Practical Spherical Embedding of Manifold Triangle Meshes

Shadi Saba
Technion
shadi.saba@intel.com

Irad Yavneh
Technion
irad@cs.technion.ac.il

Craig Gotsman
Harvard University
gotsman@eecs.harvard.edu

Alla Sheffer
UBC
sheffa@cs.ubc.ca

Abstract

*Gotsman et al. (SIGGRAPH 2003) presented the first method to generate a **provably** bijective parameterization of a closed genus-0 manifold mesh to the unit sphere. This involves the solution of a large system of non-linear equations. However, they did not show how to solve these equations efficiently, so, while theoretically sound, the method has remained impractical till now. We show why simple iterative methods to solve the equations are bound to fail, and provide an efficient numerical scheme that succeeds. Our method uses a number of optimization methods combined with an algebraic multigrid technique. With these, we are able to spherically parameterize meshes containing up to a hundred thousand vertices in a matter of minutes.*

1. Introduction

Parameterization of a closed manifold mesh with genus 0 should preferably be done on its natural domain: the unit sphere S^2. This provides a good starting point for various mesh processing algorithms such as remeshing, filtering, compression, texture mapping, and morphing. Parameterizing a triangle mesh onto the sphere means assigning positions on the sphere for each of the mesh vertices. The resulting piecewise mapping of the planar faces of the mesh to the corresponding spherical triangles must be bijective. Namely, the spherical triangles, which are the images of the mesh triangles, must form a partition of the sphere. In mathematical terms, this is called an *embedding* of the mesh on the sphere. It is also desirable that the spherical triangles reflect the shapes of the mesh triangles as much as possible, i.e., the parameterization distortion should be minimal in some sense.

A number of papers have addressed the problem of generating spherical parameterizations [1,2,6,11,12,13, 14,18,23,26,27]. The earlier papers [1,12,13,18] attempted to generalize the method of barycentric coordinates for planar parameterization of 3D meshes with disk topology, due to Tutte [28] and Floater [8]. This is an attractive approach since it is easy to control the properties of the resulting parameterization through the choice of barycentric coordinates. In the planar case, the barycentric methods are guaranteed to generate bijective parameterizations, so the hope is that this guarantee will also extend to the spherical case. Unfortunately, the extension to the sphere is not straightforward. Hence the early methods [1,12,14,18] and their extensions [13] did not guarantee that the result is bijective. Several recent spherical parameterization methods employed completely different approaches, but these are either difficult to control [6,26], or quite slow [23,27]. Most recently, Gotsman et al. [11] showed how to correctly generalize the method of barycentric coordinates, with all its advantages, to the sphere. The generalization is based on results from spectral graph theory [4] due to Colin de Verdiere [5] and the extensions of Lovasz [20] and Lovasz and Schrijver [21]. It involves assigning (symmetric) positive weights w_{ij} to each edge of the mesh and solving the following set of $4n$ non-linear equations in $4n$ unknowns for the embedding coordinates $x_i=(u_i,v_i,w_i)$ and auxiliary variables α_i:

$$\alpha_i x_i = \sum_{j\in N(i)} w_{ij} x_j \qquad i=1,..,n \tag{1}$$

$$\|x_i\|^2 = 1 \qquad i=1,..,n, \tag{2}$$

where $N(i)$ are the neighbors of the i-th vertex.

The geometric interpretation of these equations is that the difference between each vertex and the appropriate weighted combination of its neighbors, as prescribed by the (positive) w_{ij}, has only a radial component. As such, each vertex is *balanced*.

The algebraic interpretation of these equations is that the three coordinate vectors of the embedding on the sphere are in the nullspace of the following Laplacian-type matrix L associated with the graph:

$$L_{ij} = \begin{cases} -w_{ij} & i \neq j \\ \alpha_i & i = j \end{cases} \tag{3}$$

There is also a physical interpretation of the equations [10]. Assume that the weights correspond to spring constants. Then it is relatively easy to see that the equations in Eq. (1) are those obtained when applying the Lagrange multiplier technique to minimize the sum of the squared weighted edge lengths – the spring energy – subject to the vertices being on the sphere:

$$x = \arg\min \sum_{i=i}^{n} \sum_{j\in N(i)} w_{ij} \|x_i - x_j\|^2 \qquad s.t.\ \|x_i\| = 1 \tag{4}$$

Unfortunately, solving Eqs. (1) and (2) is *not sufficient* to guarantee a bijective embedding. It will be sufficient

only if a number of additional conditions on the spectrum of L are satisfied. The spectral theory [20,21] maintains that the spectrum should contain exactly one negative eigenvalue and exactly three null eigenvalues corresponding to the three coordinate functions of the embedding. The rest of the eigenvalues should be positive. In this case, L is called a *Colin de Verdiere* matrix [5] for the mesh connectivity. A solution to Eqs. (1) and (2) guarantees only partial satisfaction of these conditions – namely that the three coordinate function are contained in L's nullspace. There exists an entire family of trivial solutions to the equations, which represent non-bijective parameterizations. An obvious trivial solution is when all vertices coincide at any one point on the sphere. Here the spring energy of (4) achieves the global minimum of 0. This corresponds to the case when the co-rank of L is one. A somewhat less obvious situation is when all the vertices lie on one great circle on the sphere. This corresponds to the case when the co-rank of L is two. Another more surprising set of non-bijective solutions is when the spherical triangles "wrap" the sphere more than once. This corresponds to the case where the spectrum of L contains more than a single negative eigenvalue. Hence a major concern is to "steer" the solver toward the desirable bijective solutions.

Another more general problem is that there exist triangular connectivities for which no bijective parameterizations even exist. This is the family of non-inscribable planar triangle graphs [7]. These correspond to the class of planar triangle graphs that are not Delaunay-realizable, meaning that they cannot be drawn as straight line graphs in the plane, which are also Delaunay triangulations. This implies that for any assignment of positive weights to the mesh edges, there will be no non-trivial solution to the equations. This, however, is very rare. In most cases, not only is there a bijective solution to the equations, but an entire family of solutions. Beyond the obvious two degrees of freedom due to arbitrary rotations on the sphere, there also exist some non-trivial transformations which are invariant to the equations. See [11] for a simple example.

