600.657: Mesh Processing

Chapter 4

Outline

- Review
- Diffusion Flow
- Fairing
- Signal Processing

Review

Recall:

The Laplace operator measures the divergence of the gradient of a function about a point.

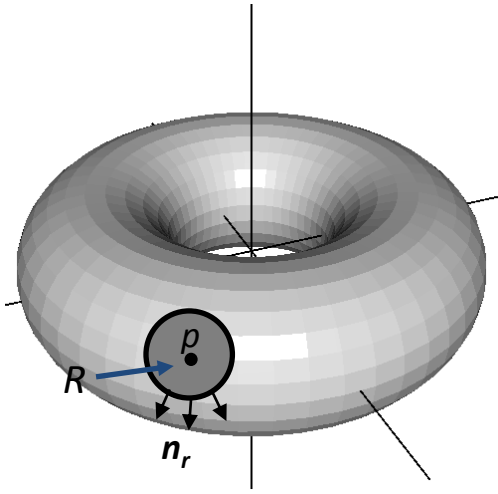
$$\Delta_S f = \mathrm{div}_S(\nabla_S f)$$

Review

Recall:

The divergence about a point is the measure of change across the boundary of a small region surrounding that point:

$$\operatorname{div}_S \nabla_S f(p) = \lim_{R \to p} \frac{1}{\operatorname{Area}(R)} \int_{\partial R} \langle n_r(q), \nabla_S f(q) \rangle dq$$



Review

Recall:

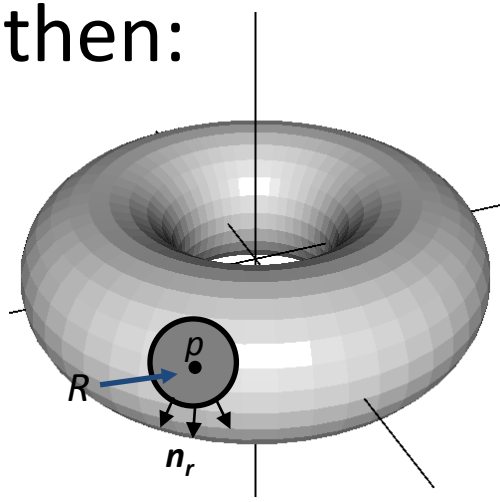
The divergence about a point is the measure of change across the boundary of a small region surrounding that point:

$$\operatorname{div}_S \nabla_S f(p) = \lim_{R \to p} \frac{1}{\operatorname{Area}(R)} \int_{\partial R} \langle n_r(q), \nabla_S f(q) \rangle dq$$

If we make *R* be a "disk with radius *r*" then:

$$\operatorname{div}_S \nabla_S f(p) \approx \lim_{R \to p} \frac{1}{\operatorname{Area}(R)} \int_{\partial R} \frac{f(q) - f(p)}{|q - p|} dq$$

$$= \lim_{R \to p} \frac{1}{\operatorname{Area}(R)} \frac{1}{r} |\partial R| \left(Avg_{\partial R}(f) - f(p) \right)$$

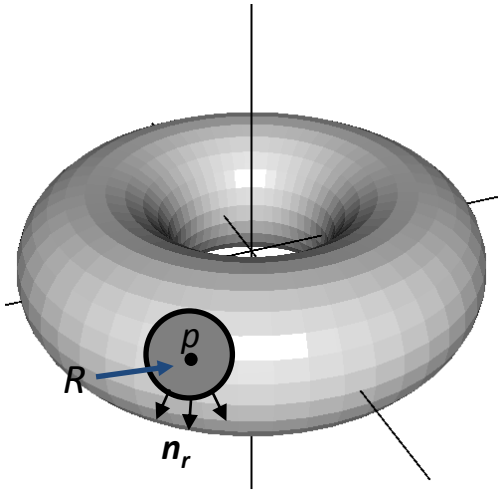


Review

Recall:

$$\operatorname{div}_S \nabla_S f(p) \approx \lim_{R \to p} \frac{1}{\operatorname{Area}(R)} \frac{1}{r} |\partial R| \left(Avg_{\partial R}(f) - f(p) \right)$$

So, the Laplacian is a measure of the difference between the value of a function at a point and the average value of its neighbors.



Outline

- Review
- Diffusion Flow
- Fairing
- Signal Processing

Diffusion Flow

Motivation:

To smooth a function, we would like to make the value at a point be closer to the value of its neighbors.

Diffusion Flow

Motivation:

To smooth a function, we would like to make the value at a point be closer to the value of its neighbors.

That is, we would like to reduce the difference between the value at a point and the average value at its neighbors (the Laplacian).

