1.7.1 Laplacian Smoothing
Theory

- Minimize energy functional
  - total curvature
    
    \[ E(S) = \int_S \kappa_1^2 + \kappa_2^2 \, dS. \]

  - estimate by polynomial-fitting
  - non-linear (very slow!)
Theory

• Minimize energy functional
  - membrane or thin-plate energy

\[ E_{\text{membrane}}(X) = \frac{1}{2} \int_{\Omega} X_u^2 + X_v^2 \, dudv \]

\[ E_{\text{thin plate}}(X) = \frac{1}{2} \int_{\Omega} X_{uu}^2 + 2X_{uv}^2 + X_{vv}^2 \, dudv. \]

- derivative is Laplacian

\[ L(X) = X_{uu} + X_{vv} \]

\[ L^2(X) = L \circ L(X) = X_{uuuu} + 2X_{uuvv} + X_{vvvv}. \]
Discrete Laplacian

The Laplacian Operator

\[ \Delta v_i = \sum_j w_{ij} (v_j - v_i) \]

for \( w_{ij} = 1/n \):

\[ v_i' = v_i + \lambda \Delta v_i \]
**Laplacian smoothing flow**

\[ P_{\text{new}} \leftarrow P_{\text{old}} + \lambda L(P_{\text{old}}) \]

Average of the vectors to neighboring vertices

Move each vertex

Jacobs University

Visualization and Computer Graphics Lab
Laplacian smoothing

• Move each vertex by the given formula
• Iterate the process for further smoothing
“Regular” diffusion

- Problem with umbrella operator
  - does not distinguish these cases:

- umbrella assumes
  regular parameterization
Umbrella problem

Initial mesh  Umbrella smoothing
Volume preservation

- Diffusion induces shrinkage!
  - Enforce exact volume preservation
    - Compute volume
    - Rescale after integration step(s)
- Incorporate other invariants
  - Volume, surface area, ...
  - Build directly into PDE
1.7.2 Non-shrinking Laplacian
Fourier analysis

\[ \Delta x_i = \sum_j w_{ij} (x_j - x_i) \quad Kx = -\Delta x \]

- Eigenvalues of \( K = I - W \) (FREQUENCIES)
  
  \[ 0 = k_0 \leq k_1 \leq \ldots \leq k_N \leq 2 \]

- Right eigenvectors of \( K \) (NATURAL VIBRATION MODES)
  
  \[ e_0, e_1, \ldots, e_N \]

Jacobs University

Visualization and Computer Graphics Lab
Fourier analysis

Geometry of low and high frequencies

\[ k_h e_{hi} = K e_{hi}' = - \sum_{j} w_{ij} (e_{hj} - e_{hi}) \]

- Low frequency
- High frequency
Fourier analysis

The Discrete Fourier Transform

- Eigenvectors form a basis of $N$-space
- Every signal can be written as a linear combination
  \[ x = \hat{x}_0 e_0 + \hat{x}_1 e_1 + \cdots + \hat{x}_N e_N \]
- Discrete Fourier Transform (DFT)
  \[ \hat{x} = (\hat{x}_0, \hat{x}_1, \ldots, \hat{x}_N)^t \]
Fourier analysis

- Polynomial Transfer Function

\[ x' = f(K)x \quad Kx = -\Delta x \]

- \( f(k) \) is a univariate polynomial
- \( f(K) \) is a matrix
- Eigenvectors of \( K \) and \( f(K) \) are the same
- Eigenvalues of \( f(K) \) are

\[ f(k_0), f(k_1), \ldots, f(k_N) \]
Fourier analysis

- After filtering
  \[ f(K)x = f(k_0)\hat{x}_0 e_0 + \cdots + f(k_N)\hat{x}_N e_N \]

- Evaluation of \( f(K)x \) based on matrix multiplication
- It does not require the computation of eigenvalues and eigenvectors (DFT)
- Low-Pass: need univariate polynomial \( f(k) \) such that
  \[
  \begin{align*}
  f(k_h) &\approx 1 & k_L &\leq k_{PB} \\
  f(k_h) &\approx 0 & k_L &> k_{PB}
  \end{align*}
  \]
Fourier analysis

Laplacian Smoothing is not Low-Pass

- After filtering
  \[ f(K)x = f(k_0)\hat{x}_0 e_0 + \cdots + f(k_N)\hat{x}_N e_N \]