The system of Eqs. (1) and (2) is sparse, having the structure of the graph adjacency matrix. Nonetheless, despite this simple structure, solving these quadratic equations is non-trivial. Gotsman et al. [11] did not even attempt to provide an efficient method to solve the equations. Instead, they simply employed a standard MATLAB solver, which did not take any advantage of the system's structure. This solver was only able to parameterize meshes with up to 2,000 vertices, and for these it ran for several minutes. Embedding significantly larger meshes was out of the question. Since practical applications tend to involve meshes of tens to hundreds of thousands of vertices, this system of equations remained unsolvable.

The early methods for spherical parameterization based on barycentric coordinates [1,12,18] employ simple Gauss-Seidel-type iterative schemes. Although not mentioned in the papers, it seems that the authors were trying to solve systems similar to Eqs. (1) and (2). Without exception, these simple Gauss-Seidel solvers ultimately collapse to a trivial solution. Some "tricks" have been employed to avoid this collapse, but these eventually prevent the system from converging to a true solution of the equations. Similar behavior occurs in a recent scheme described in [13]. In Section 2, we outline the prototype of these inadequate approaches and prove that they are bound to fail.

This paper proposes an efficient numerical method to generate spherical embeddings by generating non-trivial solutions to Eqs. (1) and (2). In Section 3, we describe our approach, which breaks the solution down into a two-step procedure involving the solution of two systems of equations, one linear and one non-linear. The linear system is solved using a multiresolution algebraic multigrid approach. The solution to this system is used as an initial guess for solving the nonlinear system. A careful iterative scheme then improves the solution until it converges, avoiding any collapses to the trivial solutions. Experimental results illustrating the quality of our embeddings and the efficiency of our methods are provided in Section 4. We conclude in Section 5.

2. Inadequate Solution Method

The simplest iterative method for solving Eqs. (1) and (2) is by a "projected" Gauss-Seidel iteration with damping parameter $0<\lambda\leq 1$:

Projected Gauss-Seidel($x(0)$, λ)
$k = 0$
repeat
  for i=1 to n do
    $s = (1-\lambda)x_i(k) + \lambda\Sigma_j w_{ij}x_j(k)$
    $x_i(k+1) = s/\|s\|$
  end
  $k = k+1$
until $\|x(k)-x(k-1)\|<\delta$

The physical interpretation of one (undamped) Projected Gauss-Seidel step at vertex i is to minimize the spring energy (4) with respect to the position of just that vertex. Damping means taking only a partial step in the direction of the new location, or, equivalently, a partial step along the sphere in the tangential direction. This scheme has been used by many authors [1,12,13,18]. Unfortunately, the Projected Gauss-Seidel steps decrease the residual for only a finite number of iterations and different values of λ affect only the speed of the process. By reducing λ it is possible to get closer to a bijective solution before it starts to diverge, but this will also cause the approach to the solution to be very slow. As it approaches a bijective solution, the scheme ultimately becomes unstable, the residual increases, and the system collapses to a

degenerate solution. We have observed this in all our experiments.

This undesirable divergence can be explained by examining the linearized form of the equations near a bijective solution. First, note that for a dense mesh embedded near a bijective solution, α_i will tend to a value on the order of $(1-O(d^2))\Sigma_j w_{ij}$, where d is the average length of an edge (which will typically be $O(1/\sqrt{n})$). Furthermore, during the iterations, the values of α_i will change much less rapidly than the values of x_i by a factor of $O(d)$. Hence we may treat the values of the unknown α_i as positive constants at this stage, and the problem reduces to solving the linear system $Lx=0$, where L is the matrix defined in (3). The Projected Gauss-Seidel method may now be recognized as a standard Gauss-Seidel iteration for this equation:

$$x_i(k+1) = -\frac{1}{L_{ii}}\sum_{j=1}^{i-1} L_{ij}x_j(k+1) - \frac{1}{L_{ii}}\sum_{j=i+1}^{n} L_{ij}x_j(k)$$
$$= \frac{1}{\alpha_i}\sum_{j=1}^{i-1} w_{ij}x_j(k+1) + \frac{1}{\alpha_i}\sum_{j=i+1}^{n} w_{ij}x_j(k).$$

The following corollary of a theorem ([15], p. 70) concerning the convergence of a Gauss-Seidel iteration will be useful in analyzing the behavior of this process:

Theorem: If A is a symmetric matrix with a positive diagonal and A is positive definite when its diagonal is multiplied by two, then the damped Gauss-Seidel iteration with parameter $0<\lambda\leq 1$ for solving $Ax=b$ converges if and only if A is positive definite.

Now our matrix L in the vicinity of a bijective solution is symmetric with a positive diagonal. Furthermore, when the diagonal is multiplied by two, the i-th diagonal element approaches $2\Sigma_j w_{ij}$. Since the sum of the absolute values of the rest of the row is $\Sigma_j w_{ij}$, this modified matrix is diagonally dominant, hence positive definite. Thus, the conditions of the theorem hold for L. On the other hand, L is a Colin de Verdiere matrix for the connectivity graph; hence it has at least one negative eigenvalue. Thus L is not positive definite, and the theorem implies that any damped Gauss-Seidel iteration for L will not converge to a bijective solution. However, since the iteration does consistently reduce the spring energy, it will ultimately converge to a trivial solution.

3. Our Solution

Since the Projected Gauss-Seidel is unstable near a bijective solution, we must replace it at that point with a more sophisticated and stable method. In this section we describe how we do this, along with some optimizations that help accelerate convergence. The stages of our method are as follows:

1. Generate a good initial guess for the embedding.
2. Perform the Projected Gauss-Seidel steps until the residue begins to increase.
3. Perform local Newton steps until convergence.

3.1 Generating an initial guess

No matter which iterative method is used, it is always beneficial to start from a good initial "guess" for the solution. The simplest initial guess is to center the mesh at the origin, and then project the vertices to the sphere. However, this will usually introduce "folds" into the embedding, which could be very difficult to eliminate later, and can push the solver in the direction of a non-bijective solution.