Diffusion Flow

Approach:

Given a function $f(p)$, define a family of functions $f_t(p)$ where:

1. The function defined at time t=0 is the original function.
2. The change in the function values over time is proportional to the Laplacian:

$$f_0(p) = f(p)$$

$$\frac{\partial}{\partial t} f_t(p) = \lambda \Delta_S f_t(p)$$

Diffusion Flow

$$f_0(p) = f(p)$$

$$\frac{\partial}{\partial t} f_t(p) = \lambda \Delta_S f_t(p)$$

Discretizing (Explicit):

The function at time step $\delta(k+1)$ is set so that the difference between the function at $\delta(k+1)$ and δk is proportional to the Laplacian at δk:

$$f_{\delta(k+1)}(p) - f_{\delta k}(p) = \lambda \delta \Delta_S f_{\delta k}(p)$$

Diffusion Flow

$$f_0(p) = f(p)$$

$$\frac{\partial}{\partial t} f_t(p) = \lambda \Delta_S f_t(p)$$

Discretizing (Explicit):

The function at time step $\delta(k+1)$ is set so that the difference between the function at $\delta(k+1)$ and δk is proportional to the Laplacian at δk:

$$f_{\delta(k+1)}(p) - f_{\delta k}(p) = \lambda \delta \Delta_S f_{\delta k}(p)$$

$$f_{\delta(k+1)}(p) = \left(Id + \lambda \delta \Delta_S\right) f_{\delta k}(p) = \left(Id + \lambda \delta \Delta_S\right)^{k+1} f(p)$$

Diffusion Flow

$$f_0(p) = f(p)$$

$$\frac{\partial}{\partial t} f_t(p) = \lambda \Delta_S f_t(p)$$

Discretizing (Implicit):

The function at time step $\delta(k+1)$ is set so that the difference between the function at $\delta(k+1)$ and δk is proportional to the Laplacian at $\delta(k+1)$:

$$f_{\delta(k+1)}(p) - f_{\delta k}(p) = \lambda \delta \Delta_S f_{\delta(k+1)}(p)$$

Diffusion Flow

$$f_0(p) = f(p)$$

$$\frac{\partial}{\partial t} f_t(p) = \lambda \Delta_S f_t(p)$$

Discretizing (Implicit):

The function at time step $\delta(k+1)$ is set so that the difference between the function at $\delta(k+1)$ and δk is proportional to the Laplacian at $\delta(k+1)$:

$$f_{\delta(k+1)}(p) - f_{\delta k}(p) = \lambda \delta \Delta_S f_{\delta(k+1)}(p)$$

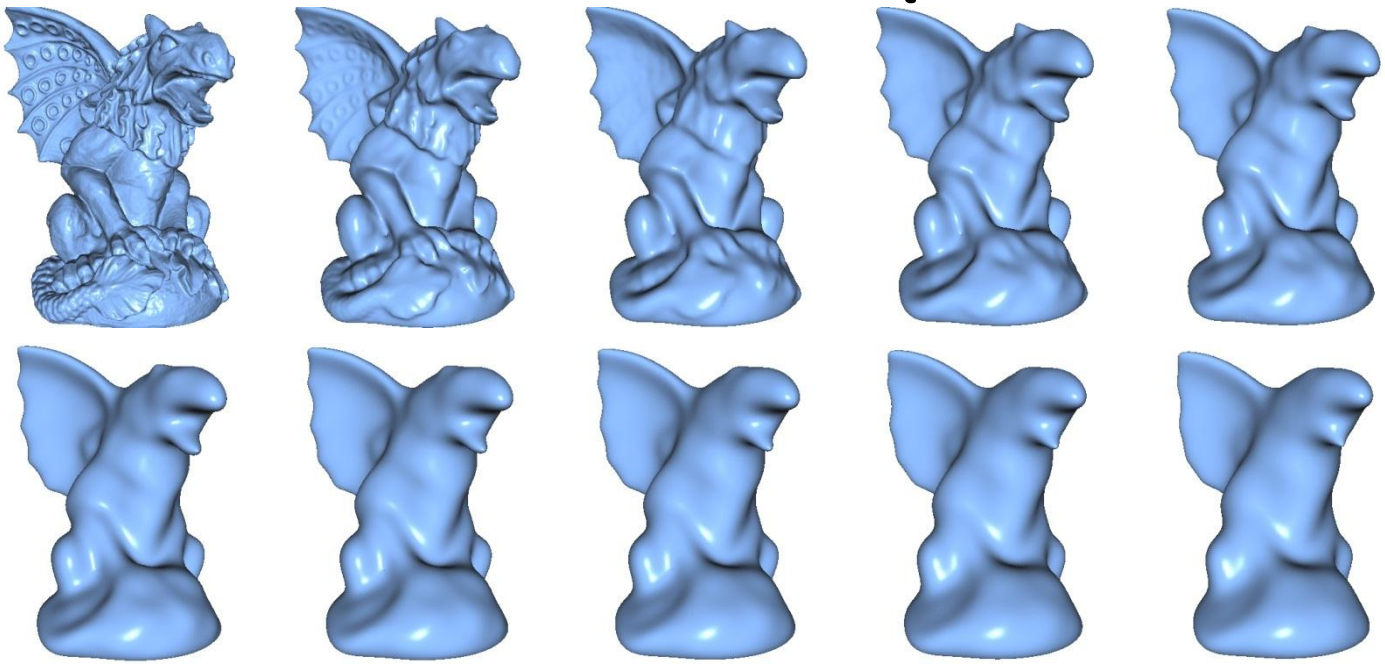
$$f_{\delta(k+1)}(p) = \left(Id - \lambda \delta \Delta_S\right)^{-1} f_{\delta k}(p) = \left(Id - \lambda \delta \Delta_S\right)^{-(k+1)} f(p)$$

Mean-Curvature Flow

We can applying this approach to the function giving the position of points on the surface:

$$f(p)=x$$

to get the smoothed surface S_t at time t.



