Eurographics Symposium on Point-Based Graphics (2004)
M. Alexa, M. Gross, H. Pfister, S. Rusinkiewicz (Editors)

Meshing Point Clouds Using Spherical Parameterization

M. Zwicker † and C. Gotsman[2‡]

Massachusetts Institute of Technology
[2] Harvard University

Abstract

We present a simple method for meshing a 3D point cloud to a manifold genus-0 mesh. Our approach is based on recent methods for spherical embedding of planar graphs, where we use instead a k-nearest neighborhood graph of the point cloud. Our approach proceeds in two steps: We first embed the neighborhood graph on a sphere using an iterative procedure, minimizing the tangential Laplacian. Then we triangulate the embedded points and apply the resulting mesh connectivity to the input points. Besides meshing, spherical embedding of point clouds may also be used for other applications such as texture mapping or morphing.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling

1. Introduction

Contemporary 3D scanners produce sets of 3D data points, sampled from the surface of a 3D object. These points are frequently unorganized, and to use them in 3D applications requires computing a polygon (usually triangular) mesh which best approximates the sampled surface. This means associating a connectivity structure with the point set.

Many algorithms exist for meshing 3D point clouds (e.g. [ACK01, FR01, HR02]). The algorithms differ in technique, assumptions made on the input, complexity, robustness and reproducability. By "reproducability", we mean that the algorithm, when fed as input the vertex set of a given 3D mesh, will usually not reproduce the original connectivity and not even the topology of the original 3D mesh. In fact, only a few of the algorithms, most notably the Crust family of algorithms [ACK01], provide conditions on the sampling density which guarantee that the output will be close to the original.

In this paper we make use of recent results on spherical embedding to mesh a 3D point cloud which has been sampled from a closed manifold genus-0 surface, i.e., a topological sphere. This generalizes a method proposed by Floater and Reimers [FR01] for meshes with disk topologies. The advantage of this method is its simplicity and robustness. It guarantees that the resulting mesh will always be closed manifold genus-0.

2. Previous Work

Floater and Reimers [FR01] proposed to mesh a 3D point set sampled from a manifold surface with disk topology and single boundary loop as shown in Figure 1:

Function $M = \text{Mesh}(\text{point set } V)$

1. Construct a graph $G = \langle V, E \rangle$ by connecting each vertex to its k nearest neighbors (k is a user parameter) in Euclidean space.
2. Determine which sequence of vertices will form the boundary B of M.
3. Embed G in the plane such that B forms a convex shape and $V - B$ are positioned at convex combinations of their neighbors. Call the planar point set V'.
4. Form $G' = \langle V', E' \rangle$ by triangulating V' (e.g. by Delaunay triangulation).
5. Use E' to form $M = \langle V, E' \rangle$. The triangles are the faces of the mesh.

Figure 1: *The Floater-Reimers algorithm to mesh a disk-like point set.*

† matthias@graphics.csail.mit.edu
‡ gotsman@eecs.harvard.edu

This procedure works reasonably well for surfaces with a boundary, the results depending on the specific recipe chosen for the convex combination weights. However, even a sophisticated choice of the weights (e.g. shape-preserving [Flo97], harmonic [PP93], mean-value [Flo03]) will result in a significant metric distortion of the 2D embedding relative to the original 3D mesh geometry if the mesh contains significantly curved regions. The results are also affected by the specific choice of the vertices forming the mesh boundary, which is usually heuristic, and the convex shape which the boundary is mapped to in the embedding procedure. Obviously, if the 3D object has the topology of a sphere, this method is not the most natural choice. In a follow-up paper to [FR01] , Hormann and Reimers [HR02] show how to mesh a spherical point cloud by segmenting it to a number of disk-like subsets. This results in a somewhat complicated algorithm.

Recently, researchers realized that for many mesh processing operations it is more natural to parameterize a closed manifold genus-0 mesh to a sphere, rather than cutting it in various ways to reduce it to the case of a disk. The parameterization operation is actually an embedding. However, embedding a graph on the sphere is much more difficult than embedding in the plane, especially since the latter may be done using essentially linear methods, and spherical embedding seems to be essentially non-linear. Parameterizing a given closed manifold genus-0 3D mesh involves embedding the 3D vertices on the sphere, such that the spherical polygons induced by the mesh connectivity do not overlap. In the meshing application, where there is no given connectivity to respect, the only requirement is that the distribution of the points over the sphere is "similar" to their distribution in space. Essentially we would like the metric distortion to be minimal, meaning that short-range distances should be preserved as much as possible. Hence spherical embedding may be viewed as a "graph-drawing" operation in our context, where the edge lengths are to be preserved as much as possible.

