 Artificial Seams

In some cases there is a need to add seams even if the patch is homeomorphic to a disk. Such an example can be seen in Figure 5. The center vertex is chosen close to the hoof, where the triangles are very large. In contrast, the triangles along the leg are smaller, and the leg itself is long and thin. When the patch is mapped to a circle, the boundary triangles are stretched and the triangles near the center are condensed. This creates a large error in terms of geodesic distance (Figure 5(d)). The solution is to find a special vertex while building the geodesic neighborhood, and add a seam to the patch (Figure 5(e)).

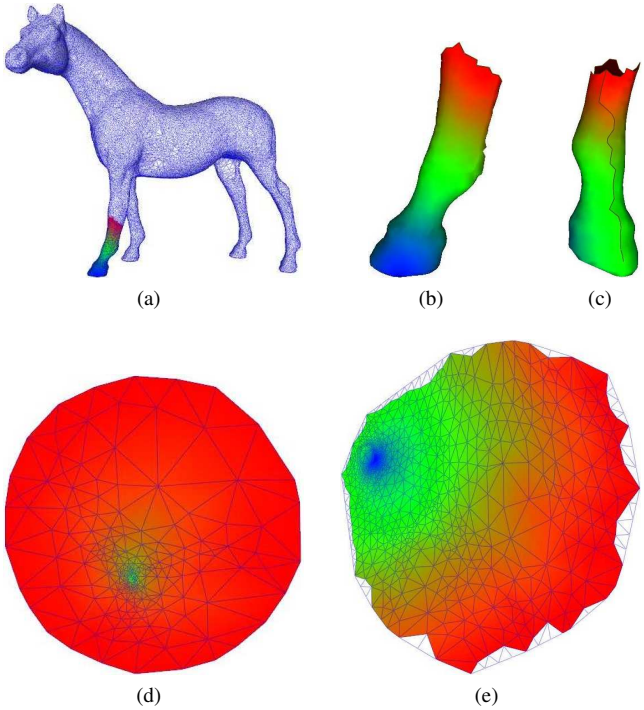


Figure 5: Adding a seam for better parametrization: compare the mapping of geodesic distance without a seam (d), and with a seam (e).

3.3 Map Storage

Performing geodesic parametrization process on all (or a selected number of) vertices in a 2-manifold mesh, creates a collection of local geodesic maps which can be used for a variety of purposes. Nevertheless, since the amount of overlap in these maps can be quite large, there is a need to analyze the storage of these maps carefully, balancing efficiency and speed with storage size and memory requirements.

The first option is to store each local map as a regular mesh, with connectivity and geometry information, and a mapping from the map vertices to the original mesh vertices. This method is efficient, but costly in terms of memory requirement. These requirements depend on the size of the original mesh and the size of the geodesic radius for the maps. If the size of the original mesh is O(n) the required memory for all local maps is roughly $O(n^2)$

Other options alow us to reconstruct the patch using different information. For example, we can save for each local map:

1. The indexes of the vertices in the original mesh which appear in the patch.
2. The distance from the center vertex to each vertex in the patch (from the front marching algorithm)
3. The set of half edges which represent the seam tree
4. The location of each vertex in the patch (original vertices which were split have more than one location)

This storage method saves the space of the connectivity information for the patch, but forces some reconstruction every time we need a map. During reconstruction we must create a mesh object for the patch, use the connectivity information from the original mesh, cut the patch along the seam tree, and then and assign the correct coordinates to each vertex.

In fact, most of the storage space is composed of the geometry information of the patch, i.e. the vertices coordinates. Hence, the last storage method is similar to the previous one without geometry (4). This creates very small memory footprint as it consists only of indices and half edge lists, with no geometry or connectivity. Nevertheless, it forces us to parameterize the actual patch anew every time we need a map resulting in a relatively high processing times to rebuild the patch.

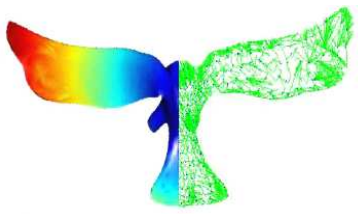


Figure 6: An example of using the mean-shift algorithm for clustering areas on a manifold mesh.

4 Sample Application

There are several possible applications for local geodesic maps such as re-meshing, local texture mapping and more. In this section we concentrate on a specific application where preserving the local geodesic distances from the center is a key constraint.

The mean-shift algorithm is a clustering or filtering algorithm which is widely used on images and video. One step in the mean shift moves a point to the average point of its neighborhood. This is done until convergence to a single point. The main operation in this clustering process is therefore a weighted averaging of the neighborhood of a point in a high-dimensional space of 'features'. The weights for averaging are relative to the distance from the center point. More details can be found in [Comaniciu and Meer 2002].

Recent results [Shamir 2003; Shamir et al. 2004] have adapted the mean-shift method to work on volumetric and manifold meshes (Figure 6). Nevertheless, unlike images or volumes, where the averaging neighborhood is convex, on manifold meshes weighting points on the mesh can easily result in a point laying outside the mesh. There is a need to constrain the movement of the mean-shift on manifold meshes to remain on the mesh. Furthermore, since the averaging is dependent on the distance from the center point, measured on the mesh, there is a need to use geodesic distances between points on the mesh for averaging.

Using local geodesic maps for averaging the geodesic neighborhood of a point solves both these problems for mean-shift on manifolds. Using the convex map, the weighted average of the neighborhood will always fall inside the map, and consequently can be mapped back to the manifold mesh. Furthermore, the map itself preserves distances from the center point as required by the algorithm (see Figure 7).