Mean-Curvature Flow

We can applying this approach to the function giving the position of points on the surface:

$$f(p)=x$$

to get the smoothed surface S_t at time t.

Note:

Since the surface is evolving with time, we also update the Laplace operator at each time step.

$$f_0(p)=x$$

$$\frac{\partial}{\partial t} f_t(p) = \lambda \Delta_{S_t} f_t(p)$$

Mean-Curvature Flow

Using the fact that the Laplacian of this function is the mean curvature vector:

$$\Delta_S x = -2H\boldsymbol{n}$$

this surface flow evolves points by moving them in the direction of the normal, by a value proportional to the mean-curvature:

$$\frac{\partial}{\partial t} f_t(p) = -2\lambda H\boldsymbol{n}(p)$$

The explicit per-vertex update of the resulting so-called Laplacian smoothing is

$$\mathbf{x}_i \leftarrow \mathbf{x}_i + h\lambda\Delta\mathbf{x}_i$$

The above flow equation is therefore is also called the mean curvature flow [Desbrunet al. 99]

Laplacian smoothing with the uniform Laplacian tries to move each vertex to the barycenter of its one-ring neighbors. This smooths the mesh geometry and at the same time also leads to a **tangential relaxation** of the triangulation (see Figure 4.6). Depending on the application, this can be a desired feature (e.g., in isotropic remeshing, Chapter 6) or a disadvantage.

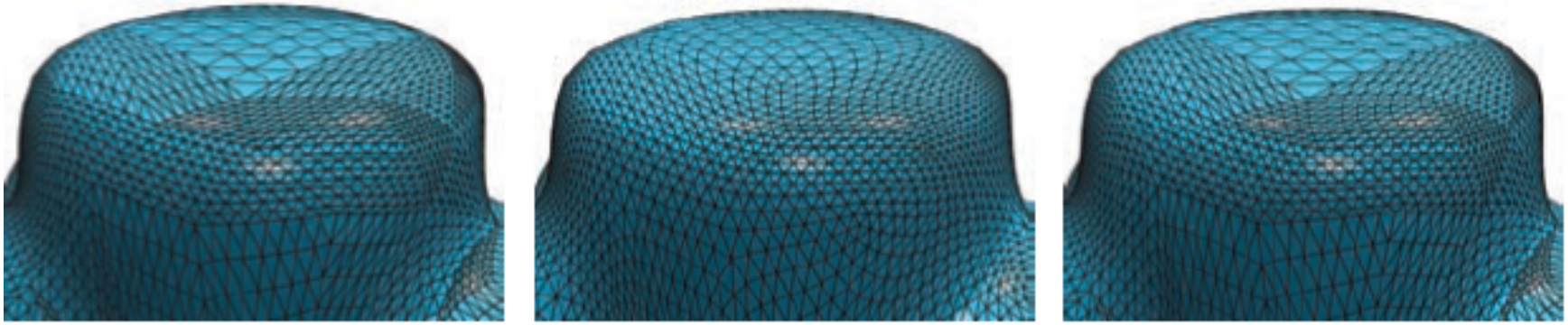


Figure 4.6. Smoothing the object on the left (ten iterations) using the uniform Laplacian also regularizes the triangulation (center), whereas the cotangent Laplacian preserves the triangle shapes (right).

higher-order Laplacian flows $\partial \mathbf{f} / \partial t = \lambda \Delta^k \mathbf{f}$ can also be used where discretizations of higher-order Laplacians are computed recursively as

$$\Delta^k f = \Delta\left(\Delta^{k-1} f\right)$$

Higher-order flows are more expensive to compute since they depend on a larger stencil of vertices, but they provide better low-pass filtering properties [Desbrun et al. 99].

In practice, bi-Laplacian smoothing (k = 2) is a good trade-off between computational efficiency and smoothing quality. When the smoothing is applied only locally, the bi-Laplacian smoothing leads to a C1 smooth blend between the smoothed and the fixed region, whereas the Laplacian smoothing achieves C0 boundary smoothness only.

Outline

- Review
- Diffusion Flow
- Fairing
- Signal Processing

denoising v.s. fairing: noise vs detail or oscillations

--[Moreton and Sequin 92, Welch and Witkin 92]

denoising v.s. fairing: noise vs detail or oscillations

--[Moreton and Sequin 92, Welch and Witkin 92]

The primary application of diffusion ow is to remove high frequency noise from a signal while preserving its low frequencies. In contrast, the goal of surface fairing is to compute shapes that are as smooth as possible. How to actually measure smoothness or fairness obviously depends on the application, but in general fair surfaces should follow the principle of simplest shape: the surface should be free of any unnecessary details or oscillations [Moreton and Sequin 92,Welch and Witkin 92].

