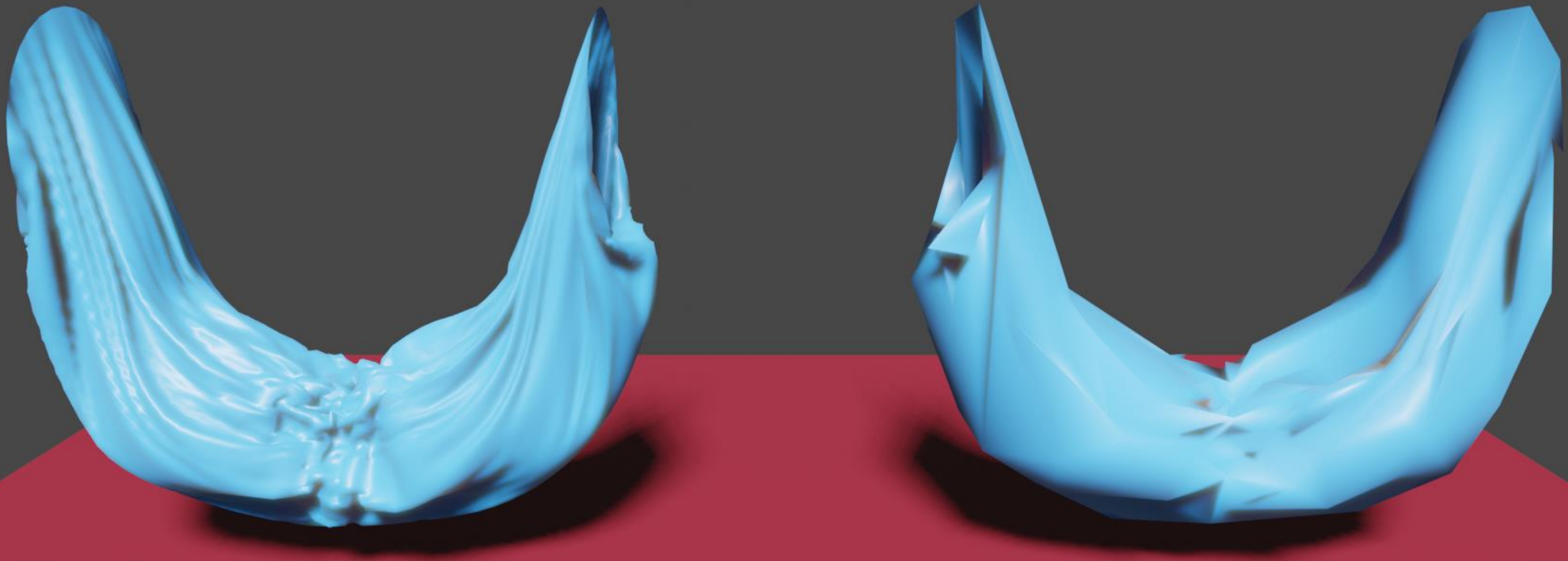
COMP 6381
Geometric Modeling and Processing

Project Demo : Mesh Optimization by Hoppe, H., DeRose, T., Duchamp, T., McDonald, J. and Stuetzle, W.



40228578, Chanhaeng Lee

Contents

- Project Goal
- Problem
- Representation
- First stage of Mesh Optimization
- Second stage of Mesh Optimization
- Result

Project Goal

Mesh Optimization

Hugues Hoppe* Tony DeRose* Tom Duchamp†
John McDonald‡ Werner Stuetzle‡

University of Washington
Seattle, WA 98195

Abstract

We present a method for solving the following problem: Given a set of data points scattered in three dimensions and an initial triangular mesh M_0, produce a mesh M, of the same topological type as M_0, that fits the data well and has a small number of vertices. Our approach is to minimize an energy function that explicitly models the competing desires of conciseness of representation and fidelity to the data. We show that mesh optimization can be effectively used in at least two applications: surface reconstruction from unorganized points, and mesh simplification (the reduction of the number of vertices in an initially dense mesh of triangles).

- Implement the method to optimize vertices for fixed simplicial complexes
- Explain the legal moves for edge operations (edge collapse, edge split, edge swap)
- Implement the method to optimize simplicial complexes to minimize an energy function

