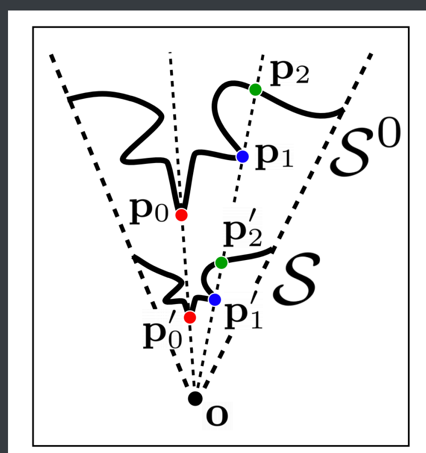
Appearance–Mimicking Surfaces

Inspired by bas-reliefs, appearance-mimicking surfaces are thin surfaces, or 2.5D images whose normals approximate the normals of a 3D shape. Given a viewpoint and per-vertex depth bounds, the algorithm proposed [1] finds a globally optimal surface that preserves the appearance of the target shape when observed from the designated viewpoint, while satisfying the depth constraints.

Problem Formulation

Let S^o be the original surface, and S be the deformed surface when observed from viewpoint $\mathbf{o}$. Each point $\mathbf{p}'$ of the deformed surface is constrained to stay on the ray emanating from the viewpoint in the direction of $\mathbf{p}$ (line $\mathbf{op}$).



The perceived difference $d(S, S^o, \mathbf{o})$ between S^o and S is measured by the sum of the L2 norm of their normals at each point:

$$d(S, S^o, \mathbf{o}) = \int_S \|\mathbf{n}^S_{\phi(\mathbf{p},\mathbf{o})} - \mathbf{n}^{S^o}_{\mathbf{p}}\|^2 d\mathbf{p} \tag{1}$$

Here, $\phi(\mathbf{p}, \mathbf{o}) = \mathbf{p}'$ on surface S. $\mathbf{n}^{S^o}_{\mathbf{p}}$ is the normal of S^o at point $\mathbf{p}$. Our goal is to minimize $d(S, S^o, \mathbf{o})$.

Discretization

When surface S is represented by a triangle mesh M, points p' on S are approximated by vertices $\mathbf{v}_i$. Each vertex $\mathbf{v}_i$ can be written as:

$$\mathbf{v}_i = \mathbf{o} + \|\mathbf{v}_i - \mathbf{o}\| \frac{\mathbf{v}_i - \mathbf{o}}{\|\mathbf{v}_i - \mathbf{o}\|} = \mathbf{o} + \lambda_i \hat{\mathbf{v}}_i \tag{2}$$

$\hat{\mathbf{v}}_i$ is the unit vector pointing in the direction of $\mathbf{ov_i}$. λ_i measures the distance between $\mathbf{o}$ and $\mathbf{v}_i$. This representation is convenient because M (deformed mesh) and M^o (original mesh) share the same set of $\hat{\mathbf{v}}_i$. Their differences are entirely expressed by λ_i and λ_i^o. Depth constraints for each vertex of M can be specified as a upper bound and a lower bound on λ_i:

$$\lambda_i^{min} \leq \lambda_i \leq \lambda_i^{max} \tag{3}$$

Using this representation, Eq. (1) can be discretized and linearized as:

$$\begin{aligned}
d(M, M^o, \mathbf{o}) = \sum_{i \in \mathbf{V}} w_i^2 A_i \|\mathbf{n}_i - \mathbf{n}_i^o\|^2 \\
&= \sum_{i \in \mathbf{V}} w_i^2 A_i \| \frac{(\mathbf{LV})_i}{H_i} - \frac{(\mathbf{L}^o\mathbf{V}^o)_i}{H_i^o} \|^2 \\
&= \sum_{i \in \mathbf{V}} w_i^2 A_i \| \frac{(\mathbf{LD}_\lambda \hat{\mathbf{V}})_i}{H_i} - \frac{(\mathbf{L}^o\mathbf{D}_{\lambda^o} \hat{\mathbf{V}})_i}{H_i^o} \|^2 \\
&= \sum_{i \in \mathbf{V}} w_i^2 A_i^o \| \frac{(\mathbf{L}^o\mathbf{D}_\lambda \hat{\mathbf{V}})_i}{H_i^o} - \frac{(\mathbf{L}^o\mathbf{D}_{\lambda^o} \hat{\mathbf{V}})_i}{H_i^o} \frac{\lambda_i}{\lambda_i^o} \|^2 \\
&= \sum_{i \in \mathbf{V}} w_i^2 A_i^o \| (\mathbf{L}^o\mathbf{D}_\lambda \hat{\mathbf{V}})_i - (\mathbf{L}^o\mathbf{D}_{\lambda^o} \hat{\mathbf{V}})_i \frac{\lambda_i}{\lambda_i^o} \|^2
\end{aligned}$$

