Lab 5 - Smoothing

Geometry Processing (GPR)

1 Iterative smoothing

Triangle mesh of n vertices represented as $\mathcal{M} = (\mathcal{G}, \mathcal{P})$ where:

- $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ models the *mesh graph*.
  - $\mathcal{V} = \{i \mid 1 \leq i \leq n\}$ represents its vertices.
  - $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ represents its set of edges, and thus its *connectivity* or *toplogy*.
- $\mathcal{P} \in \mathbb{R}^{n \times 3}$ contains the positions of the vertices, and thus its *geometry*.

Laplacians can be defined in several ways, one a generalization of the 1D version.

$$\delta(p_i) = \frac{1}{|\mathcal{N}_i|} \sum_{j \in \mathcal{N}_i} p_j - p_i$$

where $\mathcal{N}_i$ represents the 1-ring vertex neighbors:

$$\mathcal{N}_i = \{j \mid (i, j) \in \mathcal{E}\}$$

We apply the laplacians by updating the vertex positions:

$$p_i' = p_i + \lambda \cdot \delta(p_i), \qquad \lambda \in [0, 1]$$

To avoid volume loss we use the bilaplacian operator:

$$\begin{aligned} p_i' &= p_i + \lambda \cdot \delta(p_i) \\ p_i'' &= p_i' - \lambda \cdot \delta(p_i') \end{aligned}$$

Even better (and based on signal theory), we can apply Taubin's $\lambda - \mu$ operator:

$$\begin{aligned} p_i' &= p_i + \lambda \cdot \delta(p_i) \\ p_i'' &= p_i' + \mu \cdot \delta(p_i') \end{aligned}$$

where λ and μ are connected by the formula:

$$\frac{1}{\lambda} + \frac{1}{\mu} \approx 0.1 \implies \mu = \frac{\lambda}{0.1 \cdot \lambda - 1}$$

which for $\lambda > 0$ implies that $\mu < 0$.

2 Matrix form

The uniform laplacian can also be expressed and applied in matrix form. It is usually separated into the product of two matrices: the *weak laplacian matrix* and its corresponding *mass matrix*.

The weak laplacian matrix $\mathbf{C}$ is defined as:

$$(\mathbf{C})_{ij} = \begin{cases} 1 & , i \neq j \wedge (i,j) \in \mathcal{E} \\ -|\mathcal{N}_1| & , i = j \\ 0 & , otherwise \end{cases}$$

The mass matrix for the uniform laplacian is:

$$\mathbf{M} = \begin{pmatrix} |\mathcal{N}_1| & 0 & \cdots & 0 \\ 0 & |\mathcal{N}_2| & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & |\mathcal{N}_n| \end{pmatrix}$$

where $|\mathcal{N}_i|$ is the number of neighbors of vertex i.

Then the laplacian matrix $\mathbf{L}$ is:

$$\mathbf{L} = \mathbf{M}^{-1}\mathbf{C}$$

and can be applied to all vertices as a matrix product:

$$\mathbf{P}' = (\mathbf{I} + \mathbf{L}) \cdot \mathbf{P} = \mathbf{P} + \mathbf{LP}$$

if $\mathbf{P} \in \mathbb{R}^{n \times 3}$ contains all the vertices as rows. Thus matrix $\mathbf{LP}$ contains the laplacian update vectors.

3 Global smoothing

The versions of the laplacian we have seen take several iterations to smooth a mesh. There is an alternative way to perform smoothing, known as *global smoothing*.

The main idea is that we want the laplacian at each vertex to be as close to zero as possible:

$$\mathbf{LP} \approx \mathbf{0}$$

The problem is that the linear system $\mathbf{LP} = \mathbf{0}$ has the trivial solution $\mathbf{0}$. The way to avoid this is to constraint some of the vertices to either remain at their current position or be as close to it as possible.

Let us assume that the vertices we want to constrain correspond to the last indices. If we have vertex positions $\mathcal{P} = \{\mathbf{p_i}\}_{1 \leq i \leq n}$, then we want to preserve the position of any vertex $\mathbf{p_j}$ such that $m < j \leq n$.

Thus, we want to compute new positions $\mathbf{p_i}'$, where:

$$\begin{aligned} \delta(\mathbf{p_i}') &\approx 0 \qquad & 1 \leq i \leq m \\ \mathbf{p_i}' &= \mathbf{p_i} \qquad & m < i \leq n \end{aligned}$$

So for unconstrained vertices we use their laplacian equation, and for constrained ones we just fix them at their initial position. The first m rows of $\mathbf{L}$ (matrix $\mathbf{L_1}$) contain the laplacian equations of the unconstrained vertices, so:

$$\left[\begin{array}{c|c} \mathbf{L_1} & \\ \hline \mathbf{0} & \mathbf{I} \end{array}\right] \cdot \mathbf{P}' = \left[\begin{array}{c} 0 \\ \vdots \\ 0 \\ \hline \mathbf{p_{m+1}} \\ \vdots \\ \mathbf{p_n} \end{array}\right]$$

The matrix is square, so the system should be solvable, and the unconstrained vertices will take the shape of a membrane. This is why the laplacian is also sometimes called the *membrane energy*.

An alternative comes from the knowledge that the bilaplacian results from the least squares minimization of the laplacian. To exploit this fact we build a linear system that preserves all the laplacian equations and adds the constraints' equations:

$$\left[\begin{array}{c|c} \mathbf{L} & \\ \hline \mathbf{0} & \mathbf{I} \end{array}\right] \cdot \mathbf{P'} = \left[\begin{array}{c} 0 \\ \vdots \\ 0 \\ \hline \mathbf{p_{m+1}} \\ \vdots \\ \mathbf{p_n} \end{array}\right]$$

Now the matrix on the left side is not square, so we have to apply least squares to solve it. The unconstrained part of the surface will bend more smoothly. This is why the bilaplacian is sometimes known as the *thin plate energy*.