Notes - Theory session 6

Geometry Processing (GPR)

1 Smoothing

1.1 Introduction

To reduce noise in a mesh or a point cloud we need to apply smoothing algorithms.

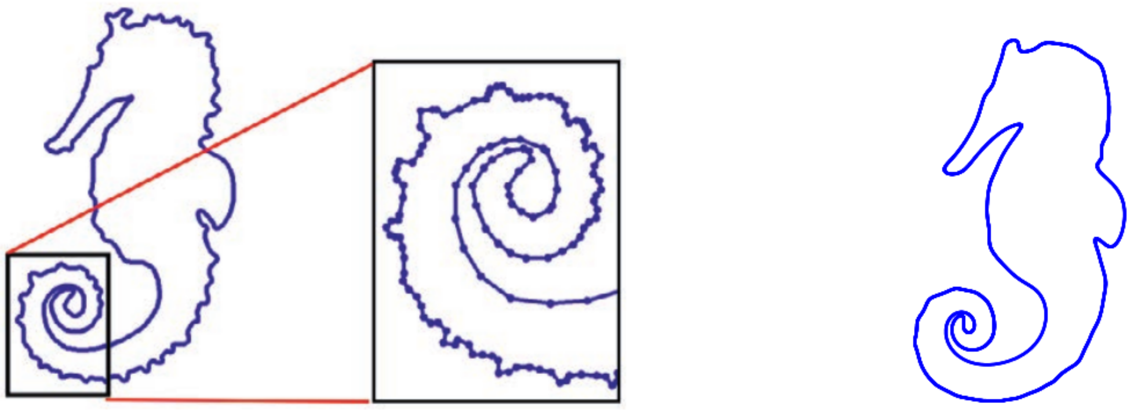


Figure 1: Smoothing a curve.

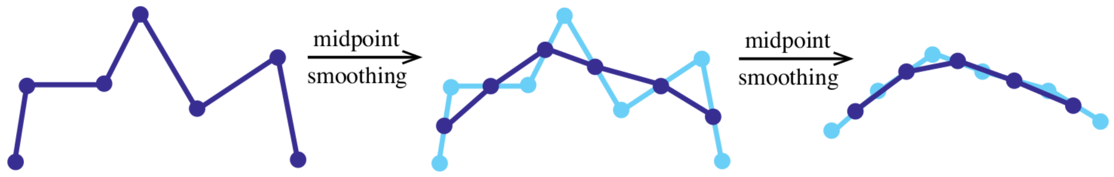
One of the simplest is known as the *midpoint smoothing* algorithm, which can be applied to polygonals. It substitutes the polygonal for a new one where every edge generates a vertex at its midpoint location, and every vertex transforms into an edge. For an edge between vertices v_{i-1} and v_i it generates a vertex at coordinates $(v_{i-1} + v_i)/2$.

Two applications of midpoint smoothing result into *laplacian smoothing*. Here a vertex remains a vertex of the result and its coordinates are:

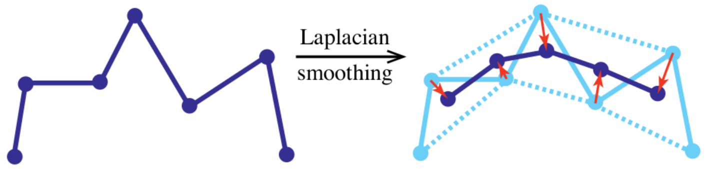
$$
\begin{aligned}
v_i' &= \frac{1}{2} \cdot \left(\frac{v_{i-1} + v_i}{2} \right) + \frac{1}{2} \cdot \left(\frac{v_i + v_{i+1}}{2} \right) = \\
&= \frac{1}{4} v_{i-1} + \frac{1}{2} v_i + \frac{1}{4} v_{i+1}
\end{aligned}
$$

and the displacement vector from its previous position to its "smoothed" position is called the laplacian of vertex v_i:

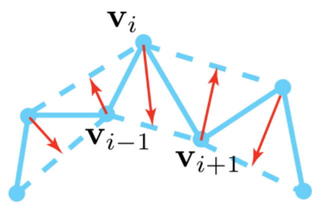
$$\delta(v_i) = v_i' - v_i = \frac{1}{4}v_{i-1} + \frac{1}{2}v_i + \frac{1}{4}v_{i+1} - v_i =$$
$$= \frac{1}{4}(v_{i-1} + v_{i+1}) - \frac{1}{2}v_i$$



(a) Two steps of midpoint smoothing.



(b) Laplacian smoothing.



(c) 1D Laplacians (red).

Figure 2: Midpoint and laplacian smoothing.