A_i is the Voronoi area ssociated with $\mathbf{v}_i$ and can be obtained from the mass matrix coefficients. w_i are weights denoting the relative importance of $\mathbf{v}_i$. Visible vertices from the viewpoint are given more weight than occluded vertices. By default w_i are 1. $\mathbf{L}$ is the discrete laplace operator of mesh M. $\mathbf{D}_\lambda$ is a diagonal matrix with entries λ_i on the diagonal.

> We follow these conventions for notations:
>
> - If $\mathbf{X}$ is a property/operator for mesh M, then $\mathbf{X}^o$ is the corresponding property/operator for mesh M^o.
> - If $\mathbf{x}$ is a vector, then $\mathbf{D}_\mathbf{x}$ is a diagonal matrix with $\mathbf{x}$ on the diagonal.

Now our goal is to find λ such that it minimizes $d(M, M^o, \mathbf{o})$. To do this, we extract the unknown variable λ from $\mathbf{D}_\lambda$. The above equation can be further vectorized as:

$$d(M, M^o, \mathbf{o}) = \|\mathbf{D}_{\sqrt{A^o}} \mathbf{D}_w (\tilde{\mathbf{L}}^o \mathbf{D}_{\hat{\mathbf{V}}} - \mathbf{D}_{\mathbf{L}_\theta}) \mathbf{S} \lambda\|^2 = \|\mathbf{Q}\lambda\|^2 \tag{4}$$

$$\mathbf{L}_\theta = \mathbf{D}_{(\mathbf{S}\lambda_o)}{}^{-1} \tilde{\mathbf{L}}^o \mathbf{D}_{\hat{\mathbf{V}}} \mathbf{S} \lambda^o \tag{5}$$

We then construct all the components of matrix $\mathbf{Q} = \mathbf{D}_{\sqrt{A^o}} \mathbf{D}_w (\tilde{\mathbf{L}}^o \mathbf{D}_{\hat{\mathbf{V}}} - \mathbf{D}_{\mathbf{L}_\theta}) \mathbf{S}$.

Let $n = |\mathbf{V}^o|$,

- $\mathbf{D}_{\sqrt{A^o}}$ is a 3n x 3n matrix with the square root of mass matrix coefficients repeated 3 times (1 for each dimension) on the diagonal.

- $\mathbf{D}_w$ is a 3n x 3n matrix with the weight vector $\mathbf{w}$ repeated 3 times on the diagonal.
- $\tilde{\mathbf{L}}^o$, also 3n x 3n, is the Kronecker product between the cotangent matrix and 3 x 3 identity matrix: $\tilde{\mathbf{L}}^o = \mathbf{L}^o \otimes \mathbf{I}_3$
- $\mathbf{S}$ is a 3n x n selector matrix that associates each λ_i with the x, y, z coordinates of $\mathbf{v}_i$: $\mathbf{I}_n \otimes [1, 1, 1]^T$

Aside from depth constraints, we also need to fix the value of λ_k for one vertex $\mathbf{v}_k$ to obtain a unique solution. λ_k is a pre-calculated value b passed into the algorithm. For example, λ_k can be set to the average of its upper and lower bound.

We now have quadratic programming problem that can be solved using the libigl active set solver:

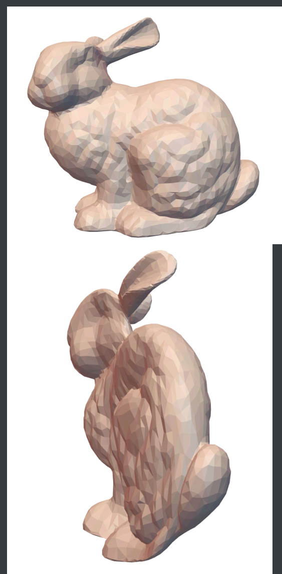
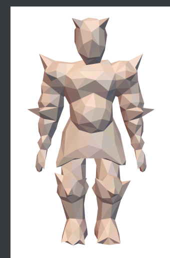
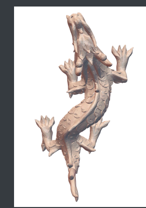
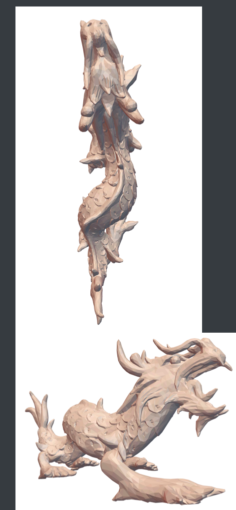
$$\min_{\lambda} ||\mathbf{Q}\lambda||^2$$