Figure 4.7. Applications of surface fairing include constructing smooth blends between given surface parts (left) and filling holes with smooth patches (right).

Alternative Derivation

Given a function *f* on a surface *S*, we can measure the Dirichlet Energy of the surface as the sum of gradient magnitudes:

$$E(f) = \int_S \left\| \nabla_S f(p) \right\|^2 dp$$

It is a linear approximation of membrane energy, which measures the area of the surface S & it is highly nonlinear: square root of the determinant of the (already nonlinear) first fundamental form.

linearize membrane energy by replacing 1st fundamental form by 1st-order partial derivatives

$$\tilde{E}_{\mathrm{M}}(\mathbf{x}) \to \min \quad \Leftrightarrow \quad \Delta \mathbf{x}(u,v) = 0 \ \text{ for } \ (u,v) \in \Omega,$$

again subject to boundary constraints on $\partial\Omega$.

Euler-Lagrange equation

Surface Fairing

$$\Delta_S f(p) = 0$$

For (connected) surfaces without boundary, the minimizer is not particularly interesting, since the only functions with zero Laplacian are the constant functions.

Surface Fairing

$$\Delta_S f(p) = 0$$

For (connected) surfaces without boundary, the minimizer is not particularly interesting, since the only functions with zero Laplacian are the constant functions.

However, for surfaces with boundary, we can prescribe the values at the boundary ∂S, and obtain non-constant functions in the interior by solving:

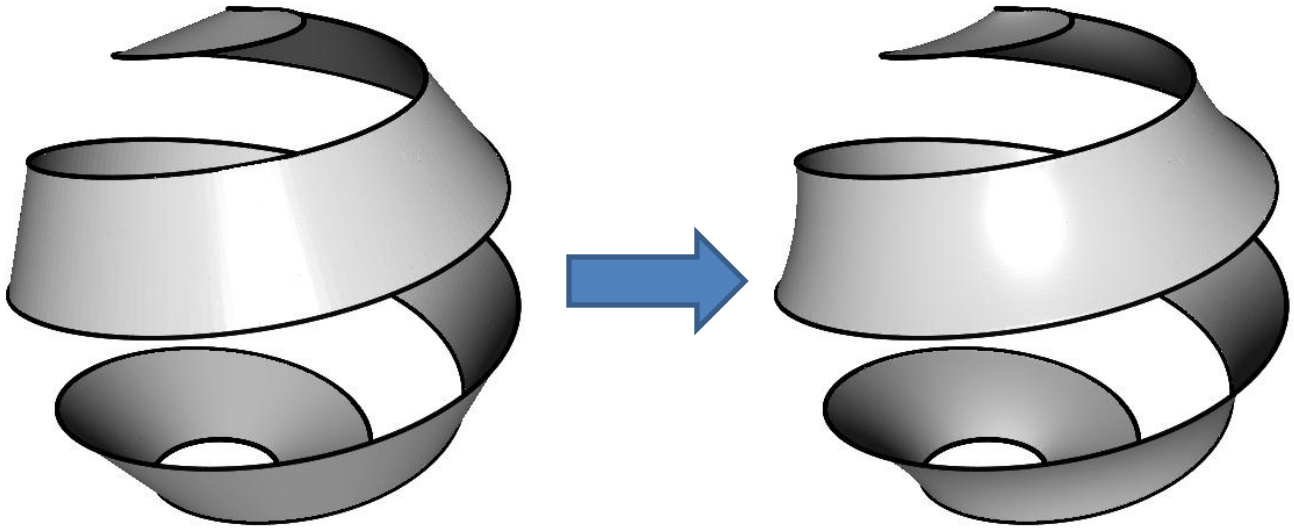
$$\Delta_S f(p) = 0$$
$$f(s) = \phi(s) \quad \text{for all} \quad s \in \partial S$$

Surface Fairing

$$\Delta_S f(p) = 0$$

$$f(s) = \phi(s) \quad \text{for all} \quad s \in \partial S$$

When ϕ:$\partial S \rightarrow \boldsymbol{R}^3$ fixes the 3D positions of points on the boundary, we get a function mapping the old surface to an almost minimal surface with the same boundary.



Surface Fairing

$$\Delta_S f(p) = 0$$

$$f(s) = \phi(s) \quad \text{for all} \quad s \in \partial S$$

When $\phi: \partial S \rightarrow \boldsymbol{R}^3$ fixes the 3D positions of points on the boundary, we get a function mapping the old surface to an almost minimal surface with the same boundary.

The surface is not quite minimal because the mean-curvature of the new surface is only zero with respect to the old surface's Laplacian.

Surface Fairing

$$\Delta_S f(p) = 0$$

$$f(s) = \phi(s) \quad \text{for all} \quad s \in \partial S$$

When $\phi: \partial S \rightarrow \boldsymbol{R}^3$ fixes the 3D positions of points on the boundary, we get a function mapping the old surface to an almost minimal surface with the same boundary.

The surface is not quite minimal because the mean-curvature of the new surface is only zero with respect to the old surface's Laplacian.