Instead we propose to use a variant of a method proposed by Isenburg et al. [16], to generate the initial guess:

InitialGuess(*M,w*)

1. Partition *M* into two balanced sub-meshes.
2. Embed each sub-mesh in a planar disk using the barycentric method with weights *w*.
3. Combine the embeddings of the two sub-meshes into one planar embedding using Moebius inversion.
4. Use inverse stereo projection to obtain a spherical embedding.

We use the MeTiS [17] graph partitioning package to obtain a balanced minimal vertex separator of the mesh graph. This means that we identify a small subset of the vertices, whose removal, along with their incident edges, leaves us with two unconnected components of approximately equal size. The removed vertices are called *separators*. We later add them back to each component, so that both have a common boundary. Because of the celebrated planar separator theorem [19], we can expect the size of the separator to be $O(\sqrt{n})$. It is important that this set not be too small.

Each subgraph is now embedded on the unit disk by fixing its (common) boundary vertices to points sampled uniformly on the unit circle, and solving the planar barycentric equations for the interior vertices:

$$x_i = \sum_{j\in N(i)} w_{ij}x_j \quad i \in \text{interior vertices.} \qquad (5)$$

This is a sparse linear system, which has a unique solution. Large systems with this structure may be solved using an algebraic multigrid (AMG) algorithm, such as the Ruge-Stueben algorithm [24], described in the Appendix.

Given the two sub-meshes embedded in the unit disk with common boundary vertices, we apply the complex Moebius inversion $f(z) = 1/\text{conj}(z)$ to one of the embeddings. This maps the interior of the unit disk to its exterior. The union of both embeddings produces a planar embedding (where the origin is mapped to infinity). Moebius

transformations are conformal; hence in a continuous setting they preserve angles. This is not strictly true in the discrete setting. However, in our experience this transformation closely preserves the shape of the triangles everywhere except at infinity; see Figure 1 (on the last page).

The planar embedding (u,v) is then mapped to the unit sphere using the inverse stereo projection:

$$P(u,v) = \frac{1}{1+u^2+v^2}\left(2u, 2v, 1-u^2-v^2\right).$$

Note that this maps the unit circle (the common boundary of the two sub-meshes) to the equator of the sphere. An alternative approach for obtaining the initial embedding from the two unit disks is to map each unit disk to a corresponding hemisphere. This will avoid the need for special treatment of the poles. Nevertheless, our method is more generic as it supports additional transformations of the planar embedding, if additional effects are desired [25].

Similarly to the Moebius transform, stereo projection is conformal in the continuous setting. In our discrete setting it closely preserves the shape of the mesh triangles, providing a good initial guess for the iterative improvement stage.

Even though each of the two planar embeddings satisfies the planar barycentric equations (5), this does not imply that they satisfy the corresponding equations (1) on the sphere. Furthermore, there is no theoretical guarantee that this initial spherical embedding is bijective, although, in practice, for dense meshes we found that this is typically the case. The hope is that from this initial embedding it will be relatively easy to converge to a bijective solution of the equations.

3.2 Local Gauss-Seidel iteration

As we observed earlier, the Projected Gauss-Seidel iteration does reduce the equation residual, at least at the beginning of the process. So it is beneficial to use it as long as it produces the desired effect.

We use the following variation on the biterative procedure of Section 2:

Projected Gauss-Seidel(x(0), λ)
$k = 0$
$R(0) = 0$
do
  for i=1 to n do
    $s = (1-\lambda)x_i(k) + \lambda\Sigma\, w_{ij}x_j(k)$
    $\alpha_i = ||s||$
    $x_i(k+1) = s/\alpha_i$
  end
  $k = k+1$
  $R(k) = Res(x)$ // L_2 residual
while $R(k) < R(k\text{-}1)$

The residual $Res(x)$ is the L_2 distance of x from the solution, defined as:

$$Res(x) = \frac{1}{n}\sqrt{\sum_{i=1}^{n}\left\|\alpha_i x_i - \sum_{j\in N(i)} w_{ij}x_j\right\|^2},$$

where the weights are normalized to sum to unity per vertex. The main difference between this method and the standard Projected Gauss-Seidel iteration is the termination condition, where we stop the process once the residual starts growing. Because of this, we do not need to be too careful which value of λ is used, so we use the simple λ=1.

3.3 Local valid Newton step

Once the Projected Gauss-Seidel iteration begins to fail, i.e., increases the residual, we switch to a more sophisticated, albeit slower, method: local Newton optimization. This method is guaranteed to reduce the residual. To reduce the number of variables and constraints, we transform the problem to spherical coordinates. Performing a vector product on each side of Eq. (1) with x_i, we are able to eliminate the auxiliary variables α_i from the equations:

$$x_i \times (\Sigma\, w_{ij}x_j) = 0 \qquad i = 1,..,n. \tag{6}$$

Furthermore, it is easy to see that the third component equation of (6) may be derived from the first two because, intuitively, the equation is indifferent to the lengths of the two vectors forming the cross product. Recall that $x_i=(u_i,v_i,w_i)$. Using the following spherical coordinates: $u_i = \cos(\theta_i)\cos(\phi_i)$, $v_i = \sin(\theta_i)\cos(\phi_i)$, $w_i = \sin(\phi_i)$, allows us to eliminate Eq. (2) and reduce Eq. (6) to the following two equations:

$$R_1(i,\theta,\phi) = \cos(\theta_i)\sum_{j\in N(i)} w_{ij}\sin(\theta_j)\cos(\phi_j) - \sin(\theta_i)\sum_{j\in N(i)} w_{ij}\cos(\theta_j)\cos(\phi_j) = 0 \quad i = 1,..,n$$

$$R_2(i,\theta,\phi) = -\sin(\phi_i)\sum_{j\in N(i)} w_{ij}\cos(\theta_j)\cos(\phi_j) - \cos(\theta_i)\cos(\phi_i)\sum_{j\in N(i)} w_{ij}\sin(\phi_j) = 0 \quad i = 1,..,n.$$