$$\text{subject to}$$
$$\lambda^{min} \leq \lambda \leq \lambda^{max}$$
$$\lambda_k = b$$

```
igl::active_set_params as;
Eigen::VectorXd lambda; // n x 1

// Q - D_A * D_w * (L_tilde0 * D_v_hat - D_L_theta) * S, 3n x n
// B - linear coefficients, set to 0
// b - index of lambda to be fixed
// Y - value of the fixed lambda
// Aeq, Beq, Aieq, Bieq - empty matrices
// lx, ux - upper and lower lambda bounds
igl::active_set(Q.transpose() * Q, B, b, Y, Aeq, Beq, Aieq, Bieq, lx, ux, as, lambda);
```

Demo

Original Mesh	Deformed Mesh	Original Mesh	Deformed Mesh	Original Mesh	Deformed Mesh
					

The example `main.cpp` deforms a mesh along the z-axis (front view).

Implementation Details

Getting $\mathbf{D}_{\sqrt{A^o}}$

- Dimension: 3n x 3n

First, we construct a mass matrix $\mathbf{M}$:

```
Eigen::SparseMatrix<double> M;
igl::massmatrix(V, F, igl::MASSMATRIX_TYPE_VORONOI, M);
```

Here, $\mathbf{M}$ is a diagonal matrix, in which the diagonal entry M_{ii} is the Voronoi area around $\mathbf{v}_i$ in the mesh [2]. We then take the diagonal entry for each vertex, take the square root, and repeat it 3 times (1 for each dimension). The resulting matrix $\mathbf{D}_{\sqrt{A^o}}$ should look like this:

$$\mathbf{D}_{\sqrt{A^o}} = \begin{bmatrix} \sqrt{M_{00}} & 0 & \dots & & & & & 0 \\ 0 & \sqrt{M_{00}} & & & & & & \vdots \\ & & \sqrt{M_{00}} & & & & & \\ & & & \sqrt{M_{11}} & & & & \\ & & & & \sqrt{M_{11}} & & & \\ & & & & & \sqrt{M_{11}} & & \\ \vdots & & & & & & \ddots & \\ 0 & 0 & \dots & & & 0 & \dots & \sqrt{M_{n-1,n-1}} \end{bmatrix} \tag{6}$$

Getting $\mathbf{D}_w$

- Dimension: 3n x 3n

Similar to $\mathbf{D}_{\sqrt{A^o}}$, we take the weight vector w of size n x 1 and repeat it along the diagonal:

$$\mathbf{D}_w = \begin{bmatrix} w_0 & 0 & \dots & & & & & 0 \\ 0 & w_0 & & & & & & \vdots \\ & & w_0 & & & & & \\ & & & w_1 & & & & \\ & & & & w_1 & & & \\ & & & & & w_1 & & \\ \vdots & & & & & & \ddots & \\ 0 & 0 & \dots & & 0 & & \dots & w_{n-1} \end{bmatrix} \tag{7}$$

Getting $\tilde{\mathbf{L}}^o$

- Dimension: 3n x 3n

First, we compute the cotangent Laplace-Beltrami operator $\mathbf{L}^o$:

```
Eigen::SparseMatrix<double> cot, M_inv, L;
igl::cotmatrix(V, F, cot);
igl::invert_diag(M, M_inv);
L = M_inv * cot;
```