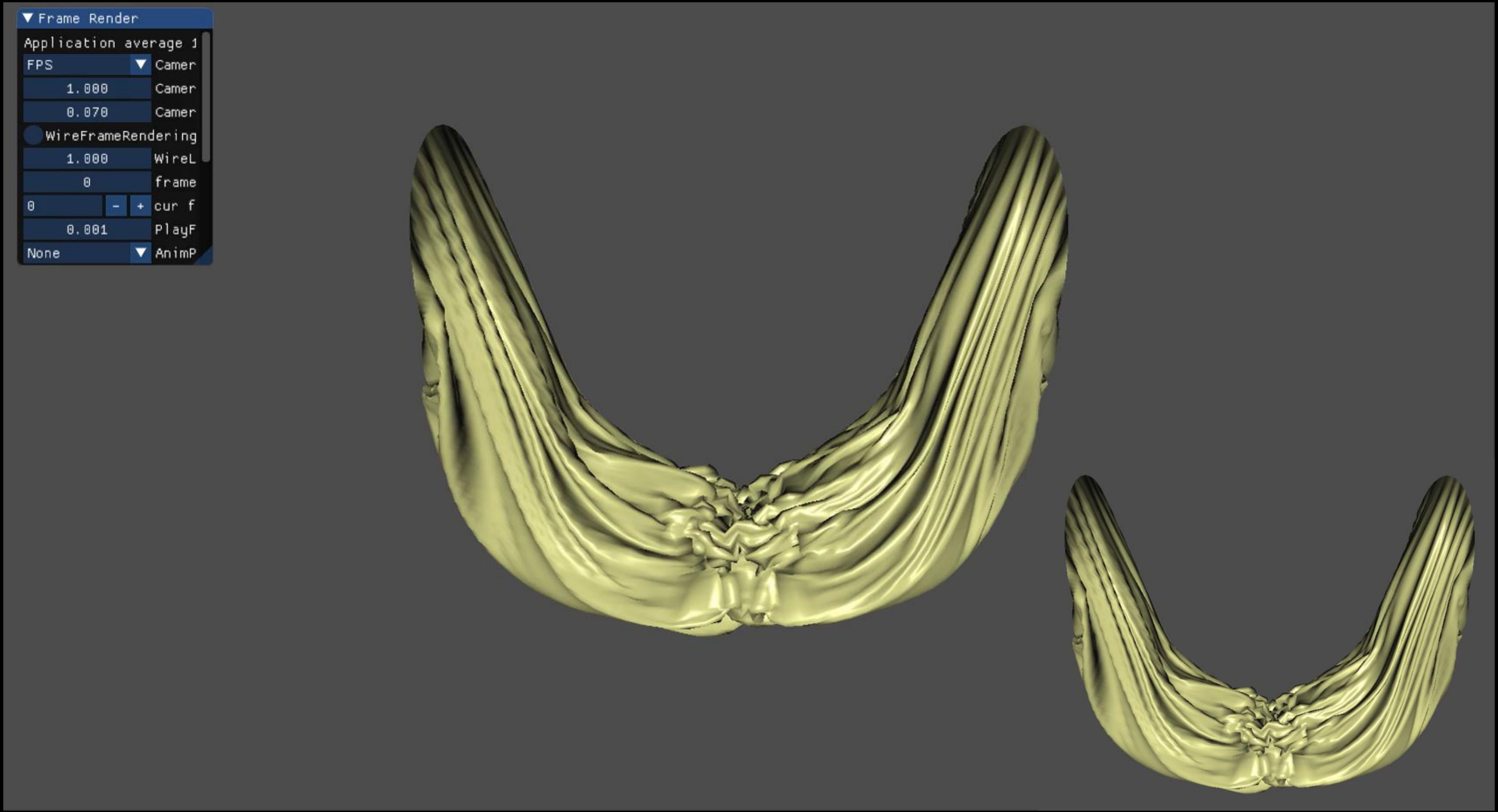
Problem : Mesh Simplification

- Given randomly sampled points from a surface of a triangular mesh M,
  produce a mesh M', of the **same topological type** as M, that **fits the sampled points** and **has a smaller number of vertices** than the original mesh.

- a research of **surface reconstruction** problem from **unorganized points**. → sampled points (**data points**)

- Casting mesh simplification as an **optimization problem** with an energy function

Problem : Animation for each step of mesh simplificiation



From the original mesh (8,320 vertices and 16,384 faces) to the simplified mesh (206 vertices and 392 faces)

Representation : Simplicial Complex

Mesh M is a pair (K, V)
Where K is a simplicial complex and V is a set of vertex positions

N-simplex : a geometry object with (n+1) vertices which lives in an n-dimensional space and cannot fit in any space of smaller dimension.

0-simplex : Point

1-simplex : Line Segment

2-simplex : Triangle

3-simplex : Tetrahedron

https://en.wikipedia.org/wiki/Simplex

Simplicial Complex (Topology) : A collection of simplices.