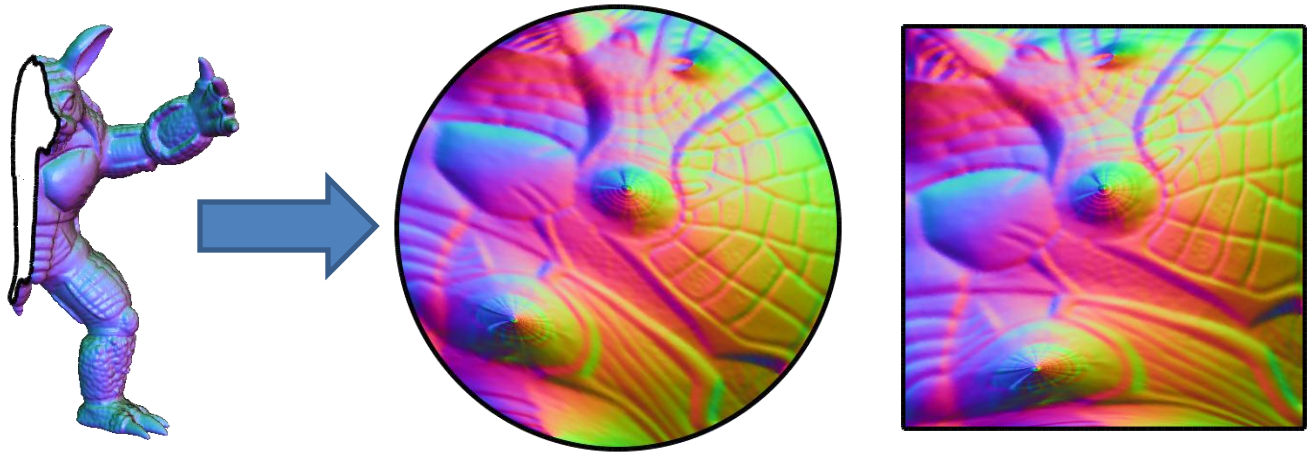
Iterating, we do get to the minimal surface.

Surface Fairing

$$\Delta_S f(p) = 0$$

$$f(s) = \phi(s) \quad \text{for all} \quad s \in \partial S$$

When $\phi: \partial S \rightarrow \boldsymbol{R}^2$ prescribes the desired 2D position of points on the boundary, we get a mapping from the surface to the 2D plane.



Surface Fairing

Generalizations:

Although we derived mean-curvature flow by minimizing the gradient norms:

$$E(f) = \int_S \left\| \nabla_S f(p) \right\|^2 dp$$

Surface Fairing

Generalizations:

Although we derived mean-curvature flow by minimizing the gradient norms:

$$E(f) = \int_S \left\| \nabla_S f(p) \right\|^2 dp$$

We can also try to minimize higher order derivatives, like the change in gradients:

$$E(f) = \int_S \left\| \nabla_S \nabla_S f(p) \right\|^2 dp$$

Surface Fairing

Generalizations:

Although we derived mean-curvature flow by minimizing the gradient norms:

$$E(f) = \int_S \left\| \nabla_S f(p) \right\|^2 dp$$

We can also try to minimize higher order derivatives, like the change in gradients:

$$E(f) = \int_S \left\| \nabla_S \nabla_S f(p) \right\|^2 dp$$

or the change of the change in gradients…

Surface Fairing

Generalizations:

When minimizing the gradient norms, we obtained the flow:

$$\frac{\partial}{\partial t} f_t(p) = \lambda \Delta_S f_t(p)$$

Surface Fairing

Generalizations:

When minimizing the gradient norms, we obtained the flow:

$$\frac{\partial}{\partial t} f_t(p) = \lambda \Delta_S f_t(p)$$

When minimizing the change in gradients we get:

$$\frac{\partial}{\partial t} f_t(p) = \lambda \Delta_S \Delta_S f_t(p)$$

Surface Fairing

Generalizations:

When minimizing the gradient norms, we obtained the flow:

$$\frac{\partial}{\partial t} f_t(p) = \lambda \Delta_S f_t(p)$$

When minimizing the change in gradients we get:

$$\frac{\partial}{\partial t} f_t(p) = \lambda \Delta_S \Delta_S f_t(p)$$

and so on.

Surface Fairing

Generalizations:

If we specify the boundary values, we obtain the minimizer of the Dirichlet Energy by solving:

$$\Delta_S f(p) = 0$$

$$f(s) = \phi(s) \quad \text{for all} \quad s \in \partial S$$

Surface Fairing

Generalizations:

If we specify the boundary values, we obtain the minimizer of the Dirichlet Energy by solving:

$$\Delta_S f(p) = 0$$
$$f(s) = \phi(s) \quad \text{for all} \quad s \in \partial S$$

When minimizing the change in gradients, we obtain the minimizer by solving:

$$\Delta_S \Delta_S f(p) = 0$$
$$f(s) = \phi(s) \qquad \text{for all} \quad s \in \partial S$$
$$\nabla_S f(s) = \vec{V}(s) \quad \text{for all} \quad s \in \partial S$$

Surface Fairing

Generalizations:

Applying this to the function giving the 3D positions of points on the surface gives a way to smoothly fill in holes:

membrane surface ($\Delta \mathbf{x} = 0$, left), thin-plate surface ($\Delta^2 \mathbf{x} = 0$, center), and minimum variation surface ($\Delta^3 \mathbf{x} = 0$, right).