$\tilde{\mathbf{L}}^o$ is the Kronecker product between the cotangent matrix and 3 x 3 identity matrix:

$$\tilde{\mathbf{L}}^o = \mathbf{L}^o \otimes \mathbf{I}_3 \tag{8}$$

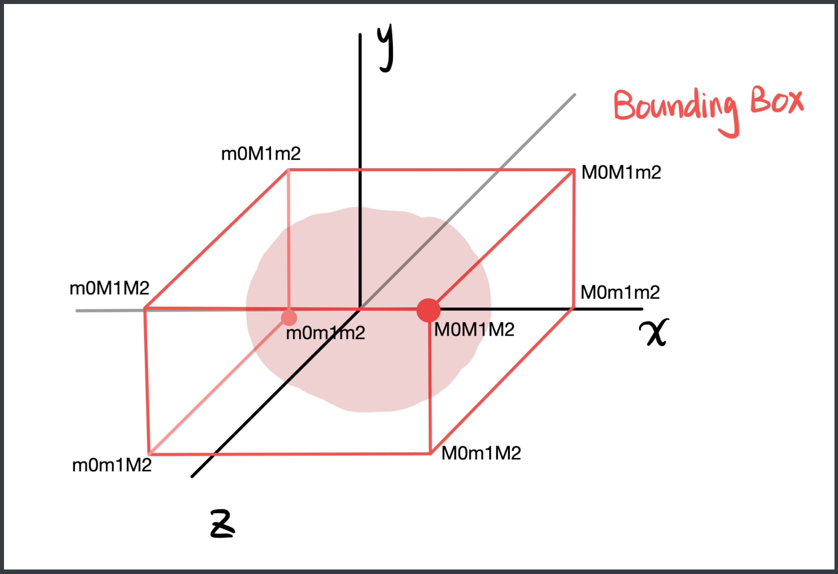
$$\tilde{\mathbf{L}}^o = \begin{bmatrix} \begin{bmatrix} \mathbf{L}^o_{00} & & \\ & \mathbf{L}^o_{00} & \\ & & \mathbf{L}^o_{00} \end{bmatrix} & \cdots & \begin{bmatrix} \mathbf{L}^o_{0,n-1} & & \\ & \mathbf{L}^o_{0,n-1} & \\ & & \mathbf{L}^o_{0,n-1} \end{bmatrix} \\ \vdots & \ddots & \vdots \\ \begin{bmatrix} \mathbf{L}^o_{n-1,0} & & \\ & \mathbf{L}^o_{n-1,0} & \\ & & \mathbf{L}^o_{n-1,0} \end{bmatrix} & & \begin{bmatrix} \mathbf{L}^o_{n-1,n-1} & & \\ & \mathbf{L}^o_{n-1,n-1} & \\ & & \mathbf{L}^o_{n-1,n-1} \end{bmatrix} \end{bmatrix} \tag{9}$$

Getting $\mathbf{S}$

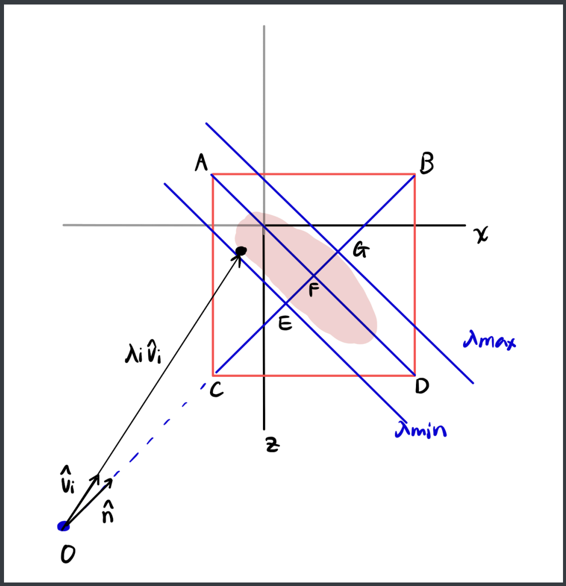
- Dimension: 3n x n
- $\mathbf{S}$ is a selector matrix that associates each λ_i with the x, y, z coordinates of $\mathbf{v}_i$: $\mathbf{I}_n \otimes [1,1,1]^T$

$$\begin{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \end{bmatrix} \tag{10}$$

Getting Bounds on λ



Bounds for λ_i varies for each vertex and each mesh. We first compute a bounding box [^3] for the mesh to help with customizing λ bounds.