Representation : Realization from a simplicial complex

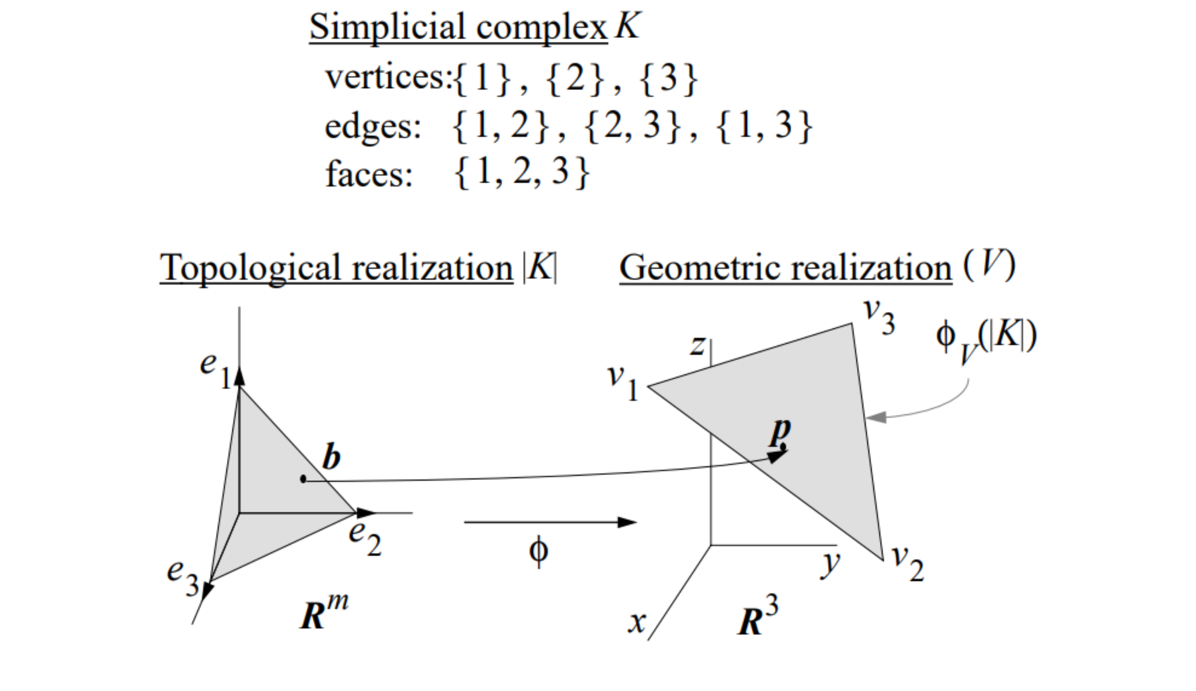


Figure 2: Example of mesh representation: a mesh consisting of a single face.

A simplicial complex K with a set of vertices $\{1, \ldots, m\}$.

Standard Basis Vectors $\{\boldsymbol{e}_1, \ldots, \boldsymbol{e}_m\}$ of $\boldsymbol{R}^m$.

$$\Big\downarrow \phi_V(|K|)$$

Mesh Vertices $\{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m\}$ of $\boldsymbol{R}^3$.

$\boldsymbol{b} \in |K|$ (barycentric coordinates)

$$\Big\downarrow \phi_V(\boldsymbol{b})$$

vertex $\boldsymbol{p}$ in the triangle face $v_1 v_2 v_3$

Energy Function : Overview

Data Points : $X = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_n\}, \boldsymbol{x}_i \in \boldsymbol{R}^3$

Vertex Positions : $V = \{\boldsymbol{v}_1, \dots, \boldsymbol{v}_m\}, \boldsymbol{v}_i \in \boldsymbol{R}^3$

Barycentric coordinates : $B = \{\boldsymbol{b}_1, \dots, \boldsymbol{b}_n\}, \boldsymbol{b}_i \in |K| \subset \boldsymbol{R}^m$

$$E(K,V) = E_{dist}(K,V) + E_{rep}(K) + E_{spring}(K,V).$$

$$E_{dist}(K,V) = \sum_{i=1}^{n} d^2(\boldsymbol{x}_i, \phi_V(|K|)).$$

$E_{rep}(K) = c_{rep} m.$ (m : number of vertices)

$$E_{spring}(K,V) = \sum_{\{j,k\} \in K} \kappa \left\| \boldsymbol{v}_j - \boldsymbol{v}_k \right\|^2$$

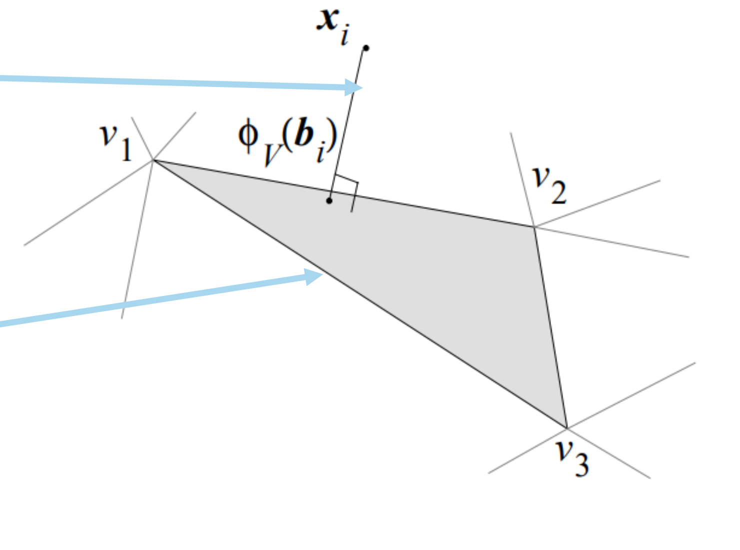


Figure 4: Distance of a point $\mathbf{x}_i$ from the mesh.

Energy Function : Overview

$$E(K,V) = E_{dist}(K,V) + E_{rep}(K) + E_{spring}(K,V).$$

$$E_{dist}(K,V) = \sum_{i=1}^{n} d^2\big(\boldsymbol{x}_i\,, \phi_V(K)\big).$$

If **a vertex** is **added** to the mesh, E_{dist} is likely to be **reduced**.
If **a vertex** is **removed** from the mesh, E_{dist} is likely to **increase**.

$$E_{rep}(K) = c_{rep} m.$$

Penalizes meshes with a large number of vertices.
Prevents adding the vertices indefinitely by E_{dist}.
The **larger** c_{rep} is, the **smaller** the number of vertices gets

$$E_{spring}(K,V) = \sum_{\{j,k\}\in K} \kappa \left\|\boldsymbol{v}_j - \boldsymbol{v}_k\right\|^2$$

E_{spring} acts as a **regularizing term** that helps guide the optimization to a desirable local minimum.

