**Skeleton Extraction**

**from Mesh Contraction**

Guilherme N. Oliveira

Jonatas Medeiros

Rosália G. Schneider

**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 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 (or Laplacian) 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 “outgoingness” 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. Equivalently, the Laplacian is the sum of all unmixed second partial derivatives of the function.

**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, its 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:

(2.2.1)

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, *e.g.* its length.

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 [Wardetzky et al. 2007] that a perfect discretization of the Laplacian operator, *i.e.* one that has all desirable properties of the continuous case, cannot exist. We’ll discuss this further on Chapter 3, as we use the Laplacian operator of the curvature function of the mesh on our implementation.

**Section 2.3. Least Squares Fitting**

Least Squares is the standard approach to find as-good-as-possible solutions to overdetermined systems that have no exact solution. The solution found minimizes the squared error in each equation.

In our case, we are dealing with Linear Least Squares. Given a system of equations Ax = b, the least squares solution can be found simply by solving ATAx = ATb instead.

**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. Fortunately the Laplacian matrix is very sparse, *i.e.*, most of its elements are 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.

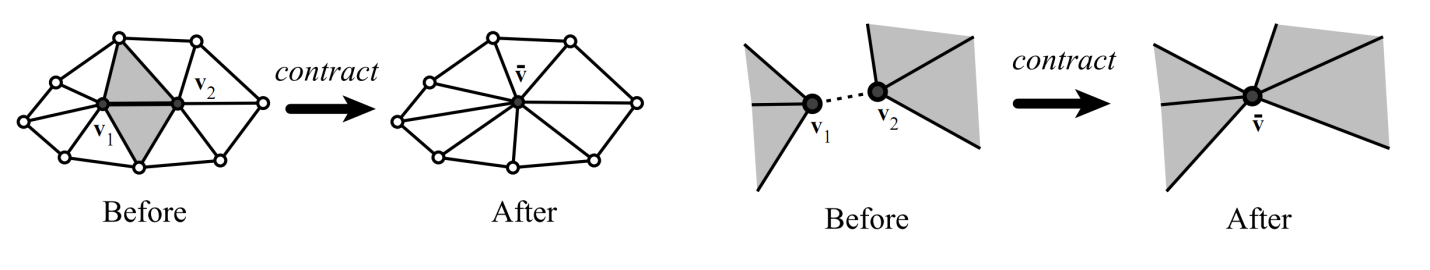
The column compressed representation is very simple. Instead of enumerating every value in every row and column, it assumes the whole matrix is filled with zeros, and then enumerates the exceptions.

The average valence, the number of neighbors of a vertex, of a triangle mesh is approximately 6 (it can be derived from Euler’s formula, V - E + F = 2(1-g), where V is the number of vertices, E the number of edges, F the number of faces and g the genus - we assume most meshes have a low genus). Using that fact, we can see that the memory usage was reduced from Θ(n2) to Θ(n).

**Section 2.5. Quadric Error Metric**

To simplify the connectivity of our mesh once its volume is small, we use an adaptation of the Quadric Error Metric Simplification, introduced by Garland and Heckbert [Garland and Heckbert 1997]. This technique takes as input a triangle mesh and outputs a simplified model that approximates the original shape as much as possible.

The algorithm is based in getting two vertices near each other and collapsing them into one (see Figure 2.5.1). Note that this is different from edge collapsing, since the vertices are not required to be connected.



**Figure 2.5.1.** The two cases in vertex contraction. The algorithm does not preserve topology or even guarantees that the resulting mesh will be a manifold. Taken from [Garland and Heckbert 1997].

Once we understood the basic operation, we are left with two questions: (1) which vertices should be considered for collapsing and (2) which contractions should be performed first, in order to achieve the best possible simplified mesh?

For the first question, the authors adopted a very simple solution: A vertex pair can be connected if it is connected by an edge or is the distance between them is smaller than a threshold.

The answer to the second question is to use quadric error matrices. The authors associate a matrix Qi to each vertex vi and define the error to be viTQivi. Given a contraction (v1, v2) → v’, the error at v’ is given by v’TQ’v’, where Q’ = Q1+Q2. The original algorithm finds an optimal v’ between v1 and v2.