If the goal is to minimize curvature instead of surface area, we start from the nonlinear *thin-plate energy*

$$E_{\mathrm{TP}}(\mathbf{x}) \;=\; \iint_{\Omega} \kappa_1^2 + \kappa_2^2 \,\mathrm{d}u\,\mathrm{d}v,$$

where κ_1 and κ_2 denote the principal curvatures. Linearization replaces curvatures by second derivatives, leading to

$$\tilde{E}_{\mathrm{TP}}(\mathbf{x}) \;=\; \iint_{\Omega} \|\mathbf{x}_{uu}\|^2 + 2\,\|\mathbf{x}_{uv}\|^2 + \|\mathbf{x}_{vv}\|^2 \,\mathrm{d}u\,\mathrm{d}v.$$

The corresponding Euler-Lagrange equation is $\Delta^2\mathbf{x}(u,v) = 0$ in Ω, with suitable C^1 boundary constraints prescribing positions $\mathbf{x}(u,v)$ and normals $\mathbf{n}(u,v)$ on $\partial\Omega$. Translated to a discrete triangle mesh, we get the linear bi-Laplacian system

$$\mathbf{L}^2\mathbf{x} \;=\; \mathbf{0}.$$

In the discrete case, it is typically easier to fix the positions of two rings of boundary vertices instead of prescribing positions and normals for one ring of boundary vertices [Kobbelt et al. 98b]. Both kinds of boundary constraints lead to (approximate) C1 boundary smoothness, as shown in previous figure (center).

Even higher-order fairness can be achieved by minimizing not curvature, but the variation of curvature

$$\iint_{\Omega} \left(\frac{\partial \kappa_1}{\partial \mathbf{t}_1} \right)^2 + \left(\frac{\partial \kappa_2}{\partial \mathbf{t}_2} \right)^2 \mathrm{d}u\,\mathrm{d}v, \tag{4.9}$$

where κ_1, κ_2 again denote principal curvatures and $\mathbf{t}_1, \mathbf{t}_2$ the corresponding principal curvature directions. The discrete approximation of these so-called *minimum variation surfaces* [Moreton and Séquin 92] can be computed by the sixth-order PDE $\Delta^3 \mathbf{x} = 0$ (see Figure 4.8 (right)).

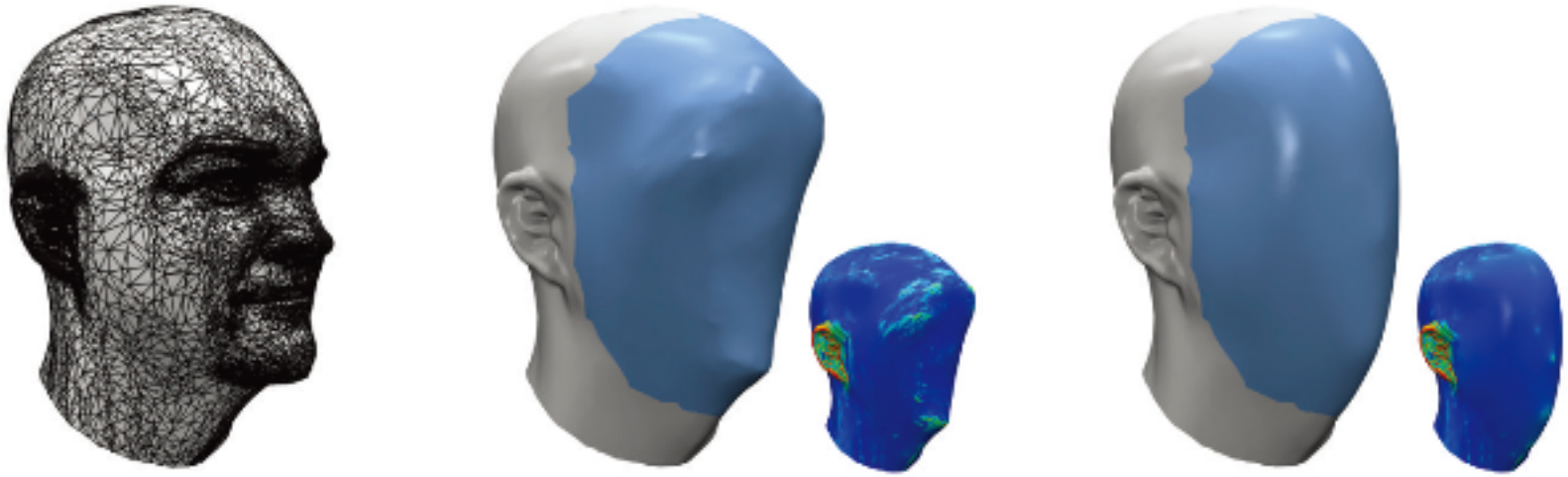


Figure 4.9. Comparison of different Laplace-Beltrami discretizations for solving $\Delta^2 \mathbf{x} = \mathbf{0}$: irregular input triangle mesh (left), uniform Laplacian (center), and cotangent Laplacian (right). The small images show the respective mean curvatures. (Model courtesy of Cyberware. Image taken from [Botsch and Sorkine 08]. ©2008 IEEE.)