Laplacians can be applied to a full dataset simultaneously in matrix form. As an example, for a 1D periodic signal:

$$\mathbf{x}' = \mathbf{S} \cdot \mathbf{x} \qquad \mathbf{S} = \begin{pmatrix} 1/2 & 1/4 & 0 & \cdots & 0 & 1/4 \\ 1/4 & 1/2 & 1/4 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1/2 & 1/4 \\ 1/4 & 0 & 0 & \cdots & 1/4 & 1/2 \end{pmatrix}$$

which can also be expressed using a laplacian update matrix L:

$$\mathbf{S} = \mathbf{I} + \frac{1}{2} \cdot \mathbf{L} \qquad \mathbf{L} = \begin{pmatrix} -1 & 1/2 & 0 & \cdots & 0 & 1/2 \\ 1/2 & -1 & 1/2 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \\ 0 & 0 & 0 & \cdots & -1 & 1/2 \\ 1/2 & 0 & 0 & \cdots & 1/2 & -1 \end{pmatrix}$$

Matrix $\mathbf{L}$ is called the *1D discrete Laplacian operator*.

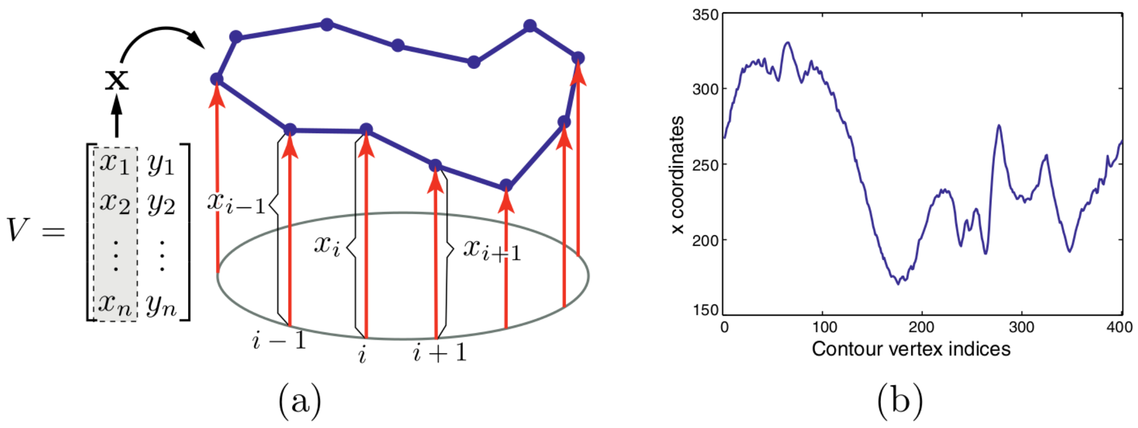


Figure 3: 1D signal.

1.2 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:

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

A more general definition allows weighting:

$$\delta(p_i) = \sum_{j \in \mathcal{N}_i} w_{ij} p_j - p_i, \qquad w_{ij} = \frac{\omega_{ij}}{\sum_{k \in \mathcal{N}_i} \omega_{ik}}$$

with the following popular weight choices:

$$\begin{aligned} Uniform \rightarrow &\quad \omega_{ij} = 1 \\ Cotangent \rightarrow &\quad \omega_{ij} = \cot \alpha_{ij} + \cot \beta_{ij} \end{aligned}$$

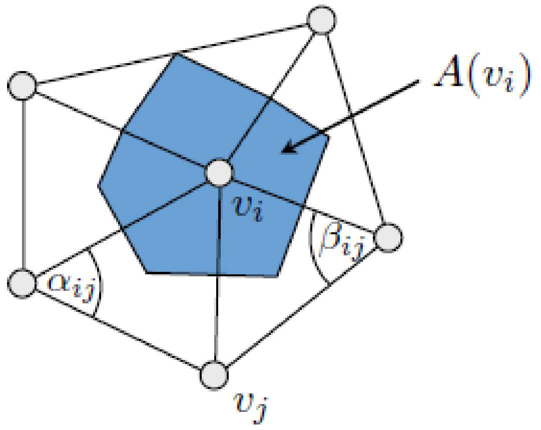


Figure 4: Cotangent weights.

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 $\mu < 0$.

The 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:

$$(\mathbf{C})_{ij} = \begin{cases} \omega_{ij} & , i \neq j \land (i,j) \in \mathcal{E} \\ -\sum_{k \in \mathcal{N}(i)} \omega_{ik} & , i = j \\ 0 & , otherwise \end{cases}$$

where:

$$\begin{aligned} \text{Uniform laplacian} \quad &\rightarrow \quad \omega_{ij} = 1 \\ \text{Cotangent laplacian} \quad &\rightarrow \quad \omega_{ij} = \frac{1}{2} \cdot (\cot \alpha_{ij} + \cot \beta_{ij}) \end{aligned}$$

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.