The algorithm is then a repetition of the following steps:

1. Select valid pairs
2. Associate a Qi matrix with every vertex
3. Compute the optimal position for v’ for every possible contraction
4. Contract the edge with the least cost.

The only remaining question is how to compute the initial Q matrices. The authors note that every vertex can be seen as a intersection of the planes defined by its adjacent faces. A possible definition to the error would be the sum of the square distances from the vertex to the planes.

(2.5.1)

where p is the vector [a b c d] T defining the ax + by + cz + d = 0 plane. With some straightforward algebra, we can derive Q.

(2.5.2)

(2.5.3)

(2.5.4)

**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 area of the mesh.

After the geometric contraction step, we are left with a version of the mesh that has no volume, yet yields the same connectivity as the original model. Since this connectivity adds no info to the shape of the skeleton, it should be simplified.

At this point, we use a similar approach to QEM introduced in Section 2.5. Since the faces of the contracted mesh are degenerated, we use the distance to the lines spanned by the edges to generate the Q matrices. Another modification is to allow contraction only of connected vertex avoiding changes in the mesh topology like in the right half of figure 2.5.1.

So, for a contracted mesh



we define for each edge its cost function



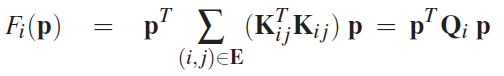
where the first term is a shape cost function and the second one is a sampling cost function weighted by the constants



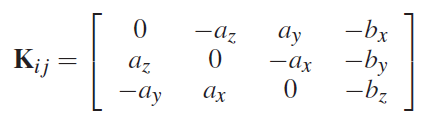
The shape cost function is used to simplify the mesh while retaining its original topology, and is given by



F is the quadric error metric of the vertex, and is defined as



with

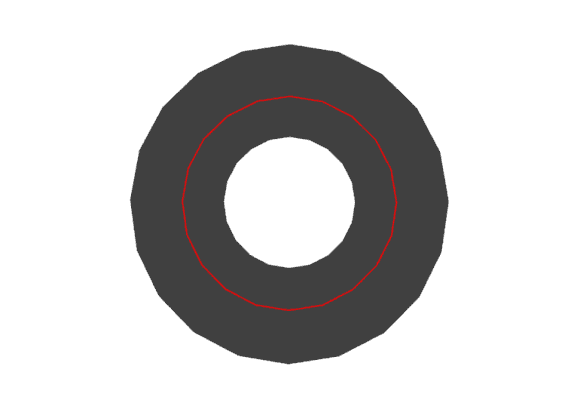
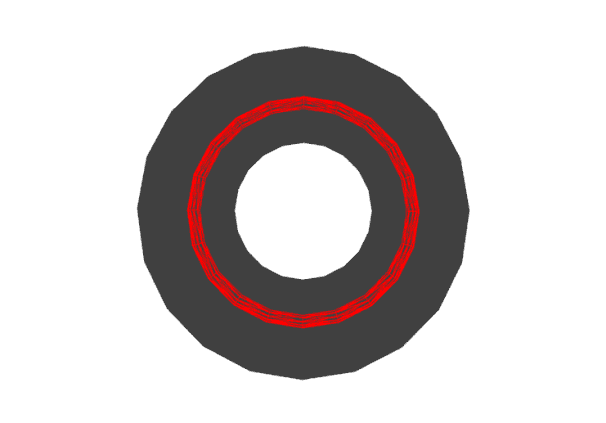
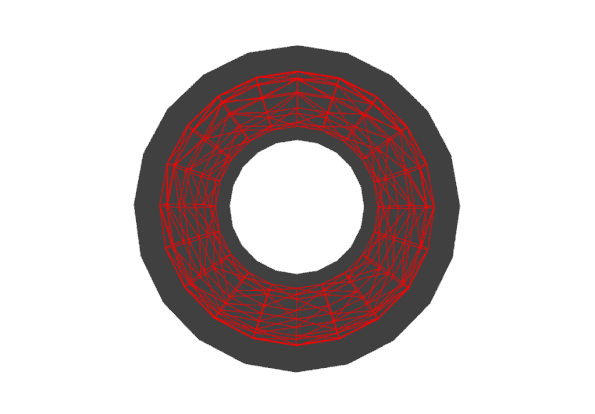
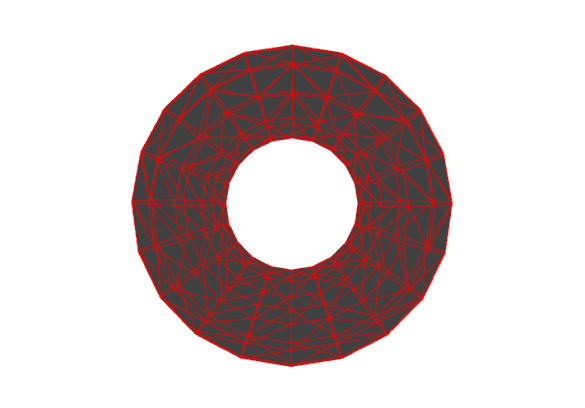