Methods to embed a planar graph on the sphere were proposed in [Ale00, DG97, GY02, KVLS, ST98]. These require a triangular graph as input. A simpler scheme was proposed by Gu and Yau [GY02], and by Gotsman et al. [GGS03]. This method is a generalization of the convex combination method of Floater [Flo97] for planar graph embedding, but is non-linear. Essentially it means that instead of each vertex being located at some convex combination of its neighbor's locations in the plane, the vector difference between this convex combination and the vertex location on the sphere has only a radial component. This method has the physical interpretation of a spring system (a zero-length spring corresponding to each edge). The vertices are relaxed to their minimal energy state, subject to the constraint that they are all located on the sphere. Computing the embedding involves solving a system of bilinear equations, which Gu and Yau suggest to do iteratively using the Laplace-Beltrami operator. This somewhat slow procedure usually converges to a local minimum of the spring energy, which Gotsman et al. speculate form a four to six dimensional subspace.

3. Meshing using Spherical Embedding

We propose to use spherical embedding to mesh a point cloud which is known to have spherical topology. This involves eliminating step 2, replacing step 3 of the Floater-Reimers Algorithm (Figure 1) with a spherical embedding routine, and step 4 by a spherical triangulation routine. The result will, by definition, be a closed triangular genus-0 manifold mesh. The modified algorithm for meshing spherical point clouds is shown in Figure 2.

Function $M = \text{Mesh}(\text{point set } V)$
1. Construct a graph $G = \langle V, E \rangle$ by connecting each vertex to its k nearest neighbors (k is a user parameter) in Euclidean space.
2. Embed G on the sphere. Call the spherical point set V'.
3. Form $G' = \langle V', E' \rangle$ by triangulating V' (e.g. by spherical Delaunay triangulation).
4. Use E' to form $M = \langle V, E' \rangle$.

Figure 2: *Meshing a spherical point set.*

The key step is the second one: spherical embedding. Although the graph G is not planar, it may be embedded on the sphere using the same technique as proposed by [GGS03] and [GY02].

We now briefly describe our algorithm for spherical embedding, but refer the reader to [GGS03] and [GY02] for more details and theoretical insights. Let us denote the positions of the input points in the set V by p_i and the points embedded on the sphere by u_i. Gu and Yauand [GY02] proposed to embed a graph on a curved surface (in particular, the unit sphere) using the Laplace-Beltrami operator, which is basically the tangential component of the Laplace operator. Discrete approximations of the Laplace operator at a point u_i have the form

$$L_i = \sum_{j \in N_i} w_{ij} \left(u_i - u_j \right), \tag{1}$$

where N_i is the index set of the neighbors of point i. In our case, these neighbors are given by the k-nearest neighbor graph of the input points p_i. Typically, the weights w_{ij} are chosen to be strictly positive. Popular options are unit weights, i.e., $w_{ij} = 1$ for all $j \in N_i$, or inverse edge length weights, i.e., $w_{ij} = 1/\|p_i - p_j\|$ for all $j \in N_i$. Other choices are discussed in [Flo97, Flo03, FR01].

On the unit sphere, the Laplace-Beltrami operator is simply approximated by the tangential component of the discrete Laplacian:

$$L_i^p = L_i - (L_i \cdot N_i) N_i, \tag{2}$$

where $N_i = u_i/\|u_i\|$ is the unit normal of point u_i. Similar as for planar embeddings [Flo97, Flo03, FR01], Gu and Yau [GY02] and Gotsman et al. [GGS03] showed that embeddings on the sphere can be obtained by solving

$$L_i^p = 0 \quad \text{for all} \quad i. \tag{3}$$

This is a non-linear system of equations that we solve using a simple iterative procedure shown in Figure 3, similar as proposed by [Ale00, GY02, KVLS].

Function $U = \text{SphericalEmbedding}(\text{point set } V)$
1. Initialize $u_i = p_i$ for all i
2. Translate the points u_i such that $\sum u_i = 0$
3. Project the u_i onto the unit sphere
4. For all i
   - $u_i = u_i + \lambda L_i^p$
   - $u_i = u_i/\|u_i\|$
5. Iterate from step 2. until convergence

Figure 3: *Embedding a point set on the unit sphere.*

Note that λ is a damping coefficient that we set to $\lambda = 0.5$ in all our experiments.