Energy Function : Minimizing Energies

$$E(K,V) = E_{dist}(K,V) + E_{rep}(K) + E_{spring}(K,V).$$

Optimize Vertices V for **fixed** simplicial complex K

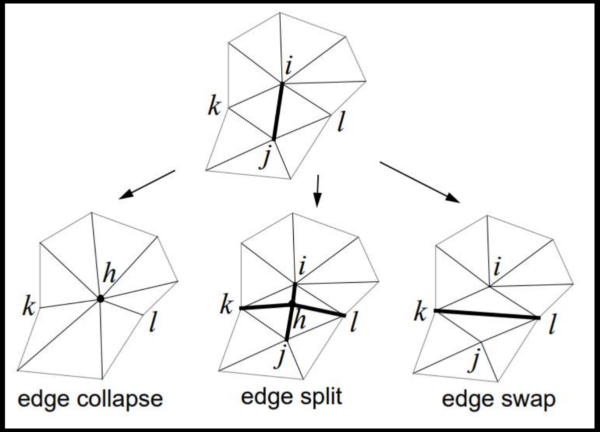
Project data point X to mesh

+

Linear Least Squares Problem

$$E(K,V) = E_{dist}(K,V) + E_{rep}(K) + E_{spring}(K,V).$$

Optimize simplicial complex K



Generate Legal Edge Operation

+ Optimize Vertices for fixed simplicial complex

Energy Function : Ideal pseudo-code

```
OptimizeMesh(K0,V0) {
    K := K0
    V := OptimizeVertexPositions(K0,V0)
    – Solve the outer minimization problem.
    repeat {
        (K',V') := GenerateLegalMove(K,V)
        V' = OptimizeVertexPositions(K',V')
        if E(K',V') < E(K,V) then
            (K,V) := (K',V')
        endif
    } until convergence
    return (K,V)
}

– Solve the inner optimization problem
–     E(K) = min_V E(K,V)
– for fixed simplicial complex K.
OptimizeVertexPositions(K,V) {
    repeat {
        – Compute barycentric coordinates by projection.
        B := ProjectPoints(K,V)
        – Minimize E(K,V,B) over V using conjugate gradients.
        V := ImproveVertexPositions(K,B)
    } until convergence
    return V
}

GenerateLegalMove(K,V) {
    Select a legal move K ⇒ K'.
    Locally modify V to obtain V' appropriate for K'.
    return (K',V')
}
```

Figure 3: An idealized pseudo-code version of the minimization algorithm.

First Stage

Second Stage

First stage : **Global Projection**

$$E_{dist}(K,V) = \sum_{i=1}^{n} d^2\left(\boldsymbol{x}_i\,, \phi_V(K)\right).$$

$$d^2\left(x_i, \phi_V(|K|)\right) = \min_{\boldsymbol{b}_i \in |K|} \|\boldsymbol{x}_i - \phi_V(\boldsymbol{b}_i)\|^2$$

$$E_{dist} = \sum_{i=1}^{n} \|\boldsymbol{x}_i - \phi_V(\boldsymbol{b}_i)\|^2$$

- Use K-d Tree BVH for a fast query of the closest triangle from a mesh for a data point $\boldsymbol{x}_i$.

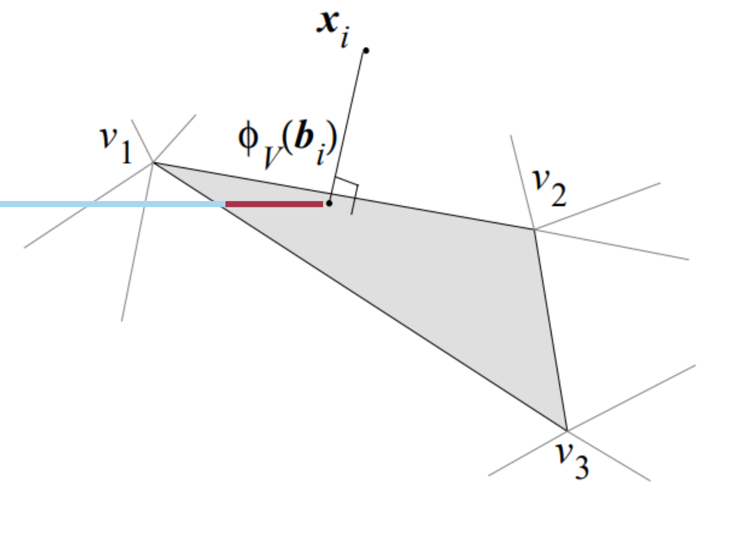
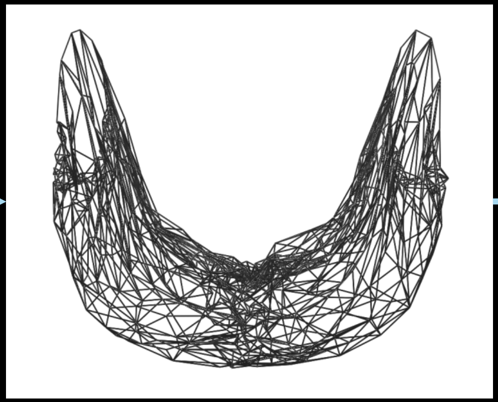


Figure 4: Distance of a point $\mathbf{x}_i$ from the mesh.