We would like to use a local Newton-type iteration to minimize R_1 and R_2 (over the entire mesh) by moving the i-th vertex. With some reuse of notation, the corresponding *residual* functions are defined as:

$$R_1(\theta,\phi) = \sum_{i=1}^{n} R_1^2(i,\theta,\phi)$$

$$R_2(\theta,\phi) = \sum_{i=1}^{n} R_2^2(i,\theta,\phi).$$

Moving vertex i affects the residual only at vertex i and its neighbors. To apply a Newton step, we need to compute the Jacobian matrix of the two residual functions by θ_i and ϕ_i. The relevant non-zero partial derivatives are:

$$\frac{\partial R_1(i,\theta,\phi)}{\partial \theta_i} = -\cos(\theta_i)\sum_{j\in N(i)} w_{ij}\cos(\theta_j)\cos(\phi_j) - \sin(\theta_i)\sum_{j\in N(i)} w_{ij}\sin(\theta_j)\cos(\phi_j)$$

$$\frac{\partial R_1(i,\theta,\phi)}{\partial \phi_i} = 0$$

$$\frac{\partial R_2(i,\theta,\phi)}{\partial \theta_i} = -\sin(\theta_i)\cos(\phi_i)\sum_{j\in N(i)} w_{ij}\sin(\phi_j)$$

$$\frac{\partial R_2(i,\theta,\phi)}{\partial \phi_i} = -\cos(\phi_i)\sum_{j\in N(i)} w_{ij}\cos(\theta_j)\cos(\phi_j) - \cos(\theta_i)\sin(\phi_i)\sum_{j\in N(i)} w_{ij}\sin(\phi_j),$$

and for $j \in N(i)$:

$$\frac{\partial R_1(j,\theta,\phi)}{\partial \theta_i} = w_{ij}\cos(\phi_j)\cos(\theta_j - \theta_i)$$

$$\frac{\partial R_1(j,\theta,\phi)}{\partial \phi_i} = -w_{ij}\sin(\phi_i)\sin(\theta_j - \theta_i)$$

$$\frac{\partial R_2(j,\theta,\phi)}{\partial \theta_i} = w_{ij}\sin(\phi_j)\sin(\theta_i)\cos(\phi_i)$$

$$\frac{\partial R_2(j,\theta,\phi)}{\partial \phi_i} = w_{ij}\sin(\phi_j)\cos(\theta_i)\sin(\phi_i) + w_{ij}\cos(\theta_j)\cos(\phi_j)\cos(\phi_i).$$

The Jacobian of R_1 and R_2 by θ_i and ϕ_i is then obtained by applying the chain rule:

$$\frac{\partial R_1}{\partial \theta_i} = R_1(i)\frac{\partial R_1(i)}{\partial \theta_i} + \sum_{j\in N(i)} R_1(j)\frac{\partial R_1(j)}{\partial \theta_i},$$

and similarly, for $\frac{\partial R_1}{\partial \phi_i}, \frac{\partial R_2}{\partial \theta_i}$ and $\frac{\partial R_2}{\partial \phi_i}$. These are the four components of the 2×2 Jacobian matrix.

Having this, the local Newton iteration is as follows:

Local Newton Iteration

$k = 0$
while $R_1(\theta(k),\phi(k)) + R_2(\theta(k),\phi(k)) > tol$
  for i=1 to n do
    J_i = Jacobian of $R_1(\theta(k),\phi(k))$ and $R_2(\theta(k),\phi(k))$ by θ_i and ϕ_i.

$$\begin{pmatrix}\theta_i(k+1)\\ \phi_i(k+1)\end{pmatrix} = \begin{pmatrix}\theta_i(k)\\ \phi_i(k)\end{pmatrix} - \lambda J_i^{-1}\begin{pmatrix}R_1(\theta(k),\phi(k))\\ R_2(\theta(k),\phi(k))\end{pmatrix}$$

  end
  $k = k+1$
end

A local Newton iteration is applied to each vertex in turn. This involves inverting a 2x2 matrix and applying the iteration. By definition, this iteration will always decrease the residual in the vicinity for some value of $\lambda>0$. However, finding the value of λ for which the decrease is maximal is non-trivial. We find the optimal value by using a *linesearch* technique. In each iteration we start with $\lambda=1$ and repeatedly decrease λ by half if this leads to a larger decrease in residual, while constraining $\lambda>0.05$ in order not to stagnate the convergence.

To better condition the system, when optimizing the i-th vertex we rotate the vertex and its 1-ring neighborhood so that the vertex's normal coincides with the x-axis. This will give us small values of θ_i and ϕ_i.

When the iterations have converged, we will be left with an embedding which is an approximate solution to Eqs. (1) and (2) up to a prescribed tolerance, and is not degenerate. Since the spectral theory also implies that all the triangles of the bijective spherical embedding will have positive areas – this guarantees that a small enough tolerance will always result in a bijective embedding.

4. Experimental Results

We have fully implemented the numerical schemes described in this paper. The resulting software is available on the Web at http://www.cs.technion.ac.il/~shadis.

We experimented with our numerical scheme using various input models and weights. We observed that the Tutte (uniform) weights embedding had the smallest residual before the local Newton improvement phase, and converged the fastest. We did not experience any collapsed or double (wrapped) embeddings.

The figures below demonstrate the results of running our algorithm with different weights and on models of different size. We explored three types of weights: uniform [28], conformal [22], and mean-value [9]. Both conformal and mean-value weights do not strictly satisfy the Colin de Verdiere conditions. Conformal weights can be negative, and mean-value weights are not symmetric. They are nevertheless very commonly used, as they tend to result in angle preserving parameterizations. In practice, on all the models we tested, both types of weights resulted in bijective parameterizations when using our method. To visualize the embeddings we used the parameterization to map the normals from the original models to the spherical versions. Figure 2 (on the last page) compares the parameterization of two irregular meshes (triceratops and gargoyle) using the three types of weights. While there are visible differences between uniform weights embedding and the other two, the conformal and mean-value embeddings are nearly identical. As expected, for meshes with regular geometry such as the torso (Figure 3), uniform and mean-value/conformal weights result in very similar parameterizations. Similarly, mean value (or conformal) parameterization of models that are close in shape to the sphere, such as the head and the skull (Figure 3), produces nearly isometric parameterizations. In contrast, mean value spherical parameterization of models with high curvature

variation, such as the human (Figure 3) or the gargoyle (Figure 2) results in significant stretch.