3.1. Algorithm Parameters

Our meshing algorithm has a number of parameters: the number of neighbors k used to form the embedded graph connectivity, the convex combination weight recipe used to weight these edges, and the spherical triangulation method.

The average number of neighbors (valence) in a spherical triangle mesh is six, so the value of k should be at least this, else we risk the resulting graph not being well-connected. This can lead to degenerate solutions of the spherical embedding iteration, i.e., all points collapse to the same location. For uniformly sampled surfaces, values bigger than nine usually work fine. To compute all the results shown below, we used a more conservative value of $k = 25$.

We used weights which are either uniform (i.e. independent of the input geometry), proportional to the inverse edge lengths, or adaptive weights similar to those proposed by Yoshizawa et al. [YBS04]. Surprisingly, there did not seem to be much difference in results when uniform and geometry-dependent weights were used. This is probably because the k nearest neighbors are more or less at the same distance from the vertex, hence distance-dependent weights will be close to uniform.

We were able to achieve significantly different results with error adaptive weights [YBS04]. In this procedure, the weights do not only depend on the geometry (points p_i), but also on the current embedding (points u_i). Hence, we need to update the weights after each iteration over all the points in the spherical embedding algorithm (Figure 3). Inspired by [YBS04], we compute adaptive weights as

$$w_{ij} = \|u_i - u_j\|/\|p_i - p_j\| \quad \text{for all} \quad j \in N - i. \tag{4}$$

As shown by Yoshizawa et al. this minimizes the *stretch* or *distortion* of the embedding.

To triangulate the points embedded on the sphere, we used the convex hull routine *qconvex* implemented in the qhull package [qhu]. The resulting spherical triangulation is a Delaunay triangulation with relatively short edge lengths.

4. Experimental Results

We have run our meshing algorithm on a variety of 3D point datasets, three of which are shown in Figure 4: Tweety (19,818 points), Max (52,809 points), and Igea (134,345 points).

In Figure 5, we show triangulations of the Max and Igea models, and the corresponding embeddings on the sphere. We used adaptive weights and $k = 25$ neighbors.

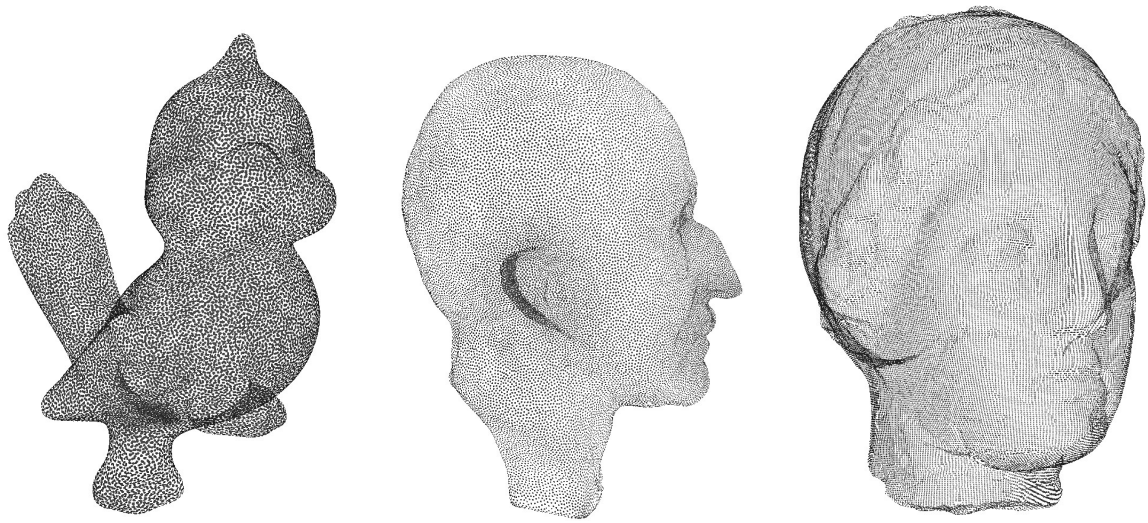
Figure 8 shows results for different choices of the weights. Figure 8(a) was generated using uniform weights, (b) using inverse edge length weights, and (c) using adaptive weights. The close-ups in Figure 8(d,e,f) show the differences in the resulting meshes. The large distortion produced by uniform and inverse edge length weights leads to very thin triangles, while adaptive weights produce a more natural triangulation. Figure 8(g,h,i) show the corresponding spherical embeddings. Clearly, uniform weights and inverse edge length weights lead to much more nonuniform distributions (i.e., high distortion) of the vertices on the sphere than adaptive weights. As proposed by Hormann and Reimers [HR02], standard mesh optimization tools may be applied as a post-process to improve the quality of the final meshes.