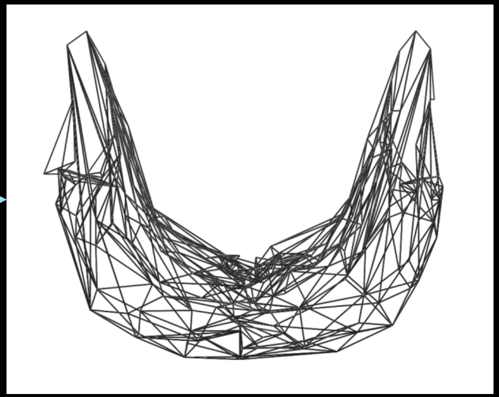
- find the closest vertex $\phi_V(\boldsymbol{b}_i)$ on the closest triangle, and get the barycentric coordinates $\boldsymbol{b}_i$ of the vertex.

First stage : **Local Projection**

Global Projection



Local Projection



Local Projection

- Use the global projection only in the **beginning** of the algorithm
- Use **only the local projection** after the beginning.
- Update the closest face and the closest vertex for only the **data points that are near the changing region**.

First stage : Solving Linear Least Squares in Local Context

$$E(K,V,B) = E_{dist}(K,V,B) + E_{spring}(K,V)$$

$$\overset{1)}{=} \sum_{i=1}^{n} \|x_i - \phi_V(b_i)\|^2 + \overset{2)}{\sum_{\{j,k\}\in K}} \kappa \|\boldsymbol{v}_j - \boldsymbol{v}_k\|^2$$

Linear Least Squares Problem $\min_{\boldsymbol{d}} \|C\boldsymbol{d} - \boldsymbol{f}\|^2$

Linear Least Squares Solution $\hat{\boldsymbol{d}} = (C^T C)^{-1} C^T \boldsymbol{f}$

Given a vertex V **to be optimized**, its adjacent vertices P_1 and P_2, and the barycentric coordinates (u, v, w) of the triangle,

Optimize the vertex V to minimize the energy function.

$$\min_V \|(uV + vP_1 + wP_2) - \mathbf{x}\|$$

$$uV + vP_1 + wP_2 - \mathbf{x} = [V \quad P_1 \quad P_2] \begin{bmatrix} u \\ v \\ w \end{bmatrix} - \mathbf{x}.$$

1)

$$\begin{bmatrix} u & 0 & 0 \\ 0 & u & 0 \\ 0 & 0 & u \end{bmatrix} \begin{bmatrix} V_x \\ V_y \\ V_z \end{bmatrix} - (\mathbf{x} - vP_1 - wP_2)$$

$$\kappa\|V_j - V_k\|^2 = \|\sqrt{\kappa}(V_j - V_k)\|^2$$

2)

$$\begin{bmatrix} \sqrt{\kappa} & 0 & 0 \\ 0 & \sqrt{\kappa} & 0 \\ 0 & 0 & \sqrt{\kappa} \end{bmatrix} \begin{bmatrix} V_{j,x} \\ V_{j,y} \\ V_{j,z} \end{bmatrix} - \sqrt{\kappa} \begin{bmatrix} V_{k,x} \\ V_{k,y} \\ V_{k,z} \end{bmatrix},$$

First stage : **Solving Linear Least Squares in Global Context**

- Solve by sparse conjugate-gradient method

- Solve by the nonlinear optimization method L-BFGS

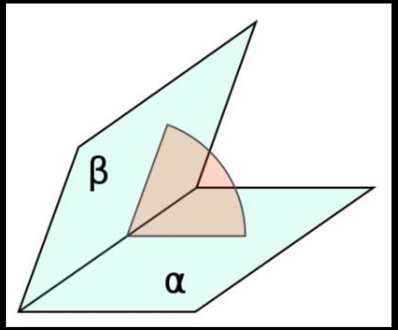
- d = number of data points
  m = d + number of edges of a mesh
  n = number of vertices

$$
\underbrace{\begin{bmatrix}
0 & \cdots & 0 & b_{0,u} & 0 & \cdots & b_{0,v} & 0 & \cdots & b_{0,w} & \cdots & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
0 & \cdots & b_{d-1,u} & \cdots & 0 & \cdots & \cdots & b_{d-1,v} & \cdots & b_{d-1,w} & \cdots & 0 \\
0 & \cdots & 0 & \cdots & \sqrt{\kappa} & \cdots & \cdots & 0 & \cdots & -\sqrt{\kappa} & \cdots & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
0 & \sqrt{\kappa} & \cdots & \cdots & \cdots & 0 & -\sqrt{\kappa} & \cdots & 0 & \cdots & \cdots & 0
\end{bmatrix}}_{\mathbb{R}^{m\times n}}
\underbrace{\begin{bmatrix}
\mathbf{v}_{0,x} & \mathbf{v}_{0,y} & \mathbf{v}_{0,z} \\
\vdots & \vdots & \vdots \\
\mathbf{v}_{n-1,x} & \mathbf{v}_{n-1,y} & \mathbf{v}_{n-1,z}
\end{bmatrix}}_{\mathbb{R}^{n\times 3}}
-
\underbrace{\begin{bmatrix}
\mathbf{x}_{0,x} & \mathbf{x}_{0,y} & \mathbf{x}_{0,z} \\
\vdots & \vdots & \vdots \\
\mathbf{x}_{d-1,x} & \mathbf{x}_{d-1,y} & \mathbf{x}_{d-1,z} \\
0 & 0 & 0 \\
\vdots & \vdots & \vdots \\
0 & 0 & 0
\end{bmatrix}}_{\mathbb{R}^{m\times 3}}
$$

First stage : More Details

- Dihedral Angle

  The new optimized vertex position is not accepted if its **minimum dihedral angle** for the optimized vertex with its adjacent vertices is **lower** than the angle before the optimization.



https://en.wikipedia.org/wiki/Dihedral_angle

- Residual Sum of Squares (RSS)

  They use their converted RSS as an energy evaluation for the second stage.