Table 1 summarizes the runtimes of our algorithm. For all the examples, the algorithm terminated once the residual was less than 10^{-6}. We break down the time into two components, the time it takes to generate the initial guess (Section 3.1) (implemented in MATLAB), and the time it takes to obtain the final result (Sections 3.2-3.3) (implemented in C++). The total runtimes, as measured on a 2.8 GHz PC with 1GB RAM, varied between 10 seconds for models of 5,000 triangles to 1,350 seconds (22 minutes) for a model of 129,000 triangles. As can be seen from the values, the runtime is roughly linear in the size of the input meshes. The time for uniform weights embedding is typically smaller than for conformal or mean value embeddings, which require roughly the same time. For comparison, Praun and Hoppe [23] quote times of 7 to 25 minutes for parameterizing models of 25,000 to 100,000 triangles, although they generate completely different parameterizations.

Two recent methods [2,13] considered variations of the Gauss-Seidel procedure (Section 2), which we showed to be unstable. Gu et al. [13] used a trial and error approach to adjust the damping parameter λ, in order to obtain a value for which the procedure may be terminated sufficiently close to a bijective solution before diverging. Where successful, they quoted times of 530 seconds for parameterizing meshes with 30,000 faces. Birkholz [2] used a hierarchical method that works with the spherical angles of the embedding. His method requires approximately 600 seconds to parameterize meshes with about 100,000 faces. The hierarchical approach stabilizes the conversion, but is able to obtain only an approximate solution.

5. Discussion and Conclusion

We have presented an efficient numerical scheme to solve the non-linear equations described by Gotsman et al. [11], which guarantee a bijective spherical parameterization of a closed manifold genus-0 mesh. This scheme enables us to parameterize meshes containing hundreds of thousands of vertices in a matter of minutes.

We have no theoretical guarantee that our method indeed solves the equations, namely reduces the residual to zero given enough time. Theoretically, the iterations may get stuck in a local minimum. However, we have overwhelming experimental evidence that this is not the case. The good initial guess seems to bring the solver sufficiently close to a bijective solution, and, in all our experiments, we were able to bring the residual below any reasonable tolerance by sufficient iteration.

Our scheme uses a spherical embedding constructed through inverse stereo projection of a planar embedding as an initial guess for its careful iterative scheme. Although this initial guess is not guaranteed to be bijective, the theory guarantees that the final solution will always be bijective (up to the solution tolerance) if the weights are symmetric.

For non-symmetric (or negative) weights, the theory does not guarantee that the solution to the equations will be bijective, even if we start out with an embedding that is. However, in this case, we can extend the algorithm to guarantee that the embedding remains bijective throughout the process. When a new position is computed for any vertex, we check that the bijectivity has not been violated. Should that be the case, the vertex is "backtracked" slowly towards its previous position until it becomes valid again. Consequently, the final result will also be bijective.

Table 1. Performance statistics

Model	# triangles	Weights	Initial Guess (sec)	Solution (sec)
Triceratops	5,660	uniform	2.7	8.15
		conformal	1.3	14.8
		mean-value	2.3	16.1
Gargoyle	20,000	uniform	10.2	39.3
		conformal	4.7	86.8
		mean-value	4.7	162.5
Torso	22,720	Uniform	9.9	43.4
		mean-value	5.2	87.2
Skull	40,000	Uniform	10.8	120.5
		mean-value	10.1	149.8
Bunny	69,664	Uniform	21.3	306.5
		mean-value	17.4	456.0
Human	119,998	Uniform	35.0	878.8
		mean-value	32.0	1,134.3
Head	128,750	Uniform	41.9	1,045.9
		mean-value	36.3	1,307.2

The test for validity of a vertex is straightforward: calculate the normals for all the oriented incident faces. Then the mapping is bijective at that vertex iff these normals form acute angles with the normal to the sphere at the vertex.

The runtime in our current implementation is dominated by the second phase – iterations over the entire mesh on the sphere. We would like to apply a multigrid method to accelerate this phase as well. However, this will be more difficult since the algebraic multigrid is primarily designed for linear systems, and the second phase is inherently non-linear.

6. References

[1] M. Alexa, "Merging Polyhedral Shapes with Scattered Features", *The Visual Computer*, 2000, 16 (1), 26-37.

[2] H. Birkholz, "Shape-preserving Parametrization of Genus 0 Surfaces", *Proc. Winter Conference on Computer Graphics (WSCG)*, 2004.

[3] W.L. Briggs, V.H. Henson and S.F. McCormick, *A Multigrid Tutorial*. SIAM, 2000.

[4] F.R.K. Chung, *Spectral Graph Theory*, *CBMS 92, AMS*, 1997.

[5] Y. Colin de Verdiere, "Sur un Nouvel Invariant des Graphes et un Critere de Planarite", *Journal of Combinatorial Theory B*, 1990, 50, 11-21. [English translation: "On a new graph invariant and a criterion for planarity", in: *Graph Structure Theory* (N. Robertson, P. Seymour, Eds.) Contemporary Mathematics, AMS, pp. 137-147, 1993.]

[6] G. Das and M.T. Goodrich, "On the Complexity of Optimization Problems for 3-dimensional Convex Polyhedra and Decision Trees", *Computational Geometry*, 1997, 8, 123-137.

[7] M. B. Dillencourt and W. D. Smith, "Graph-theoretical Conditions for Inscribability and Delaunay Realizability", *Proc. 6th Canad. Conf. Computational Geometry*, 1994, pp. 287-292.