- For Laplacian smoothing

\[ f(k_0) = 1 \]
\[ f(k_j) = (1 - \lambda k_j)^N \rightarrow 0 \quad j \neq 0 \quad 0 \leq \lambda < 1 \]
Non-shrinking Laplacian

- Minor modification of Laplacian smoothing algorithm
- Two Laplacian smoothing steps
- First shrinking step with positive factor
- Second unshrinking step with negative factor
- Use inverted parabola as transfer function

\[ f(k) = ((1 - \mu k)(1 - \lambda k))^{N/2} \quad \text{with} \quad -\mu > \lambda > 0 \]
Alternative: Length-scaled Laplacian

• Length-scale weighted operator:

\[ L_P(M) = \frac{1}{\sum l_j} \sum_j \frac{Q_j - P}{l_j} \]

(Fujiwara operator)

- scale-dependent operator

• But: stability issues
  - tiny explicit time step: \( dt < \frac{l_{\text{min}}^2}{\lambda} \)
  - implicit integration a must!
Non-shrinking

Umbrella smoothing  Scale-dependent smoothing
**Solver**

- Explicit Euler iterations:
  - (Diffusion Eqn)

\[
\dot{M} = \lambda L(M)
\]

\[
M_{t+dt} = M_t + \lambda L(M_t)dt
\]

- Strict stability requirements
- Small time steps
Stability

Explicit Euler scheme vs. Implicit Euler scheme

\[ y(t + dt) = y(t) + \dot{y}(t)dt \quad \text{vs.} \quad y(t + dt) = y(t) + \dot{y}(t + dt)dt \]
Explicit solver

- Explicit integration:

\[ M_{n+1} = M_n + \lambda dt L(M_n) \]

\[ M_{n+1} = (I + \lambda dt L) M_n \]

Small time steps for large meshes
Implicit solver

- Implicit integration:

\[ M_{n+1} = M_n + \lambda dt L(M_{n+1}) \]

\[ \Rightarrow (I - \lambda dt L) M_{n+1} = M_n \]

Conjugate Gradient for efficiency

(Solve \( Ax = b \) by iterating \( p_{k+1} = r_k - \frac{p_k^T Ar_k}{p_k^T A p_k} p_k \) with \( r_k = b - Ax_k \).)

Large time steps
Signal analysis

- Benefit of implicit scheme evident in transfer functions

\[
\begin{align*}
\text{explicit} & : 1 - \lambda dt \omega^2 \\
\text{implicit} & : \frac{1}{1 + \lambda dt \omega^2}
\end{align*}
\]
Results

Initial mesh
Results

smoothed mesh
Shape preservation

Preventing tangential shift
1.7.3 Shape-preserving Smoothing
Shape denoising

- **Noise components**
  - Normal - think “shape”
  - Tangent - think “parameterization”

- **Alternative to diffusion:**
  curvature flow equation (Laplace-Beltrami)

\[
\Delta_S f (v) := \frac{2}{A (v)} \sum_{v_i \in N_1 (v)} (\cot \alpha_i + \cot \beta_i) (f (v_i) - f (v))
\]
Curvature flow

- Replace Laplacian operator with Laplace-Beltrami operator.
- Proceed as before.
Comparison

Initial Mesh  Regular Diffusion  Improved Diffusion  Curvature Flow
Results on 3D Scanned Data

Initial mesh

After one fairing
Results on 3D Scanned Data
Curvature operator

Curvature visualization using false colors

Low curvature
High curvature
Constraint enforcement

- **Fixed points/Fixed regions:**
  - Set the Laplacian to zero

- **Soft constraints:**
  - Locally adjust smoothing amount ($\lambda$)
1.8 Summary
Modeling

• **Object representation:** Meshes
  - Subdivision Surfaces
  - Arbitrary Meshes

• **Mesh data structures**
  - Half-edge structure etc.

• **Multiresolution modeling:**
  (LOD, local coefficients, filtering, editing)
  - Wavelets
  - Simplification
  - Progressive Meshes
  - Normal Meshes

• **Mesh Parametrization**
  - Planar mappings
  - Non-planar mappings
  - MAPS

• **Curves on Surfaces**

• **Smoothing**