where is the normalized edge vector of the edge (i,j) and

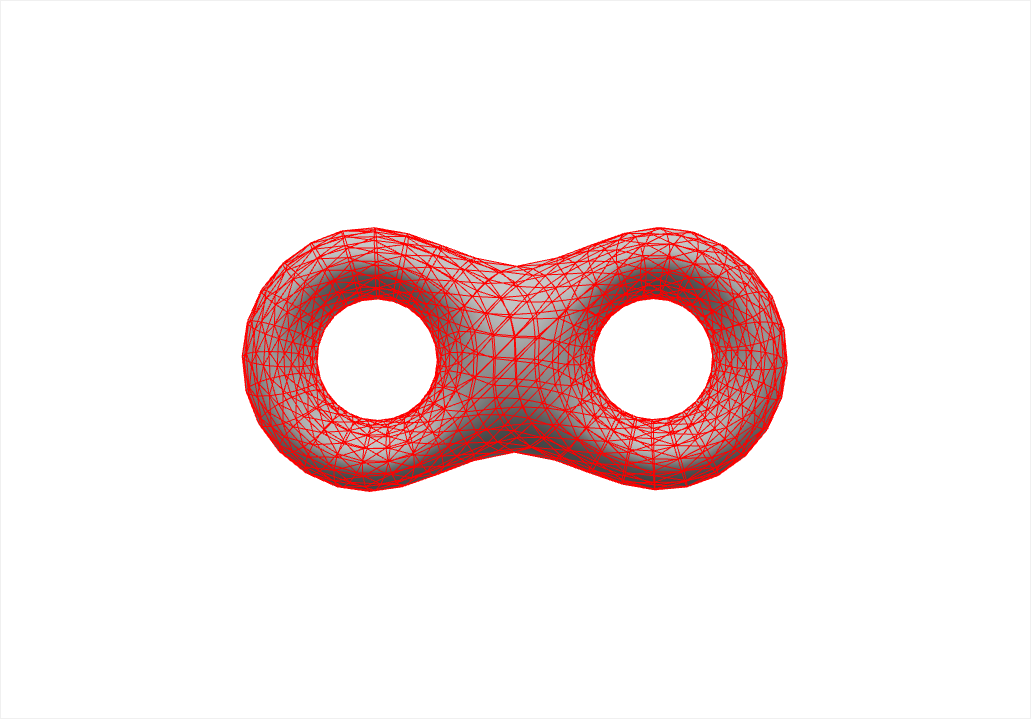
And after the edge (i,j) collapses to the vertex j, we update the error matrix of vertex j as to associate now to j the edges previously associated with i.