The above figure shows how `main.cpp` deforms a mesh from a left viewpoint. We assume the mesh faces the positive z-axis and the bounding box is roughly square.

$$\lambda_{max} = |OG|$$
$$\lambda_{min} = |OE|$$
$$(\lambda_i \hat{\mathbf{v}}_i) \cdot \hat{\mathbf{n}}_i \leq \lambda_{max} \rightarrow \lambda_i \leq \lambda_{max}/(\hat{\mathbf{v}}_i \cdot \hat{\mathbf{n}}_i)$$
$$(\lambda_i \hat{\mathbf{v}}_i) \cdot \hat{\mathbf{n}}_i \geq \lambda_{min} \rightarrow \lambda_i \geq \lambda_{min}/(\hat{\mathbf{v}}_i \cdot \hat{\mathbf{n}}_i)$$
$$lowerbound = \lambda_{max}/(\hat{\mathbf{v}}_i \cdot \hat{\mathbf{n}}_i)$$
$$upperbound = \lambda_{min}/(\hat{\mathbf{v}}_i \cdot \hat{\mathbf{n}}_i)$$

Future Directions & Challenges

Allowing Disconnected Pieces

The paper introduces a vector μ that allows the depth range constraints to be discontinuous. Each element μ_g is a scaling factor for an independent group of vertices such that $\mu_g \lambda_i^{min} \leq \lambda_i \leq \mu_g \lambda_i^{max}$ holds for all vertices i in group g. $|\mu| =$ number of groups. The paper optimizes for both λ and μ:

$$\min_{\lambda, \mu} ||\mathbf{Q}\lambda||^2 + \alpha ||\mu||^2$$

$$\text{subject to}$$
$$\mathbf{C}_I [\lambda \;\; \mu]^T \leq \mathbf{d}$$
$$\mathbf{C}_E [\lambda \;\; \mu]^T = \mathbf{b}$$

α Is set to 10^{-7} in the paper. $\mathbf{C}_I$ are the inequality constraints. $\mathbf{C}_E$ are the equality constraints.

Combining λ and μ into one unknown vector $\mathbf{x}$, the input into the active set solver becomes:

```
// B - linear coefficients, set to 0
// b - index of lambda to be fixed
// Y - value of the fixed lambda
// Aeq, Beq, Aieq, Bieq - empty matrices
// lx, ux - upper and lower lambda and mu bounds
igl::active_set(F.transpose() * F, B, b, Y, Aeq, Beq, Aieq, Bieq, lx, ux, as, lambda);
```

where F is:

$$F = \begin{bmatrix} \mathbf{Q} & \mathbf{0} \\ \mathbf{0} & \sqrt{\alpha} \cdot \mathbf{I} \end{bmatrix} \tag{11}$$

The deformed vertices are retrieved by multiplying each λ_i with μ_g.

Unfortunately, I was not able to complete the implementation for this part. One challenge is that I am not sure what the inequality constraints ($\mathbf{C}_I$) are when μ is involved. Without μ, the inequality constraints are clear: $\lambda^{min} \leq \lambda \leq \lambda^{max}$. Then I could conveniently set `lx` = λ^{min}, `ux` = λ^{max}. However, the paper does not explicitly mention the constraints on μ.

- If there are no constraints, the energy minimization encourages μ to take on a magnitude of 0.
- If μ is constrained by $\mu_g \lambda_i^{min} \leq \lambda_i \leq \mu_g \lambda_i^{max}$, I am having trouble re-formulating this into $\mathbf{C}_I [\lambda \ \ \mu]^T \leq \mathbf{d}$ because both μ_g and λ_i are unknowns.

Other Challenges

The paper is very detailed in describing how to construct $\mathbf{Q}$, which greatly helped me in my implementation. However, the description for the constraints are less detailed. Aside from $\mathbf{C}_I$, I had trouble finding $\mathbf{C}_E$, $\mathbf{d}$, and $\mathbf{b}$ as well.

$$\begin{aligned} \mathbf{C}_I [\lambda \ \ \mu]^T \leq \mathbf{d} \\ \mathbf{C}_E [\lambda \ \ \mu]^T = \mathbf{b} \end{aligned} \tag{12}$$

I guessed from the line "the scale invariance introduces rank deficiency in the optimization, but can be fixed by constraining the λ_i of a single vertex i", that $\mathbf{C}_E [\lambda \ \ \mu]^T = \mathbf{b}$ means $\lambda_i = b$ for a pre-defined value b.

References

1. Christian Schuller, Daniele Panozzo, Olga Sorkine-Hornung, *Appearance-Mimicking Surfaces*, 2014 ↩

2. Mark Meyer, Mathieu Desbrun, Peter Schröder and Alan H. Barr, Discrete Differential-Geometry Operators for Triangulated 2-Manifolds, 2003. ↩