Linear Least Squares Problem

$$\min_{d} \|C\boldsymbol{d} - \boldsymbol{f}\|^2$$

$$\hat{\boldsymbol{d}} = (C^T C)^{-1} C^T \boldsymbol{f}$$

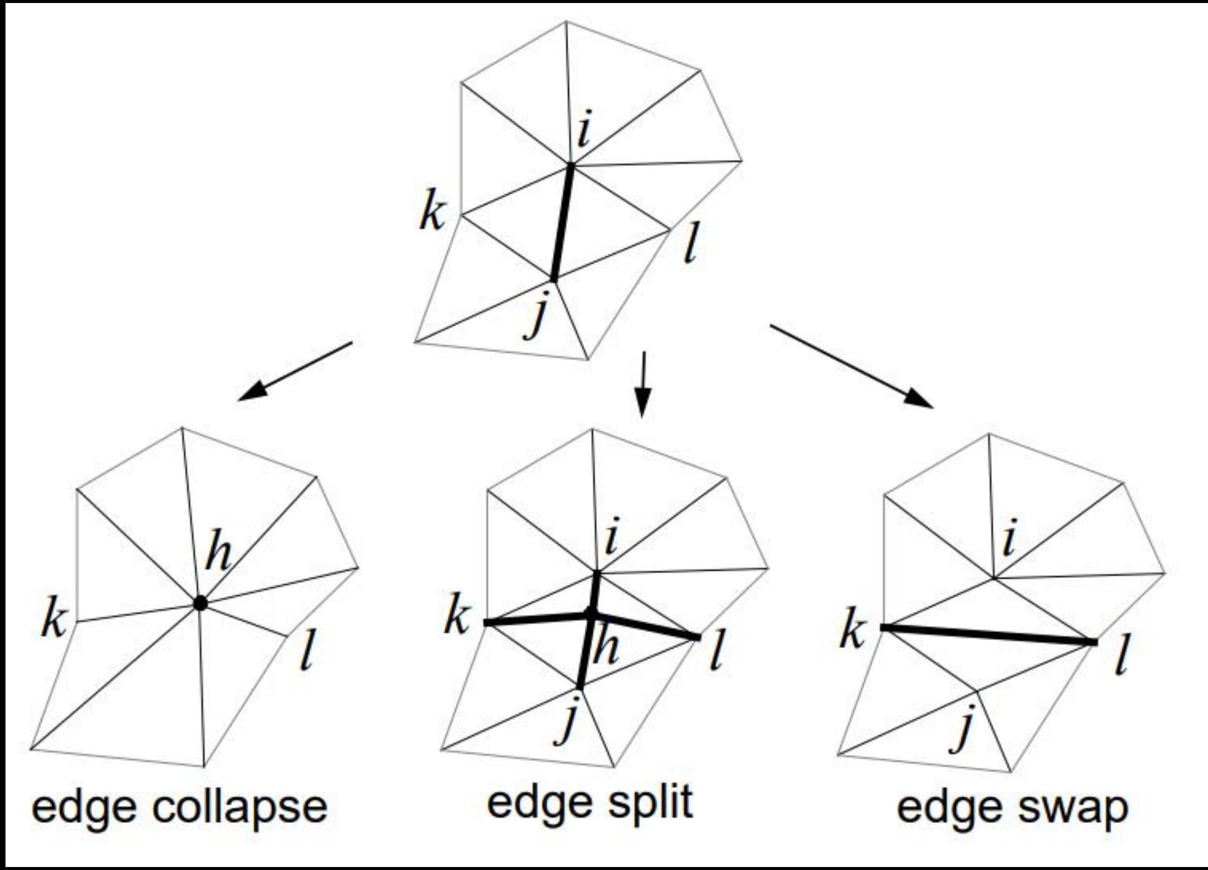
Normal RSS : $\|C\boldsymbol{d} - \boldsymbol{f}\|^2$

Their RSS : $\|\boldsymbol{f}\|^2 - \|C\boldsymbol{d}\|^2 \rightarrow \|\boldsymbol{f}\|^2 - \boldsymbol{d}^2\|C\|^2$

- Local Projection

  Run the local projection for the vertices and the data points related to the local optimization