4.1. Complexity

Constructing the k-nearest neighbors graph on n points may be done in $O(n\log n)$ [Cla83]. Computing the spherical embedding iteratively seems to run in $O(n^2)$. Computing the 3D convex hull requires $O(n\log n)$ [Ski97], so the entire algorithm complexity is no more than $O(n^2)$.

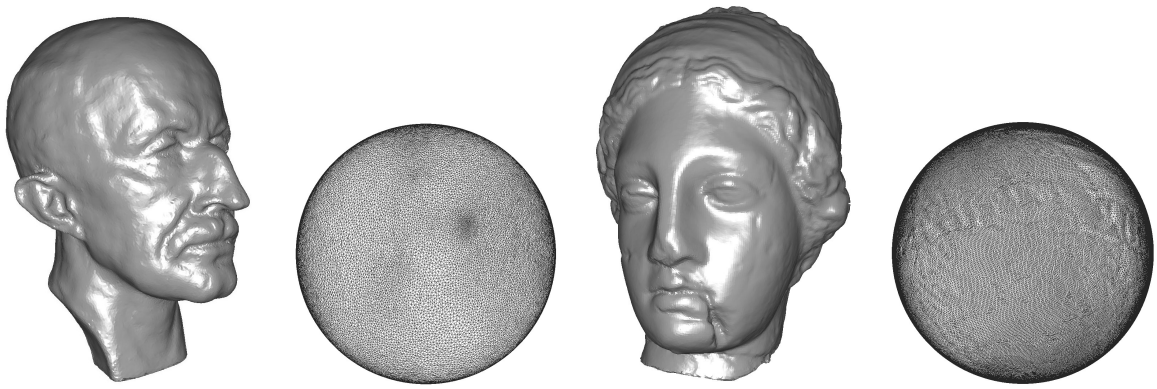
Our experimental results suggest that the spherical embedding procedure does not converge to a vanishing value of the Laplace-Beltrami operator, probably because the embedded graph is not planar. In practice the iterative procedure is stopped when it reaches a local minimum. The convergence of the iteration with the Tweety model for different values of k is illustrated in Figure 6. Here we plot the average length of the tangential Laplacian over the number of iterations. We suspect that the residual error is larger for larger values of k because the corresponding graph is "less planar".

The residual error is also different for the various choices



(a) (b) (c)

Figure 4: *Sample input point clouds: (a) Tweety (19,818 points), (b) Max (52,809 points), and (c) Igea (134,345 points).*



(a) (b) (c) (d)

Figure 5: *(a) Triangular mesh of the Max model, (b) Embedding of the Max model on the sphere, (c) Triangular mesh of the Igea model, (d) Embedding of the Igea model on the sphere.*

of the weights. As shown in Figure 7, uniform weights lead to a smaller error than inverse edge length and adaptive weigths. However, adaptive weights converge faster, which is further illustrated in Figure 9. This figure shows how the tail of the Tweety, which generates a fold-over in the initial projection, is unfolded during the iteration. Adaptive weights lead to a faster unfolding because they do not only depend on the geometry, but also on the distortion produced by the embedding [YBS04].

The current runtimes for the spherical embedding on a state-of-the-art PC (2.8 GHz P4 processor with 1 GB RAM) are reported in Table 1. For all three models, we performed 1600 iterations for these measurements.

Model	Number of Points	Runtime (sec)
Tweety	19,818	125
Max	52,809	424
Igea	134,345	843

Table 1: *Runtime of the spherical embedding iteration (1600 iteration steps) for different models.*

Preliminary results suggest that the spherical embedding procedure can be accelerated significantly using hierarchical algebraic multigrid methods, similarly to Aksoylu et al. [AKS04]. Triangulation of the convex hull of the embed-

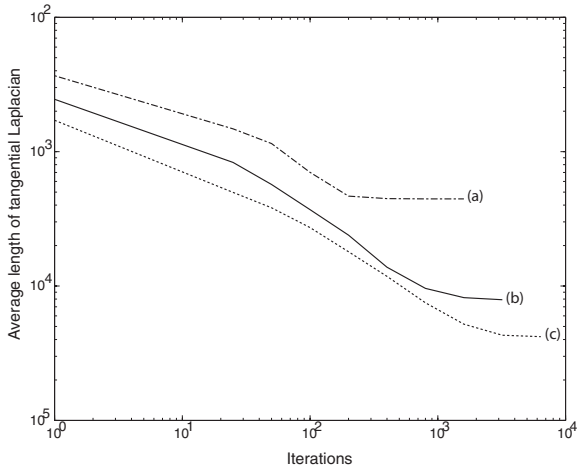


Figure 6: *Convergence of the embedding iteration with uniform weights for different parameters k: (a) k=100, (b) k=25, (c) k=9.*