Outline

- Review
- Diffusion Flow
- Fairing
- Signal Processing

goto Spectral Mesh Processing - Siggraph 2010 Course
instead of using this part for explaination

Discrete Dirichlet Energy

Recall:

To compute the Dirichlet Energy of a function *f* on *S*, we integrate the square norms of the gradient of *f*:

$$E(f) = \int_S \left\| \nabla_S f(p) \right\|^2 dp$$

Discrete Dirichlet Energy

$$E(f) = \int_S \left\| \nabla_S f(p) \right\|^2 dp$$

If *S* has no boundary, or we restrict ourselves to looking at functions that vanish on the boundary, we get:

$$E(f) = \int_S \left\| \nabla_S f(p) \right\|^2 dp = -\int_S f \cdot \Delta_S f \; dp$$

Discrete Dirichlet Energy

$$E(f) = \int_S \|\nabla_S f(p)\|^2 \, dp$$

If *S* has no boundary, or we restrict ourselves to looking at functions that vanish on the boundary, we get:

$$E(f) = \int_S \|\nabla_S f(p)\|^2 \, dp = -\int_S f \cdot \Delta_S f \; dp$$

Thus, if we represent the function $f=\{f_1,\ldots,f_n\}$ by its values on the vertices, and we set *L* to be the discrete (cotangent) Laplacian, we get:

$$E(f) = -f^t L f$$

Discrete Dirichlet Energy

$$E(f) = -f^t L f$$

Q: What is the most “energetic” function?

Discrete Dirichlet Energy

$$E(f) = -f^t L f$$

Q: What is the most "energetic", unit-norm function?

$$\arg\max_f = \frac{-f^t L f}{f^t f}$$

Discrete Dirichlet Energy

$$E(f) = -f^t L f$$

Q: What is the most "energetic", unit-norm function?

$$\arg\max_f = \frac{-f^t L f}{f^t f}$$

A: The maximizer of the energy is the eigenvector* of *L* with the largest (negative) eigenvalue.

*generalized eigenvector

Discrete Dirichlet Energy

$$E(f) = -f^t L f$$

Q: What is the most "energetic", unit-norm function?

$$\arg\max_f = \frac{-f^t L f}{f^t f}$$

A: The next maximizer of the energy is the eigenvector* of *L* with the next largest (negative) eigenvalue.

*generalized eigenvector

Discrete Dirichlet Energy

$$E(f) = -f^t L f$$

Computing all the eigenvectors* of the Laplace operator, we get a set of functions $\{f^1,\ldots,f^n\}$ with associated eigenvalues $\{-\lambda_1 \leq \ldots \leq -\lambda_n\}$ such that:

$$Lf^i = \lambda_i f^i$$

[Vallet et al. 2008]

*generalized eigenvector

Discrete Dirichlet Energy

$$E(f) = -f^t L f$$

Computing all the eigenvectors* of the Laplace operator, we get a set of functions $\{b^1,\ldots,b^n\}$ with associated eigenvalues $\{-\lambda_1 \leq \ldots \leq -\lambda_n\}$ such that:

$$Lb^i = \lambda_i b^i$$

Since the Laplace operator is symmetric and (negative) semi-definite, the eigenvectors are all orthogonal, and the eigenvalues are all negative.

*generalized eigenvector

Signal Processing

$$Lb^i = \lambda_i b^i$$

Definition:

The values λ_i are called the *natural frequencies* of the surface and the functions b^i are called the *manifold/mesh-harmonics*.

Signal Processing

$$Lb^i = \lambda_i b^i$$

Alternate Smoothing:

Given a function *f* defined on the mesh, we can express *f* as a linear combination of the manifold harmonics:

$$f = \sum_i f_i b^i$$

Signal Processing

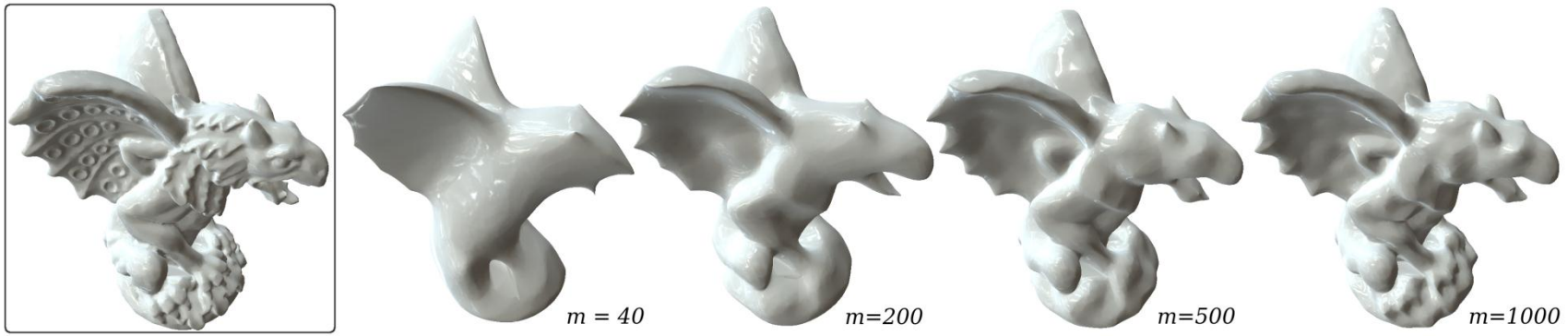
$$f = \sum_i f_i b^i$$

Alternate Smoothing:

