Spherical Parameterization of Meshes

Chris Fawcett
CMPT 469 Final Project
Spring 2005
Overview

- The parameterization problem.
- Spherical parameterization in general.
- Methods that reduce to the planar case.
- Methods for direct parameterization onto the sphere.
- Gotsman et al.’s method for spherical parameterization.
- Explanation and demo of my framework.
Parameterization

• Given a mesh $M$, possibly of genus $>0$ and with boundary, find a continuous invertible map $\phi: D \to M$ from some parameter domain to the mesh.

• Generally, one wants a parameter domain $D$ that is homeomorphic to the original mesh $M$. 
Why is it useful?

• Parameterization has direct applications to texture mapping, meshing, remeshing, morphing, and more.

• Some problems become much easier or easier to quantify when dealing with a uniform domain.
Spherical Parameterization

- Given a closed triangle mesh $M$ of genus 0, find a mapping from the unit sphere to $M$.
- This is equivalent to embedding the connectivity graph of $M$ onto the unit sphere $S$, such that the resulting spherical triangles form a partition of $S$. 
Spherical Parameterization

• Steinitz: A graph $G$ may be embedded on the sphere $S$ if and only if $G$ is planar and 3-connected.

• Consequently, any closed triangle mesh of genus 0 has a spherical triangulation.

• The problem is actually finding a good one!
Quality of a Parameterization

- There are two major considerations: distortion and validity

- Distortion, intuitively, is a measure of how much the triangles in our mesh $M$ are “stretched” by applying the parameterization.

- A parameterization is valid if and only if the spherical triangles that result form a partition of the unit sphere (i.e., there are no triangle overlaps).
Previous Methods

- Method 1: Reduce the problem to the planar parameterization case, and use one of the numerous methods available to find a solution (Tutte embedding, angle-based flattening, etc).
How about this?

- Cut some triangle out of the mesh, leaving a mesh that is homeomorphic to a disc, and parameterize over the unit triangle.
- Afterwards, map the planar parameterization to the unit sphere by way of the inverse stereo projection.
• Problem: Severe distortion. Most planar parameterization methods using the unit triangle as a boundary tend to cluster the remaining vertices in the center of the triangle.

• Also, the result is not even guaranteed to be a valid spherical parameterization.
Or this?

• Find some cut of the connectivity graph, and parameterize each piece over a planar region with a common boundary.

• Afterwards, map each disk to a hemisphere.

• Since we used a common boundary in the planar parameterizations, the hemispheres will line up.
• Less distortion than the previous method, since the common boundary was (probably) more than three vertices.

• However, the quality depends heavily on the graph cut we choose, which is a rather hard optimization problem.
Previous Methods

- Method 2: Parameterize onto the unit sphere directly, instead of reducing the problem to planar parameterization.
- This is much more intuitive and natural.
Heuristic relaxation.

- Alexa[2000, 2002] presents methods for direct parameterization that use the so-called the “Straight-Arc Embedding Algorithm.”

- Pick some peripheral cycle in the connectivity graph, and anchor the corresponding vertices. Now, repeatedly move the remaining vertices to the centroid of their neighbours projected back onto the sphere.
• Requires careful selection and placement of the anchor vertex, and tweaking of heuristics to do the relaxation.

• Can result in collapsed embeddings and/or triangle foldover, requiring the algorithm to be restarted, until a valid embedding is found.

• Even if the result is valid, we don’t have any bound at all on the distortion.
Praun and Hoppe [2003] present an algorithm for direct spherical parameterization.

- Decimate the mesh M to a tetrahedron, while constructing a progressive mesh favoring triangles with good aspect ratios.
- Map the base tetrahedron to the sphere.
Progressive Meshes

- Traverse the progressive mesh sequence, inserting vertices on the sphere while incrementally maintaining an embedding and minimizing a chosen stretch metric.