[8] M.S. Floater, "Parameterization and Smooth Approximation of Surface Triangulations", *Computer Aided Geometric Design*, 1997, 14, 231-250.

[9] M. S. Floater, "Mean Value Coordinates", *Computer Aided Geometric Design*, 2003, 20(1), 19-27.

[10] M.S. Floater, "Some Examples of Barycentric Mappings on Spheres", Private Communication, 2003.

[11] C. Gotsman, X. Gu, and A. Sheffer, "Fundamentals of Spherical Parameterization for 3D Meshes", *ACM Trans. Graph.*, 2003, 22(3), 358–363, (Proc. SIGGRAPH 2003).

[12] X. Gu and S.-T. Yau, "Computing Conformal Structures of Surfaces", *Communications in Information and Systems*. 2002, 2(2):121-146.

[13] X. Gu, Y. Wang, T.F. Chan, P.M. Thompson and S.T. Yau, "Genus Zero Surface Conformal Mapping and its Application to Brain Surface Mapping", *IEEE Transactions on Medical Imaging*, 2004, 23(8), 949-958.

[14] S. Haker, S. Angenent, A. Tannenbaum, R. Kikinis and G. Sapiro, "Conformal Surface Parametrization for Texture Mapping", *IEEE Transactions on Visualization and Computer Graphics*, 2000, 6(2), 1-9.

[15] E. Isaacson and H.B. Keller, *Analysis of Numerical Methods*. Wiley, 1966.

[16] M. Isenburg, S. Gumhold and C. Gotsman, "Connectivity Shapes", *Proceedings of Visualization*, 2001.

[17] G. Karypis and V. Kumar, "Multilevel k-way Partitioning Scheme for Irregular Graphs", *Journal of Parallel and Distributed Computing*, 1998, 48:96-129.

[18] L.P. Kobbelt, J. Vorsatz, U. Labisk and H.-P. Seidel, "A Shrink-Wrapping Approach to Remeshing Polygonal Surfaces", *Proceedings of Eurographics*, 1999.

[19] R.J. Lipton and R.E. Tarjan, "A Separator Theorem for Planar Graphs", *SIAM J. Appl. Math.*, 1979, 36, 177-189.

[20] L. Lovasz, "Steinitz Representations of Polyhedra and the Colin de Verdiere Number", *Journal of Combinatorial Theory B*, 2001, 82,223-236.

[21] L. Lovasz and A. Schrijver, "On the nullspace of a Colin de Verdiere matrix", *Annales de l'Institute Fourier*, 1999, 49, 1017-1026.

[22] U. Pinkall and K. Polthier, "Computing Discrete Minimal Surfaces and their Conjugates", *Experimental Mathematics*, 1993, 2(1), 15-36.

[23] E. Praun, and H. Hoppe, "Spherical Parametrization and Remeshing", *ACM Transactions on Graphics*, 2003, 22(3), 340-349 (Proc. SIGGRAPH 2003).

[24] J. Ruge and K. Steuben, "Algebraic multigrid." In S. McCormick, editor, *Multigrid Methods*. SIAM, 1987.

[25] S. Saba, *Barycentric Spherical Embeddings – Theory and Algorithms*. M.Sc. Thesis, Computer Science, Technion, 2005.

[26] A. Shapiro and A. Tal, "Polygon Realization for Shape Transformation", *The Visual Computer*, 1998, 14(8-9), 429-444.

[27] A. Sheffer, C. Gotsman and N. Dyn, "Robust Spherical Parameterization of Triangular Meshes", *Computing*, 2004, 72(1-2), 185-193.

[28] W.T. Tutte, "How to Draw a Graph", *Proc. London Math. Soc.*, 1963, 13(3), 743-768.