So smoothing *f* corresponds to dampening the more energetic frequencies:

$$\text{Smooth}(f) = \sum_i w_i f_i b^i$$

where w_i is a set of weights that drops off with frequency (e.g. w_i=1/0 for small/large *i*.)



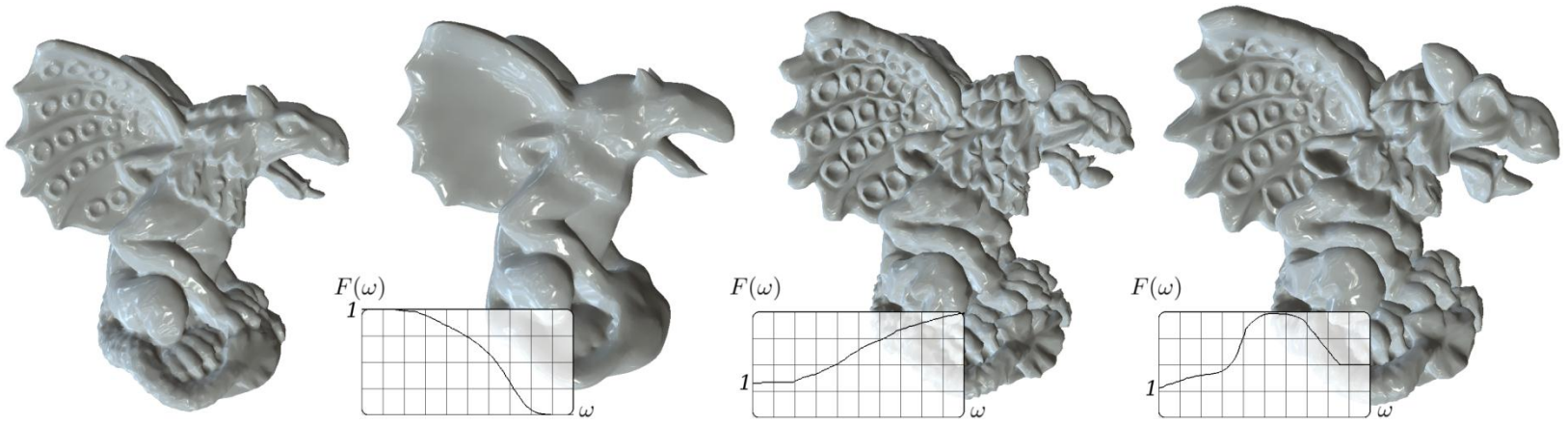
[Vallet et al. 2008]

Signal Processing

$$f = \sum_i f_i b^i$$

Alternate Smoothing:

Of course, we can also generalize the approach to a broader class of surface edits by selectively amplifying/dampening different frequencies:



[Vallet et al. 2008]

Signal Processing

$$f = \sum_i f_i b^i$$

Recall:

In performing diffusion flow with step size δ, we considered two different integration schemes:

- Explicit: $f(p) \leftarrow \left(Id + \delta\Delta_S\right) f(p)$
- Semi-Implicit: $f(p) \leftarrow \left(Id - \delta\Delta_S\right)^{-1} f(p)$

Signal Processing

$$f = \sum_i f_i b^i$$

Recall:

If f is an eigenvector of the Laplacian, with eigenvalue $-\lambda$, this gives:

- Explicit: $f(p) \leftarrow (1 - \delta\lambda) f(p)$
- Semi-Implicit: $f(p) \leftarrow \frac{1}{1 + \delta\lambda} f(p)$

Signal Processing

- Explicit: $f(p) \leftarrow (1-\delta\lambda) f(p)$
- Semi-Implicit: $f(p) \leftarrow \frac{1}{1+\delta\lambda} f(p)$

That is:

- Explicit: scales the -λ-th frequency by $(1-\delta\lambda)^k$.
- Implicit: scales the -λ-th frequency by $1/(1+\delta\lambda)^k$.

Signal Processing

- Explicit: $f(p) \leftarrow (1-\delta\lambda) f(p)$
- Semi-Implicit: $f(p) \leftarrow \frac{1}{1+\delta\lambda} f(p)$

That is, after *k* iterations:

- Explicit: scales the -λ-th frequency by $(1-\delta\lambda)^k$.
- Implicit: scales the -λ-th frequency by $1/(1+\delta\lambda)^k$.

Explicit: Smooth/converges for small time steps ($\delta\lambda<2$)
Implicit: Smooth/converges for all time steps.