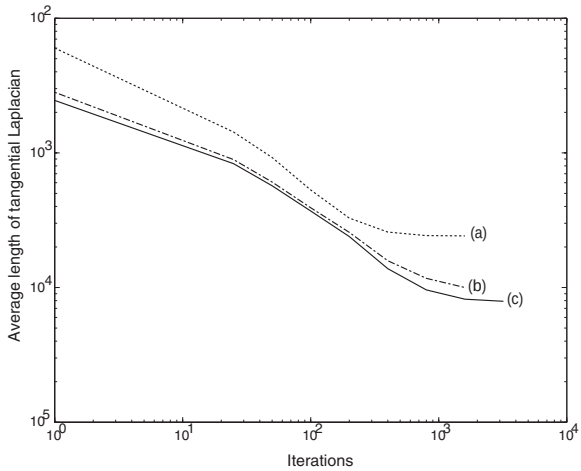


Figure 7: *Convergence of the embedding iteration for $k = 25$ and different choices of the weights: (a) adaptive weights, (b) inverse edge length weights, (c) uniform weights.*

ded points using qconvex [qhu] took less than five seconds for all models.

4.2. Sensitivity to Noise

The algorithm in its basic form will always produce a closed manifold genus-0 mesh on the entire input data set, which is a major advantage of this particular algorithm. In the event of noisy input, however, this mesh may intersect itself. Note that this does not contradict the fact that the mesh is manifold. It can be rectified by a post-processing smoothing stage. It would also be desirable to eliminate outliers in the input. This could be done in a pre-process where points too far from the average of their neighbors are removed.

We tested our algorithm on a noisy point cloud that was obtained by randomly perturbing the points of a smooth model along their normal direction. We chose a range of perturbation b relative to the average distance to the $k = 25$ nearest neighbors. Figure 10 shows a mesh of the Max data set with $b = 1$. We also applied smoothing after meshing to improve the quality of the result, as shown in Figure 10(c). Figure 10(d,e,f) shows close-ups of results for $b = 0$, $b = 1$, and $b = 3$.

5. Conclusion and Discussion

We have described an algorithm for spherical meshing of a 3D point cloud which is simple and efficient. As its main advantage, it guarantees a closed manifold genus-0 result, even for very noisy inputs. As with every other meshing algorithm, a number of independent pre-processing techniques may be applied to the input point set and post-processing techniques to the output mesh.

Embedding the point set on the sphere has an important side-effect: the coordinates of the embedded points on the sphere may be used as spherical texture coordinates, or converted to planar texture coordinates in a variety of standard ways, to allow texture mapping onto the model. The embedding, when viewed as a parameterization, can also be used to establish a one-to-one correspondence between different objects, which is useful for applications such as morphing. Further, the spherical parameterization could be used to reconstruct a continuous surface similar as described by Zwicker et al. [ZPKG] for points parameterized to a planar domain.

In our experiments we observed that we do not obtain spherical embeddings with completely vanishing tangential Laplacians. While it has been proven that it is possible to embed planar 3-connected graphs on the sphere with vanishing Laplacians [GGS03], we do not know of any result for non-planar k-nearest neighbor graphs. An in-depth analysis of this problem seems interesting. In the future, we would also like to investigate hierarchical techniques such as algebraic multigrid or geometric clustering (as in [ZPKG] for planar embedding) to accelerate the spherical embedding iteration. We also plan to experiment with a graph given by local Delaunay neighborhoods as proposed by Floater [FR01] instead of using k-nearest neighbors. This might improve the convergence of the embedding procedure.

Acknowledgements

The work of Craig Gotsman was partially supported by Israel Ministry of Science grant 01-01-01509, German-Israel Fund (GIF) Grant I-627-45.6/1999, and European FP6 NoE grant 506766 (AIM@SHAPE). Matthias Zwicker is partially supported by a stipend of the Swiss National Science Foundation.