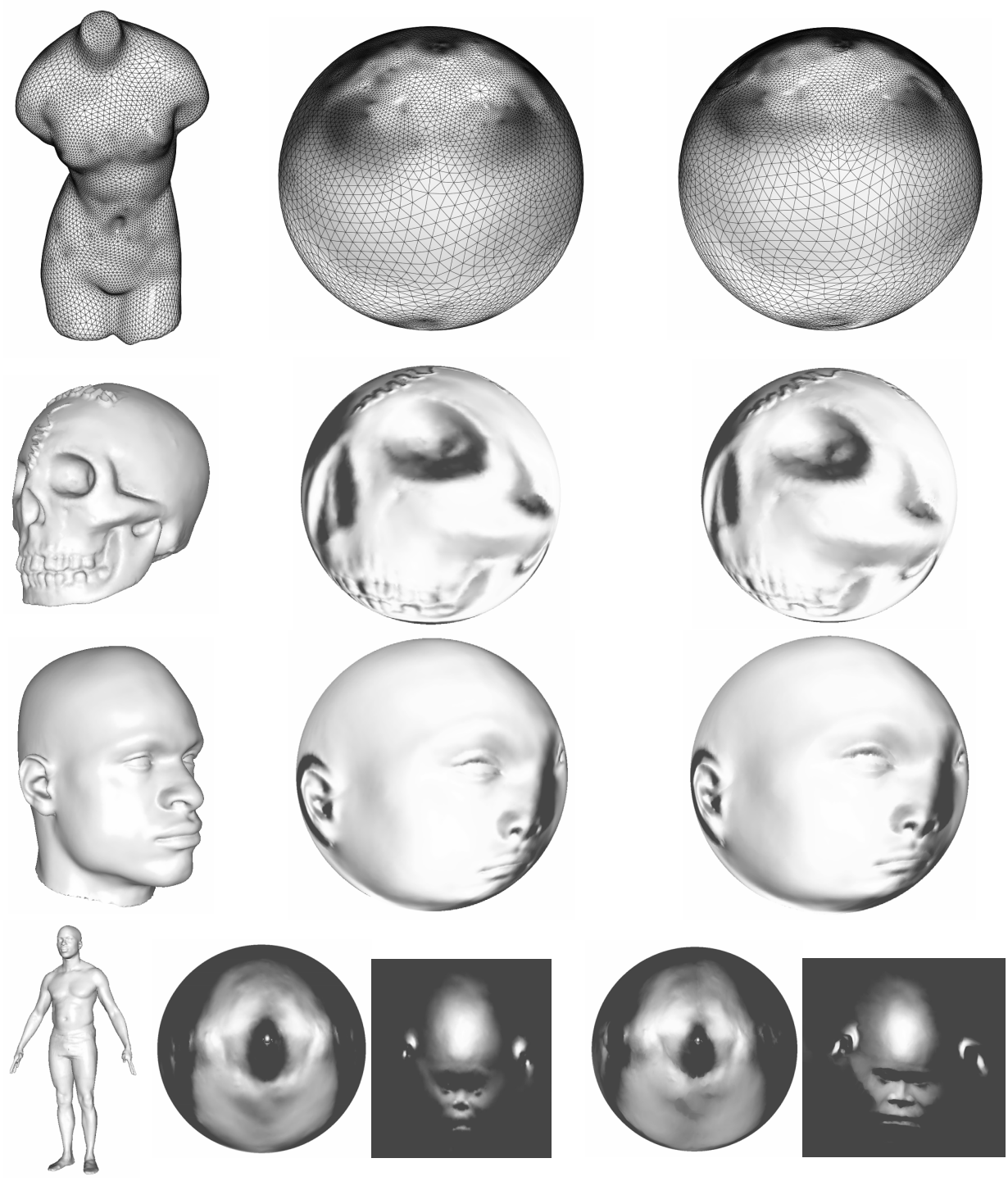

Figure 3. Spherical parameterization using uniform (middle column) and mean-value (right column) weights. The zoom in on the head region of the human model highlights the superior shape preservation achieved by mean value weights compared to uniform weights for irregular meshes (sharper features such as ears, mouth and nose)

Appendix

Multigrid computational methods employ a hierarchy of progressively coarser grids to accelerate the solution of linear systems of equations, usually arising from the discretization of boundary-value problems. Invented for regular grids, the scope of multigrid methods has been widened considerably with the introduction of Algebraic Multigrid (AMG) techniques. This also allows the solving of sparse unstructured problems and handling of non-PDE applications. In particular, it is very well suited to the linear systems that arise in mesh parameterization to the plane using barycentric coordinates. We adopt the classical approach of Ruge and Steuben [24], which is briefly described below. For a complete elementary presentation of AMG, see Chapter 8 of [3].

The AMG algorithm for the iterative solution of the sparse linear system, $Lx = f$, consists of a *setup* phase and a *solve* phase. In the setup, a set of progressively coarser systems and associated approximations is constructed, starting with the full vector of variables, and ending with a very coarse system of just a few variables. Each coarsening is performed by choosing a subset of variables in the current system. The main criterion for choosing the coarse variables is the "strength of connections" as reflected in the relative sizes of elements of the matrix, L, and its approximations in the coarser systems. Specifically, a threshold, $0 \le t \le 1$, is chosen, typically 0.25. Then, if $L(i,j)$ is larger in absolute value than t times the largest off-diagonal element of L, then variable i is said to *depend* on variable j, and variable j is said to *influence* variable i. Based on this notion of dependence, the variables are partitioned into two categories: C variables – the subset that will comprise the next-coarser system, and F variables – the rest. The set C is generated by a graph algorithm based on two criteria:

1. Every variable j, which influences variable $i \in F$, must itself be in C, or else variables i and j must both depend on at least one variable $k \in C$.
2. C should be a maximal subset with the property that no variable in C depends on another variable in C.

It is sometimes not possible to satisfy both criteria. In this case, it is preferable to satisfy the first, while using the second as a guide. These criteria generally lead to coarse systems that strike a good balance between a significant reduction in the number of variables and good approximation properties. Note that the construction of the level is based on pure algebraic considerations, and is not directly related to the geometry of the mesh from which the problem is derived.

In addition to constructing the hierarchy, suitable sparse *prolongation* (interpolation) operators are constructed for transferring information between levels of the hierarchy. These operators are also determined by L and its coarse versions. Generally, the set $C_i \in C$ of coarse variables used in the interpolation to variable $i \in F$, is the set of C-variables that influence variable i. The interpolation coefficients are chosen such that errors that are *not* eliminated efficiently by the Gauss-Seidel iteration will be interpolated accurately. This allows the coarse approximation to correct such errors very effectively, and thus all errors are eliminated efficiently by the algorithm described below.

More formally, if the solution vector, x, is of length n, then in the setup phase we construct a set of progressively coarser vectors of sizes $n_0 = n, n_1, n_2, \ldots, n_k$, corresponding prolongation matrices $P_0, P_1, P_2, \ldots, P_{k-1}$, where P_j is a matrix of n_j rows and n_{j+1} columns that are used to interpolate data from level j+1 to level j, and a set of matrices, $L_0 = L, L_1, L_2, \ldots, L_k$. The latter matrices are constructed in order, using the Galerkin principle: $L_{j+1} = P_j^T L_j P_j$. The number of levels, k+1, is chosen such that n_k is small enough for the problem at level k to be solved directly at a negligible cost.

Once the setup phase is complete, the problem is solved iteratively by multigrid *cycles*, defined recursively below. Here, $\tilde{x}_j$ denotes at all times the most up-to-date approximation for the solution of the level-j problem. To solve the linear system $Lx = f$, call algorithm AMG with $j = 0$, some initial approximation, $\tilde{x}_0$, and $f_0 = f$. The parameters ν_1 and ν_2 are fixed small non-negative integers, typically both equal to 1.

AMG$(\tilde{x}_j, f_j, j)$

1. If $j = k$, return $L_j^{-1} f_j$.
2. Perform ν_1 Gauss-Seidel iterations on the system $L_j x_j = f_j$ with initial guess $\tilde{x}_j$.
3. Compute the residual, $r_j = f_j - L_j \tilde{x}_j$, and restrict it to the next coarser level by $f_{j+1} \leftarrow P_j^T r_j$. Set $\tilde{x}_{j+1} = 0$.
4. Recursively call $\tilde{x}_{j+1} \leftarrow$ **AMG**$(\tilde{x}_{j+1}, f_{j+1}, j+1)$.
5. Interpolate and add the coarse-level correction by $\tilde{x}_j \leftarrow \tilde{x}_j + P_j \tilde{x}_{j+1}$.
6. Perform ν_2 Gauss-Seidel iterations on the system $L_j x_j = f_j$ with initial guess $\tilde{x}_j$ and return the result.