Second stage : Legality of Edge Operations



complicated　　Always Legal　　Legal only if edge $\{k, l\}$ is not defined

Second stage : Legal Condition For Edge Collapse

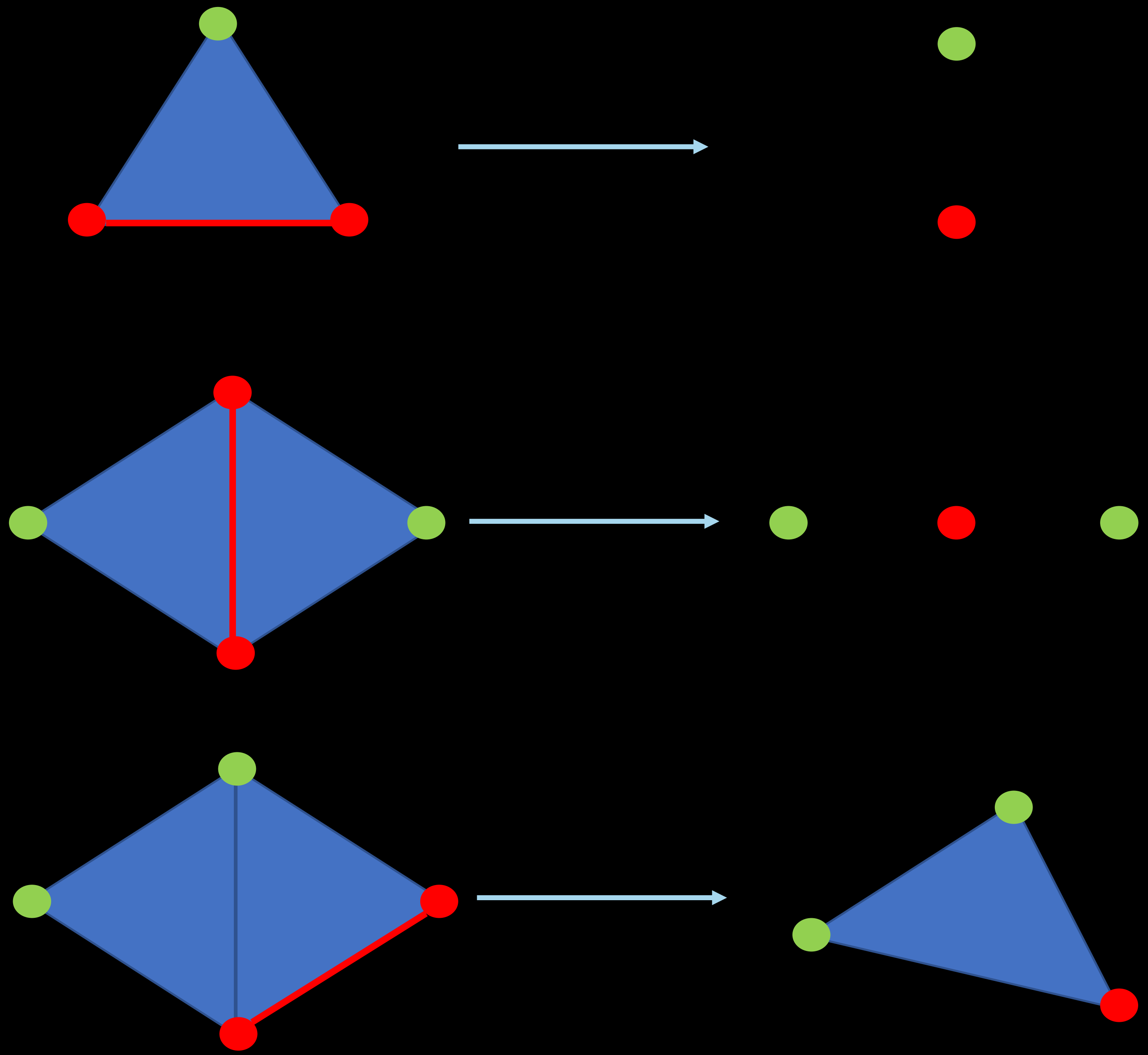
- For all vertices $\{k\}$ adjacent to both $\{i\}$ and $\{j\}$, $\{i, j, k\}$ is a face of K.

- If $\{i\}$ and $\{j\}$ are both boundary vertices, $\{i, j\}$ is a boundary edge.

- K has more than 4 vertices if neither $\{i\}$ nor $\{j\}$ are boundary vertices, or
  K has more than 3 vertices if either $\{i\}$ or $\{j\}$ are boundary vertices.

Second stage : Edge Collapse Cases for 3/4 vertices

- Edge Collapse on three vertices
- Edge Collapse on four vertices



Second stage : Algorithm

Edge Candidates ⟵ All edges from Mesh

Until the array of candidates is empty

Randomly select one edge e from the candidates. Remove it from the array

Try **Edge Collapse(e)**

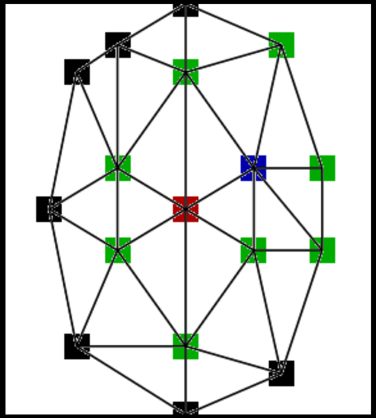
If the collapse fails, Try **Edge Split(e)**

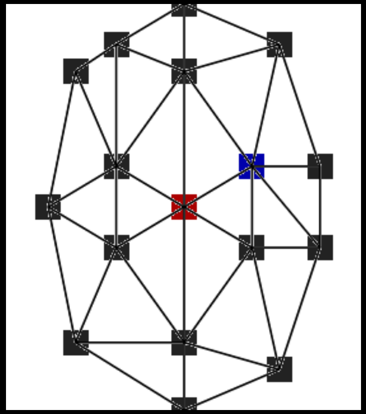
If the collapse and the split fail, Try **Edge Swap(e)**

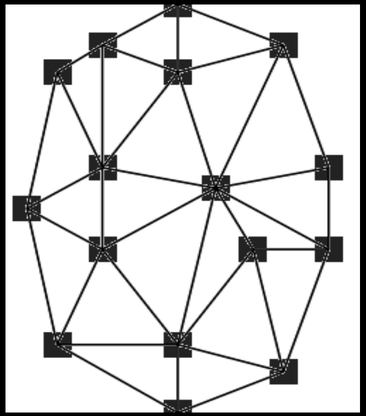
Add/Remove Edges for the candidates array by the edge operations

Second stage : Edge Collapsing Strategy

$$E(K,V) = E_{dist}(K,V) + E_{rep}(K) + E_{spring}(K,V).$$







Gather data points around the ring of an edge to be collapsed.

