**Skeleton Extraction**

**from Mesh Contraction**

**Chapter 1. Introduction**

The extraction of curve-skeletons from 3D meshes has inumerous applications in computer graphics and visualization. The most obvious is the animation of models using the curve as a controller, but other possibilities include shape retrieval, shape deformation and morphing. Overall, the dimensionality reduction of a representation can be useful in many domains, as it simplifies the data structures and the subsequent analysis. Despite the importance of the problem, [Cornea et al. 2007] stated in 2007 that there were still no efficient and robust algorithms to automatically generate a skeleton from a mesh.

In this work, we'll implement the article "Skeleton Extraction by Mesh Contraction", presented at SIGGRAPH 2008. The method efficiently extracts a curve-skeleton directly from the mesh, without converting it to a volumetric counterpart. The algorithm can be divided in three steps: (a) geometric smoothing (b) topological simplification and (c) embedding correction. The first step consists of applying a constrained Laplacian smoothing until the volume of the mesh is near zero. After that, we simplify the topology of the skeleton by collapsing edges that do not contribute to the overall shape of the skeleton. The metric used to decide which edges to collapse is derived from the seminal paper by Garland and Heckbert on Quadric Error Metrics [Garland and Heckbert 1997]. Finally, we perform a correction on the embedding of the skeleton, so that every part of the skeleton is guaranteed to lie within the mesh.

**Chapter 2. Theoretical Background**

The method we implemented can be divided in three main parts: (a) geometric smoothing (b) topological simplification and (c) embedding correction. The first part is based on the minimization of the Laplacian coordinates of the mesh. In Sections 2.1 and 2.2, we'll go through the definition of the Laplace Operator in continuous domains and the discretization of this operator to be applied to 3d meshes. For the sake of completeness, we briefly review Least Squares Minimization in Section 2.3. We also added a short explanation of the column compressed format for sparse matrices in Section 2.4, since the skeletonization of any mesh except very small ones would be unfeasible without it. Finally, in Section 2.5, we will discuss the Quadric Error Metric, as we used an adaptation of this metric in the topological simplification.

**Section 2.1. The Laplace Operator**

The Laplace operator is defined as the divergence of the gradient of a function on Euclidean space. Given a vector field, the divergence is a measure of the behavior of this vector field around an infinitesimal region. Suppose we have a vector field that describes the movement of a fluid. The divergence would be, for an infinitesimal region, the difference how much of the fluid is going into this region and how much fluid is going out of it. Figure 2.1.1 helps explaining this concept. Equivalently, the Laplacian is the sum of all unmixed second partial derivatives of the function.

Figure1

**Section 2.2. The Discrete Laplace Operator**

The discrete Laplace operator is the analogous of the continuous Laplace operator, defined on graphs and discrete grids, instead of functions. It is commonly called Laplacian matrix, since its representation is a matrix, when applied to finite graphs. The application of the Laplace operator to infinite graphs is out of the scope of this work.

While the mathematical theory behind the discrete Laplace operator is the same as in the continuous case, it's implementation on graphs is very simple. For every node, the Laplacian matrix calculates the difference between the value of the node and the weighted sum of its neighbors, as in the following equation:

(1)

where is a function defined in each node of the graph, w is a node adjacent to v, and γ*wv* is the weight for the wv edge, usually its length. As better explained in section 3, we use the laplacian operator of the curvature function of the mesh.

When discretizing the Laplacian, we would probably wish to keep (a subset of) the properties of the continuous case. Which properties we wish to maintain will have an influence in the weight matrix. Indeed, it has been proved [] that a perfect discretization of the Laplacian operator, i.e. one that has all desirable properties of the continuous case, cannot exist.

**Section 2.3. Least Squares Minimization**

**Section 2.4. Column Compressed Format for Sparse Matrices**

The representation of an object as a 3D mesh often demands thousands of vertices. A naïve data structure to hold the Laplacian of such a shape would need huge amounts of memory and would impair the usage of the algorithm in any but very small meshes. To our luck, however, the Laplacian matrix is very sparse, i.e., is has a huge quantity of zeros. This has a straightforward geometric intuition: while there may be thousands of vertices in a mesh, every vertex is usually connected to only a few neighbors.

The column compressed representation is very simple. Instead of enumerating every value in every row and column, it assumes the whole matrix is filled with zeros, and then enumerates the exceptions.

**Section 2.5. Quadric Error**

Lorem ipsum lorem ipsum

**Chapter 3. Algorithm**

Starting from a mesh of triangles G = (V,E) where V = (V1T, V2T, ... VnT) are the vertices coordinates, and E is the set of edges, we create the matrix Lnxn as the curvature-flow Laplacian operator, with

(2)

Solving *LV'=0* would lead to new vertex coordinates *V'* representing a smoothly contracted version of the original mesh *G*. However, the matrix *L* created is singular, and we need more constraints to ensure a unique solution for *V'*. Thus we add new constraints to the system, that also make the contracted mesh keep the original overall shape, we call these new constraint *attraction constraints*. The constraints defined by the rows in L are called *contraction constraints.*

Then the system becomes

(3)

where and are diagonal matrices to balance the contraction and attraction factors respectively. This new system is over-determined, thus we solve it with least-squares approach.

It requires several iterations of system (3) to reduce the mesh to the skeleton form, where the constraints weights and the matrix L must be updated in each iteration. To increase the collapsing speed, the weight is increased on each iteration, and to avoid over contraction is updated to each vertex according to its one-ring area.

Now, the iteration t is as follows:

1. Solve for
2. Update
3. Update where and are the current and the original one-ring area of the vertex i.
4. Compute the new with the new vertex positions

We used the following initial values, as suggested by the authors:

* where A is the average face are of the mesh.

**Chapter 4. Results**

Lorem ipsum lorem ipsum

**Chapter 5. Conclusion**

Lorem ipsum lorem ipsum

**References (REPASSAR!)**

[Au et al. 2008] Au, O. K. C., Chiew-Lan Tai, Hung-Kuo Chu, Daniel Cohen-Or, and Tong-Yee Lee. 2008. Skeleton extraction by mesh contraction. ACM Trans. Graph. 27, 3, Article 44 (August 2008), 10 pages.

[Cornea et al. 2007] CORNEA , N. D. , MIN , P. , and SILVER , D. 2007. Curve-skeleton properties, applications, and algorithms. *IEEE Transactions on Visualization and Computer Graphics.* 13, 3, 530–548.

[Garland and Heckbert 1997] GARLAND , M . , and HECKBERT, P. S . 1997. Surface simpliﬁcation using quadric error metrics. In *Proceedings of ACM SIGGRAPH 97*. 209–216.

[Max Wardetzky et al. 2007] Wardetzky, M., Mathur, S., Kälberer, F. and Grinspun, E. 2007. Discrete laplace operators: no free lunch. In Proceedings of the fifth Eurographics symposium on Geometry processing (SGP '07). Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 33-37.