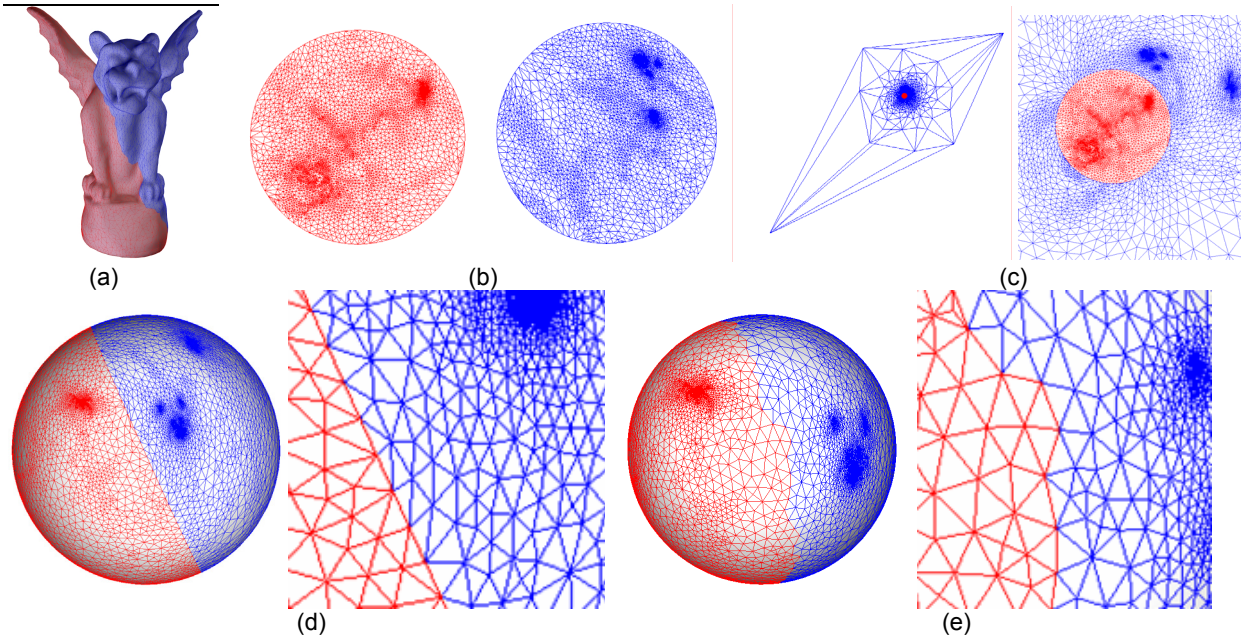

Figure 1. Stages in generating a spherical embedding for the gargoyle model (using uniform weights): (a) Partition into two sub-meshes using MeTiS. (b) Planar parameterization of the sub-meshes. (c) Combined planar embedding (with zoom); (d) Initial spherical parameterization generated by inverse stereo projection (with zoom); (e) final result after projected Gauss-Seidel and local Newton iterations (with zoom).

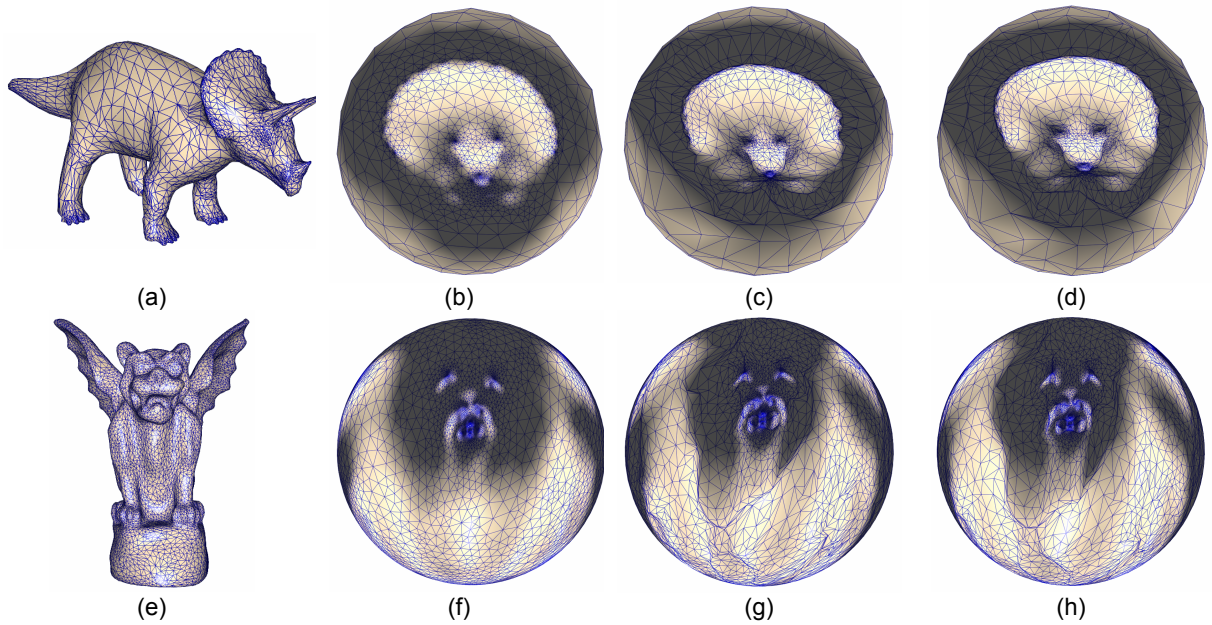

Figure 2. Spherical parameterization of the triceratops (a) and gargoyle (e): using uniform weights (b, f), using conformal weights (c, g) and using mean value weights (d, h).