While for the cotangent:

$$\mathbf{M} = \begin{pmatrix} A_1 & 0 & \cdots & 0 \\ 0 & A_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & A_n \end{pmatrix}$$

where A_i is the area of influence of vertex i:

$$A_i = \frac{1}{8} \cdot \sum_{j \in \mathcal{N}_i} (\cot \alpha_{ij} + \cot \beta_{ij}) \cdot \|p_i - p_j\|^2$$

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.

The bilaplacian can also be applied in matrix form:

$$\mathbf{P}' = (\mathbf{I} - \mathbf{L}) \cdot (\mathbf{I} + \mathbf{L}) \cdot \mathbf{P} = (\mathbf{I} - \mathbf{L}^2) \cdot \mathbf{P} = \mathbf{P} - \mathbf{L}^2 \cdot \mathbf{P}$$

which means that the bilaplacian update vectors are contained in $-\mathbf{L}^2 \cdot \mathbf{P}$.

1.3 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.

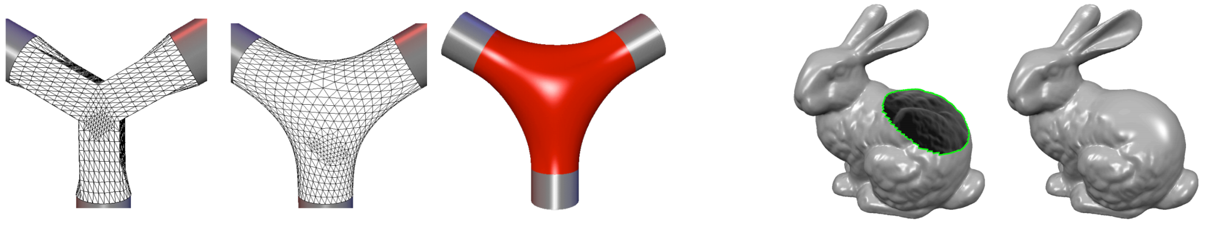


Figure 5: Applications of constrained global smoothing.

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

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

$$\begin{array}{ll} \delta(\mathbf{p_i}') \approx 0 & \quad 1 \le i \le m \\ \mathbf{p_i}' = \mathbf{p_i} & \quad m < i \le n \end{array}$$

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:

$$\left[\begin{array}{c|c} \multicolumn{2}{c}{\mathbf{L_1}} \\ \hline \mathbf{0} & \mathbf{I} \end{array}\right] \cdot \mathbf{P}' = \left[\begin{array}{c} 0 \\ \vdots \\ 0 \\ \hline \mathbf{p_{m+1}} \\ \vdots \\ \mathbf{p_n} \end{array}\right]$$

The matrix is square, so the system should be solvable, and the unconstrained vertices will take the shape of a membrane (see Figure 6). This is why the laplacian is also sometimes called the *membrane energy*.

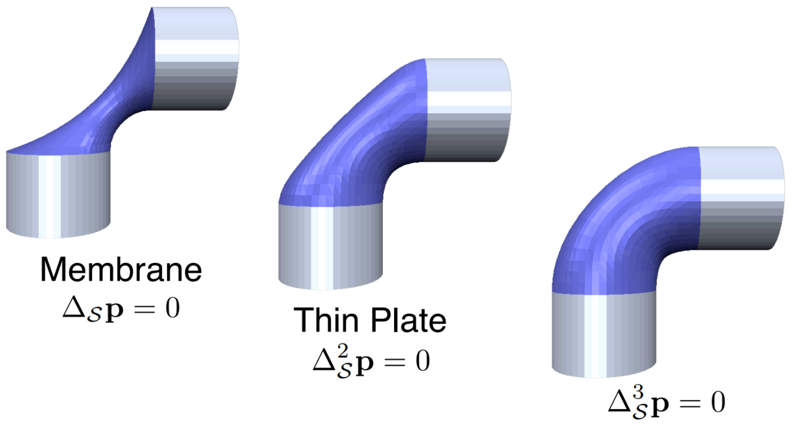

Membrane
$\Delta_{\mathcal{S}}\mathbf{p} = 0$
Thin Plate
$\Delta^2_{\mathcal{S}}\mathbf{p} = 0$
$\Delta^3_{\mathcal{S}}\mathbf{p} = 0$

Figure 6: Membrane vs Thin Plate 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:

$$\left[\begin{array}{c|c} \multicolumn{2}{c}{\mathbf{L}} \\ \hline \mathbf{0} & \mathbf{I} \end{array}\right] \cdot \mathbf{P}' = \left[\begin{array}{c} 0 \\ \vdots \\ 0 \\ \hline \mathbf{p_{m+1}} \\ \vdots \\ \mathbf{p_n} \end{array}\right]$$

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* (see Figure 6).