**Chapter 4. Results**

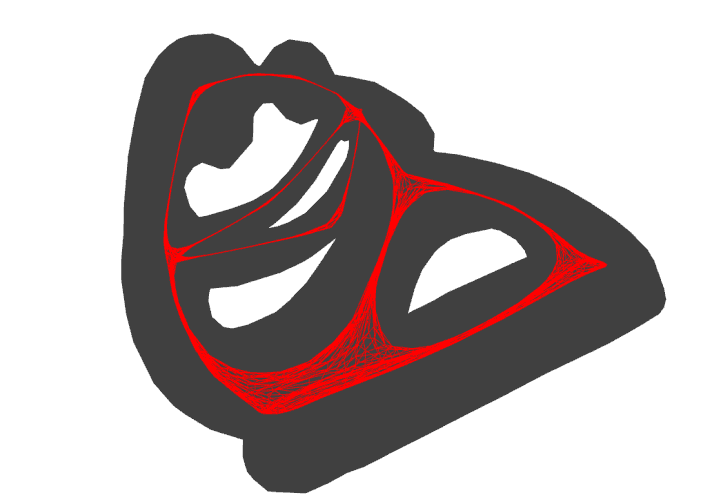
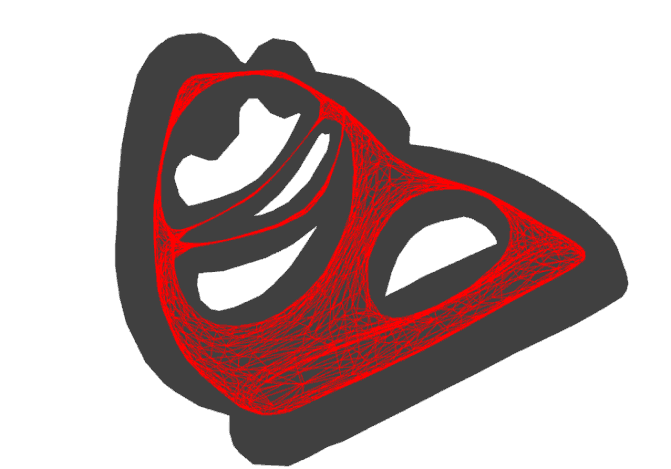
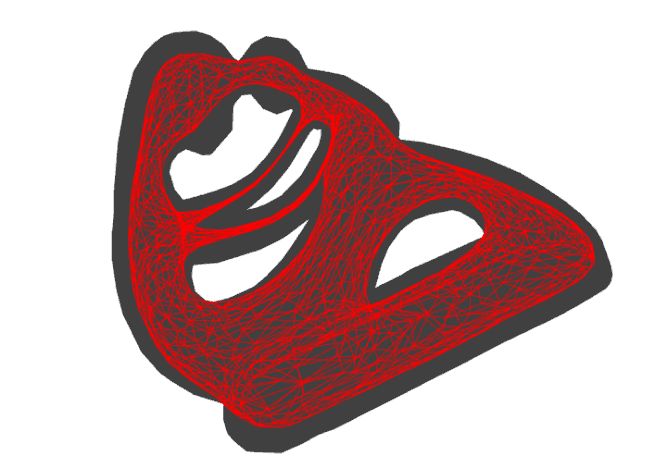
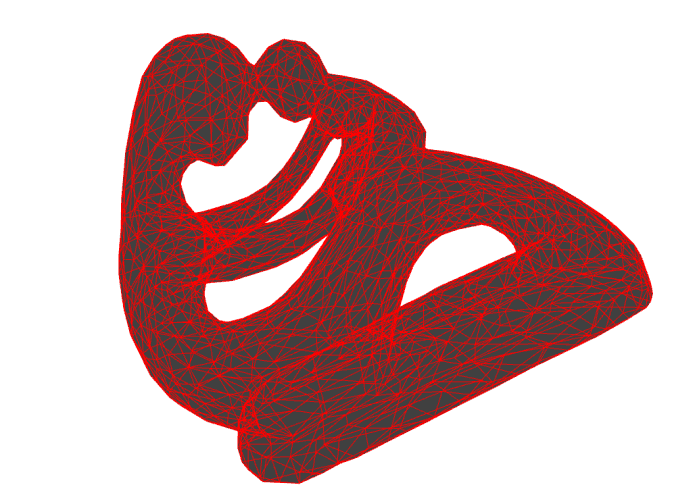
We implemented the skeleton extraction method using C++, OpenGL, Qt for the Graphical Interface, OpenMesh for the mesh management and Cholmod for the solution of the linear system of equations. We applied the algorithm to …

****

**Figure 4.1** Results for iterations 1, 2, 3 and 4 for the torus model.

****

**Figure 4.2** Results for iterations 1, 2, 3 and 4 for the bitorus model.

****

**Figure 4.3** Results for iterations 1, 2, 3 and 4 for the fertility model.

**Chapter 5. Conclusion**

The skeleton extraction by mesh contraction technique turned out to be a good option when generating curved-skeletons automatically. One of the main difficulties faced was the …

**References**

[Au et al. 2008] AU, O. K. C., TAI, C. L., CHU, H. K., COHEN-OR, D., and LEE, T. Y. 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.

[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.