- Finds a local minima of the stretch metric, but not necessarily the global minimum.
Gotsman et al.’s method

- Assuming the existence of a stable numerical procedure for solving quadratic systems, this method will find a valid parameterization with distortion properties dictated by a chosen symmetric Laplacian $L_w$. 
CdV Matrices

• Given an n-vertex graph $G = (V, E)$, consider the class of symmetric matrices $M(G)$ where $M_{ij} = \begin{cases} \text{negative number if } (i,j) \in E, & \text{anything if } i = j, \\ 0 & \text{otherwise} \end{cases}$

• Clearly $M(G)$ is a superset of the symmetric Laplacians for $G$. 
CdV Matrices

• let \( \{\lambda_0, \lambda_1, ..., \lambda_{n-1}\} \) be the spectrum of some \( M \). Let \( r \) be the maximal integer such that \( \lambda_1 = \lambda_2 = ... = \lambda_r \) over all matrices \( M(G) \). Let \( M_p \) be some matrix attaining this maximum.

• \( r \) is called the Colin de Verdiere (CdV) number of \( G \), \( M_p \) is a CdV matrix for \( G \), its \( r \) identical eigenvalues are CdV eigenvalues, and the associated eigenvectors are CdV eigenvectors.
CdV Matrices

- G is a 3-connected planar graph if and only if $r(G) \leq 3$.
- Lovasz and Schrijver [1999] showed that CdV eigenvectors of a graph G may be used to embed G in $\mathbb{R}^{r(G)}$.
- For $r(G) = 3$, G describes the edges of a convex polyhedron in $\mathbb{R}^3$ containing the origin if the three eigenvectors of a CdV matrix of G are used as basis vectors for its vertices.
Gotsman et al.’s method

- With proper scaling of the eigenvalues, the CdV eigenvectors are a basis for the nullspace of the CdV matrix M, and form a spherical nullspace embedding of G.

- We can do this by solving a system of $4n$ quadratic equations, where $n$ is the number of vertices in G.
Quadratic System

\[ x_i^2 + y_i^2 + z_i^2 = 1 \quad i = 1, \ldots, n \]
\[ \alpha_i x_i - L_{W[i]} x = 0 \quad i = 1, \ldots, n \]
\[ \alpha_i y_i - L_{W[i]} y = 0 \quad i = 1, \ldots, n \]
\[ \alpha_i z_i - L_{W[i]} z = 0 \quad i = 1, \ldots, n \]

- variables are \( \alpha_i, x_i, y_i, z_i \).
- \( L_{W[i]} \) is the ith row of some chosen symmetric Laplacian, and \( x, y, z \) are column vectors containing the original coordinates.
Gotsman et al.’s method

- There are some gotcha’s to avoid degenerate solutions.
- Can eliminate two degrees of freedom from the system by anchoring two vertices.
My Parameterization Framework
My Parameterization Framework

• GLUI-based GUI for examining meshes, computing parameterizations of meshes and evaluating/comparing the quality of the result.

• Extremely easy to add new parameterization procedures (Once the procedure itself is coded, that is).
My Parameterization Framework

- Based around a half-edge mesh data structure allowing for fast lookup and traversal.
- augmenting the SMF file format to hold parameterizations.
- Able to load meshes and save meshes/parameterizations.
Screen shots
Screen Shots
The downside?

- After all this work, I couldn’t find a library for solving Quadratic Systems that was able to work. (Gotsman et al. used MATLAB for everything)

- Submitted request to use the QPA/QPB component of GALAHAD, but have yet to hear a response. To their credit, it’s been less than a week.
The downside?

• To demonstrate the system, I coded a simple spherical projection that doesn’t at all guarantee validity (see cow image).

• However, we are left with a program that does some very nice things, and provides a great way to visualize parameterizations created by other programs (All the Hoppe examples, for instance).
Extensions

- Implementing more parameterization algorithms, obviously.
- Optimizing the half-edge data structure to avoid unnecessary pointer dereferencing.
- A better way to visualize the parameterizations of large meshes. Currently, you can’t really see the geometry.
- Some method of easily painting sections of the original mesh and having them mirrored in the parameterization display would be nice.