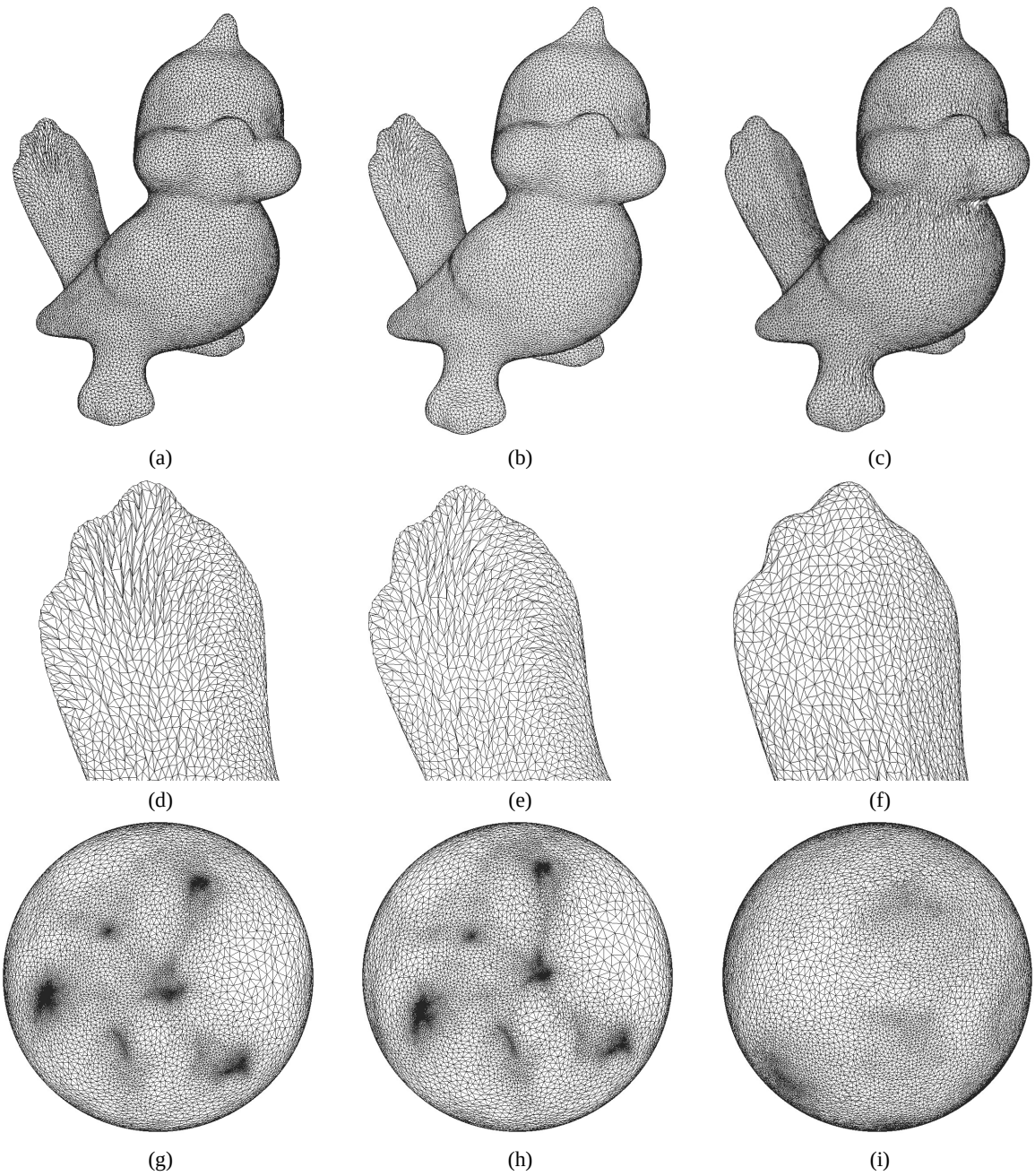


Figure 8: *Sample results using $k = 25$ neighbors in the graph: (a,d) use uniform weights, (b,e) inverse edge length weights, and (c,f) were computed using the adaptive weights; (g,h,i) depict the corresponding embeddings on the sphere.*

References

[ACK01] AMENTA N., CHOI S., KOLLURI R.: The power crust. In *Proceedings of 6th ACM Sympo-sium on Solid Modeling* (2001), pp. 249–260.

[AKS04] AKSOYLU B., KHODAKOVSKY A., SCHRÖDER P.: Multilevel solvers for unstructured surface meshes. *To appear in SIAM Journal of Scientific Computing* (2004).