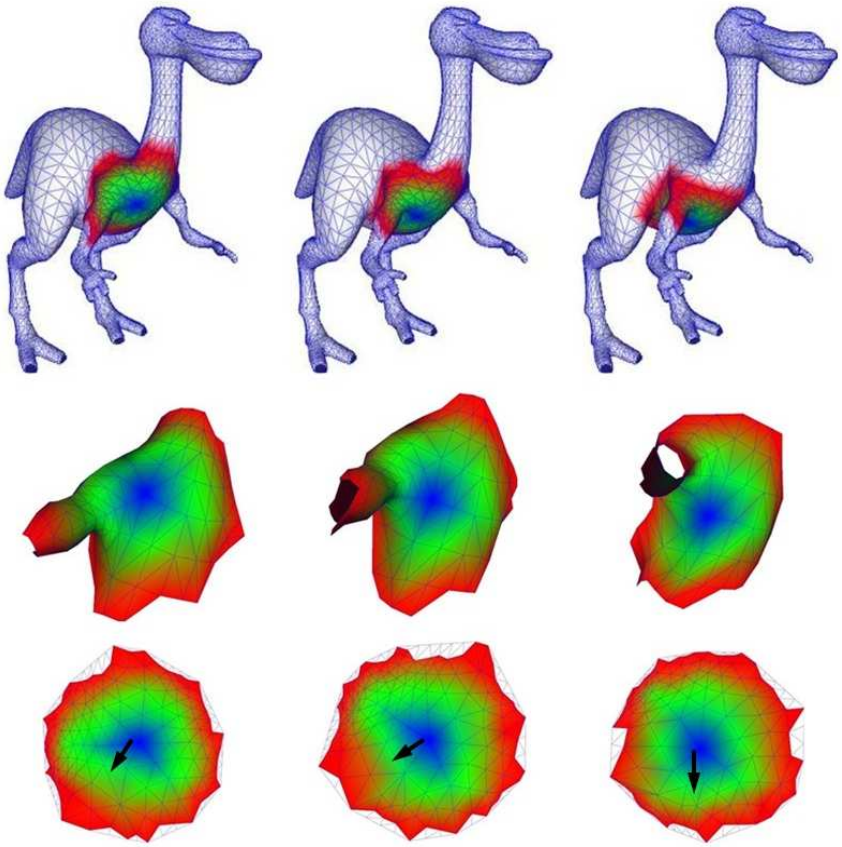


Figure 7: The mean-shift algorithm on a manifold mesh. Each step the geodesic neighborhood is changed (top) and a new patch is parametrized (bottom).

5 Conclusion and Future Work

We have presented a new type of parametrization of manifold meshes which we call *local-geodesic*. This type of parametrization is targeted at preserving geodesic distances from one central point to other points in its geodesic neighborhood. This results in a type of local perspective map similar to the point of view of an Ant living on the manifold.

As an example for successful usage of this type of parametrization we presented the mean-shift process of points on a manifold mesh. In future we would like to pursue other applications and uses for local geodesic parametrization. We are also working on methods to constrain the center point to stay exactly in the middle of the patch and to distribute the error between geodesic and Euclidean distance more evenly over the patch.

References

COMANICIU, D., AND MEER, P. 2002. Mean shift: A robust approach towards feature space analysis. *IEEE Trans. Pattern Analysis and Machine Intelligence 24* (May), 603–619.

ERICKSON, J., AND HAR-PELED, S. 2002. Optimally cutting a surface into a disk. In *Proceedings of the 18th Annual ACM Symposium on Computational Geometry*, 244–253.

FLOATER, M., AND HORMANN, K. Surface parameterization: a tutorial and survey. In *Advances on Multiresolution in Geometric Modelling (to appear)*.

FLOATER, M. 1995. Parametrization and smooth approximation of surface triangulations. *Computer Aided Geometric Design 14*, 231–250.

FLOATER, M. 2003. Mean value coordinates. *Computer Aided Geometric Design 20*, 19–27.

KIMMEL, R., AND SETHIAN, J. 1998. Computing geodesic paths on manifolds. vol. 95, 8431–8435.

LEE, Y., KIM, H., AND LEE, S. 2002. Mesh parameterization with a virtual boundary. In *Computers & Graphics (Special Issue of the 3rd Israel-Korea Binational Conf. on Geometric Modeling and Computer Graphics)*, vol. 26, 677–686.

LEVY, B., AND MALLET, J.-L. 1998. Non-distorted texture mapping for sheared triangulated meshes. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, 343–352.

SHAMIR, A., SHAPIRA, L., COHEN-OR, D., AND GOLDENTHAL, R. 2004. Geodesic mean shift. In *Proceedings of The 5th Korea-Israel Bi-National Conference on Geometric Modeling and Computer Graphics*.

SHAMIR, A. 2003. Feature-space analysis of unstructured meshes. In *Proceedings IEEE Visualization 2003*, 185–192.

SHEFFER, A., AND DE STURLER, E. 2000. Surface parameterization for meshing by triangulation flattening. In *Proceedings of the 9th International Meshing Roundtable*, 161–172.

SHEFFER, A. 2002. Spanning tree seams for reducing parameterization distortion of triangulated surfaces. In *Proceedings of the International Conference on Shape Modeling and Applications 2002 (SMI'02)*, 61–66.

WEISSTEIN, E. W., 2004. Manifold. From MathWorld–A Wolfram Web Resource. http://mathworld.wolfram.com/Manifold.html.

ZIGELMAN, G., KIMMEL, R., AND KIRYATI, N. 2002. Texture mapping using surface flattening via multi-dimensional scaling. *IEEE Transactions on Visualization and Computer Graphics 8*, 2 (April), 198–207.