Evaluate E_{dist} and E_{spring} for the ring of the edge and the data points

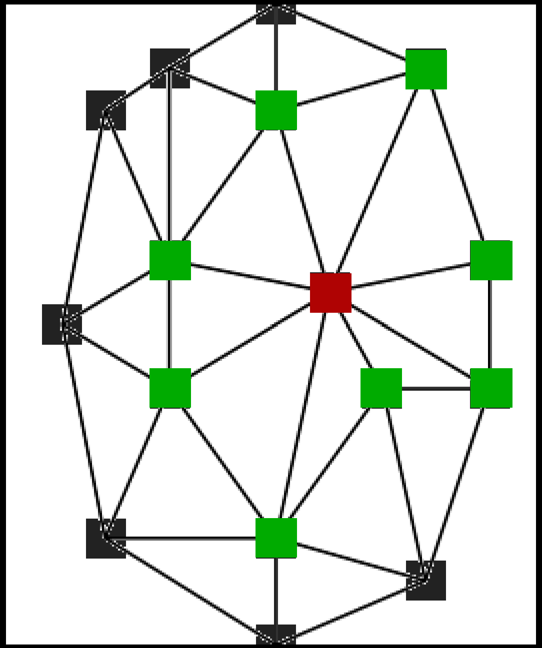
Find **the best collapsing point** between the vertices of the edge

Use the **first stage method** to evaluate the energy RSS for the collapsing position.

If $RSS - (E_{dist} + E_{spring} + c_{rep}) \geq 0$:
    Reject this operation
else:
    run edge collapsing operation

Second stage : Edge Collapsing Strategy in more details

- If the vertices of the edge to be collapsed has some boundary edges, then reject the edge operation.



- Reproject locally for the faces and the data points related to the edge collapsing

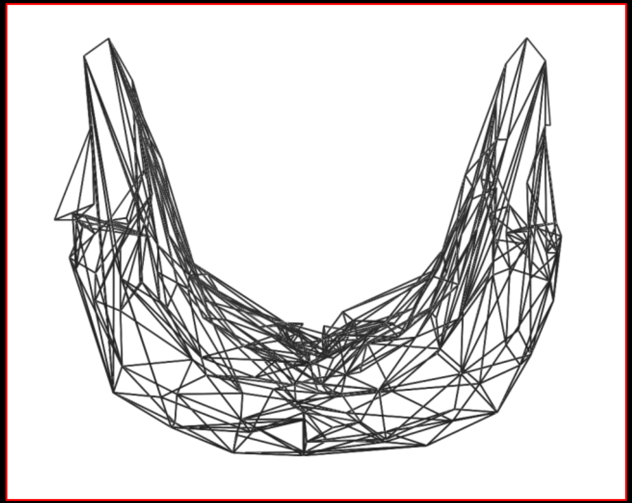
- Do the first stage method for **the residual vertex** after edge collapsing and its neighboring vertices

Simplification Strategy

```
float kappa_scheme[] = { 1e-2f, 1e-3f, 1e-4f, 1e-8f };
int scheme_size = sizeof(kappa_scheme) / sizeof(kappa_scheme[0]);
int scheme_index = 0;

while (scheme_index < scheme_size)
{
    oc->kappa = kappa_scheme[scheme_index];
    ++scheme_index;

    moi_local_fit(oc);

    moi_simplicies_fit(oc);

    moi_local_fit(oc);
}
```

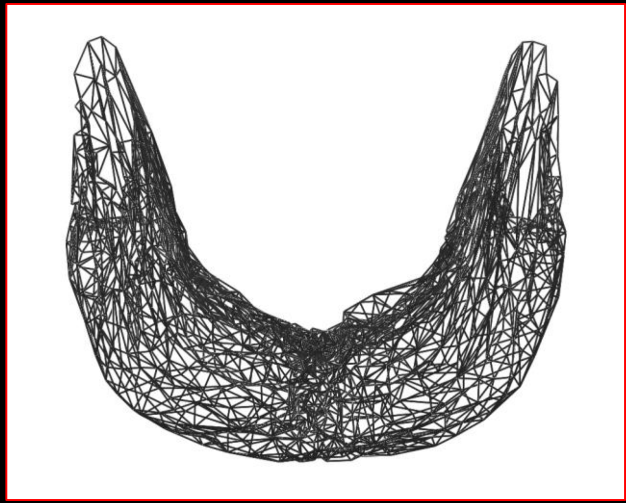
Result

Original Mesh M.
8,320 Vertices and **16,384** Faces



Optimized Mesh M' from M with $c_{rep} = 1e^{-3}$.
206 Vertices and **392** Faces

Optimized Mesh M'' from M with $c_{rep} = 1e^{-4}$.
565 Vertices and **1102** Faces



Optimized Mesh M''' from M with $c_{rep} = 1e^{-5}$.
1433 Vertices and **2813** Faces