[Ale00] ALEXA M.: Merging polyhedral shapes with scattered features. *The Visual Computer 16*, 1

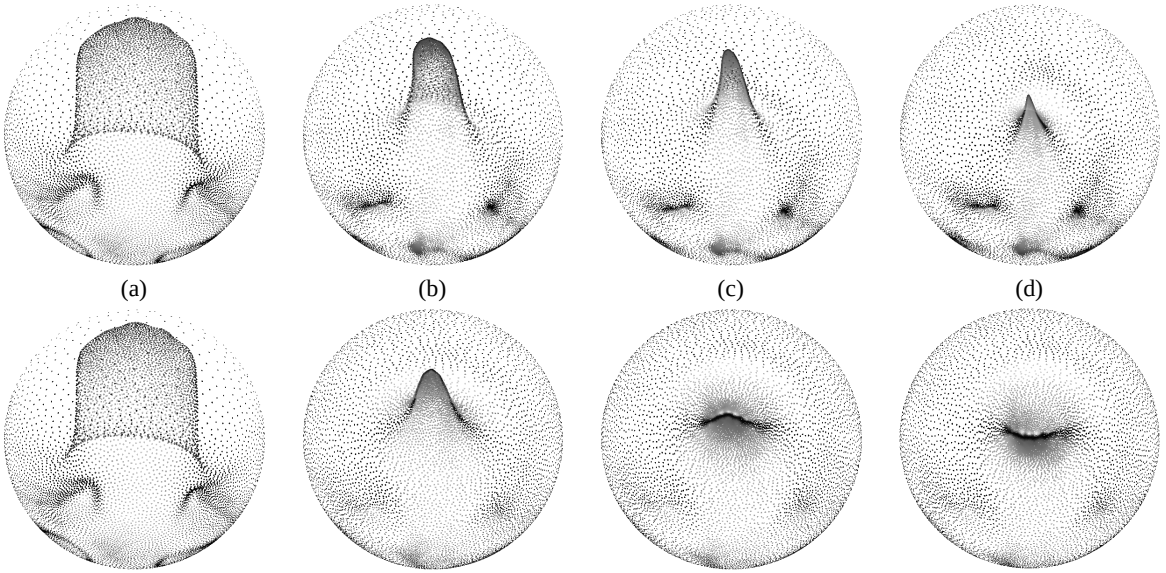


Figure 9: *Unfolding of the tail of the tweety model. (a-d) uniform weights, (e-h) adaptive weights, (a,e) initial embedding, (b,f) after 100 iterations, (c,g) after 200 iterations, (d,h) after 800 iterations.*

(2000), 26–37.

[Cla83] CLARKSON K.: Fast algorithms for the all nearest neighbors problem. In *Proceedings of 24th IEEE FOCS* (1983), pp. 226–232.

[DG97] DAS G., GOODRICH M.: On the complexity of optimization problems for 3-dimensional convex polyhedra and decision trees. *Computational Geometry 8* (1997), 123–137.

[Flo97] FLOATER M.: Parameterization and smooth approximation of surface triangulations. *Computer Aided Geometric Design 14* (1997), 231–250.

[Flo03] FLOATER M. S.: Mean-value coordinates. *Computer Aided Geometric Design 20* (2003), 19–27.

[FR01] FLOATER M. S., REIMERS M.: Meshless parameterization and surface reconstruction. *Computer Aided Geometric Design 18* (2001), 77–92.

[GGS03] GOTSMAN C., GU X., SHEFFER A.: Fundamentals of spherical parameterization for 3d meshes. In *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2003)* (2003), pp. 358–363.

[GY02] GU X., YAU S.-T.: Computing conformal structures of surfaces. *Communications in Information and Systems 2*, 2 (2002), 121–146.

[HR02] HORMANN K., REIMERS M.: Triangulating point clouds with spherical topology. In *Proceedings of Curve and Surface Design* (2002), pp. 215–224.

[KVLS] KOBBELT L., VORSATZ J., LABISK U., SEIDEL H.-P.: A shrink-wrapping approach to remeshing polygonal surfaces. In *Proceedings of Eurographics 1999*, pp. 119–130.

[PP93] PINKALL U., POLTHIER K.: Computing discrete minimal surfaces and their conjugates. *Experimental Mathematics 2* (1993), 15–36.

[qhu] : *www.qhull.org*.

[Ski97] SKIENA S.: *The algorithm design manual*. Telos, 1997.

[ST98] SHAPIRO A., TAL A.: Polygon realization for shape transformation. *The Visual Computer 14*, 8-9 (1998), 429–444.

[YBS04] YOSHIZAWA S., BELYAEV A., SEIDEL H.-P.: A fast and simple stretch-minimizing mesh parameterization. In *Proceedings of SMI'04, to appear* (2004).

[ZPKG] ZWICKER M., PAULY M., KNOLL T., GROSS M.: Pointshop3d: An interactive system for point-based surface editing. In *Proceedings of SIGGRAPH 2002*, pp. 322–329.

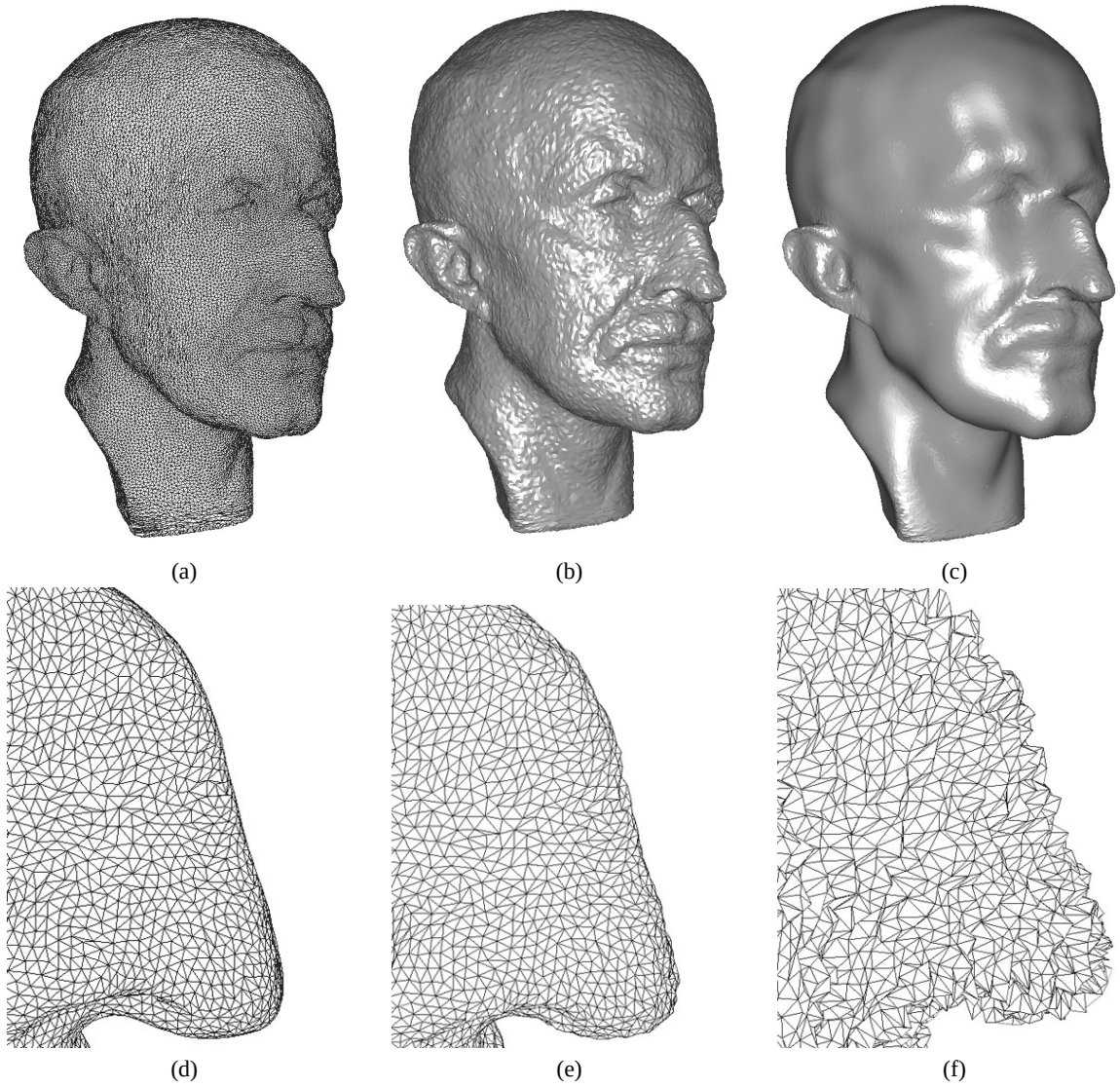


Figure 10: *Triangular mesh of a noisy point cloud with $b = 1$: (a) Mesh, (b) Flat shaded version of (a), and (c) flat shaded mesh after smoothing. Close-ups of some results on inputs with different amounts of noise: (d) original with $b = 0$, (e) $b = 1$, and (f) $b = 3$. We used $k = 25$ and adaptive weights. While the entire mesh is guaranteed to be manifold genus-0, self-intersections may occur.*