Mesh Smoothing Algorithms

Overview
- Laplacian Smoothing
- Problems and fixes
- Vertex and Normal Constraints
- Normal Constraints at Boundary Vertices
- Isotropic vs. Anisotropic
- Linear vs. Nonlinear
- Filtering of Normal Fields
- Filters that Integrate Normal Fields
- Related Problems

Different Approaches
- Digital Signal Processing
- Physics-based / PDE Surfaces
- Variational / Regularization
- Multi-resolution
- Subdivision Surfaces

Classical Digital Signal Processing
- Signals defined on regular grids
  - 1D: music / speech
  - 2D: images / video
  - 3D: medical imaging
- Solid theoretical foundation and practical algorithms
  - Sampling Theorem
  - DFT/FFT Fourier Analysis
  - FIR/IIR Linear Filters / Convolution
  - Non-linear filtering
  - Multi-rate filtering / up-sampling / down-sampling
  - Etc.

Graph and Mesh Signals
- Graph signal
  Signal defined on a graph (irregular grids)
  - \( x = (x_1, \ldots, x_v)^T \)
  - \( G = (V,E) \)
    - \( V = \{i,j,\ldots\} \) --- vertices
    - \( E = \{e=(i,j),\ldots\} \) --- non-oriented edges
    - \( E = \{e=(i,j),\ldots\} \) --- oriented edges
- Mesh Signal
  Signal defined on the graph of a polygonal mesh

Laplacian Smoothing in Mesh Generation
- Used to improve quality of 2D meshes for FE computations
- Keep boundary vertices fixed
- Move each internal vertex to the barycenter of its neighbors

\[
\mathbf{v}_i' = \mathbf{v}_i + \lambda \Delta \mathbf{v}_i
\]
The Laplacian Operator

\[ \Delta v_i = \sum_j w_{ij} (v_j - v_i) \quad 1 = \sum_j w_{ij} \]

\[ 0 \leq w_{ij} \]

\[ v_i' = v_i + \lambda \Delta v_i \]

Laplacian Smoothing: Advantages

- Algorithm Simplicity
- Linear time and storage
- Edge length equalization
  (advantage depending on the application)
- Constraints and special effects
  by weight control

\[ \Delta v_i = \sum_j w_{ij} (v_j - v_i) \quad v_i' = v_i + \lambda \Delta v_i \]

Laplacian Smoothing: Challenges

- How to solve all the problems preserving
  - Algorithm Simplicity
  - Linear time and storage
- Proposed Solution:
  - Modify the Laplacian Operator
  - Isotropic / Anisotropic
  - Linear / non-linear (avoid)
  - Define Laplacian Operator on Normal fields
  - Use FIR linear filters
  - Dynamic connectivity resampling

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

Laplacian Smoothing: Disadvantages

- Overall Shrinkage
  - Solved by Taubin’s Low-Pass filter algorithm
  - Why? Fourier Analysis
- Edge length equalization
  (disadvantage depending on the application)
  - Solved by non-linear filtering
    - Fujiwara / Desbrun-et-al weights (curvature flow)
- Shrinkage at boundaries
  - Solved by hierarchical filtering?
- Smoothing of ridges
  - Solved by Anisotropic diffusion

Laplacian Smoothing Demo

Fourier Analysis

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

- Eigenvalues of \( K = I - W \) (Frequencies)
  \[ 0 = k_0 \leq k_1 \leq \cdots \leq k_N \leq 2 \]
- Right eigenvectors of \( K \) (Natural vibration modes)
  \[ e_0, e_1, \ldots, e_N \]
**Geometry of low and high frequencies**

\[
\mathbf{k}_n \mathbf{e}_n = \mathbf{K} \mathbf{e}_n' = -\sum_i w_{ij} (\mathbf{e}_i - \mathbf{e}_j)
\]
- Low frequency
- High frequency

**Natural vibration modes**

**The Discrete Fourier Transform**
- Eigenvectors form a basis of \(N\)-space
- Every signal can be written as a linear combination
  \[
  \mathbf{x} = \hat{x}_0 \mathbf{e}_0 + \hat{x}_1 \mathbf{e}_1 + \cdots + \hat{x}_N \mathbf{e}_N
  \]
- Discrete Fourier Transform (DFT)
  \[
  \hat{\mathbf{x}} = (\hat{x}_0, \hat{x}_1, \ldots, \hat{x}_N)^T
  \]

**FIR Linear Filters**
- Polynomial Transfer Function
  \[
  \mathbf{x}' = f(K)\mathbf{x} \quad \mathbf{K}\mathbf{x} = -\Delta\mathbf{x}
  \]
- \(f(k)\) is a univariate polynomial
- \(f(K)\) is a matrix
- Eigenvectors of \(K\) and \(f(K)\) are the same
- Eigenvalues of \(f(K)\) are
  \[
  f(k_0), f(k_1), \ldots, f(k_N)
  \]

**Laplacian Smoothing is not Low-Pass**
- After filtering
  \[
  f(K)\mathbf{x} = f(k_0)\hat{x}_0 \mathbf{e}_0^T + \cdots + f(k_N)\hat{x}_N \mathbf{e}_N
  \]
- Evaluation of \(f(K)\mathbf{x}\) based on matrix multiplication
- It does not require the computation of eigenvalues and eigenvectors (DFT)
- Low-Pass: need univariate polynomial \(f(k)\) such that
  - \(f(k_0) \approx 1\)
  - \(k_L \leq k_{PB}\)
  - \(f(k_0) \approx 0\)
  - \(k_L > k_{PB}\)
- For Laplacian smoothing
  \[
  f(k_0) = 1
  \]
  \[
  f(k_j) = (1 - \lambda k_j)^N \to 0 \quad j = 0 \quad 0 \leq \lambda < 1
  \]
Taubin Smoothing (Siggraph’95)

- 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

\[ f(k) = ((1 - \mu k)(1 - \lambda k))^\mu with \quad -\mu > \lambda > 0 \]

Taubin-Zhang-Golub (ECCV’96) FIR Filter Design

- Efficient algorithm to evaluate any polynomial transfer function
- Based on Chebyshev polynomials defined by three-term recursion
- All classical Finite Impulse Response (FIR) filter design techniques can be used with no modifications
- Implemented method of “windows” based on truncated Fourier series expansion of ideal transfer function and coefficient weighting to remove Gibbs phenomenon

Parameters

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

- Weights
  - Neighborhoods: non-zero weights
  - Prevention of tangential drift
  - Edge-length equalization
  - Boundaries and creases / hierarchical smoothing
  - Vertex-dependent smoothing parameters

Preventing tangential drift

- Fujiwara (P-AMS’95)
  - Weights inversely proportional to edge length
- Desbrun-Meyer-Schroder-Barr (SG’99)
  - Based on better approximation of curvature normal
- Guskov-et-al (SG’99) based on divided differences and second order neighborhood

Linear / Non-Linear

- Linear Laplacian Operator
  - Weights are computed once and kept constant for all iterations
- Non-Linear Laplacian Operator
  - Weights are recomputed at every iteration

Mesh Signal Processing

- Optimal Surface Smoothing as Filter Design, by G. Taubin, T. Zhang, and G. Golub, Fourth European Conference on Computer Vision (ECCV’96)
Hierarchical Neighborhoods

- Assign a numeric label to each vertex
- Vertex \( j \) is a neighbor of vertex \( i \) only if \( i \) and \( j \) are connected by an edge, and the label of \( i \) is less or equal than the label of \( j \)

Boundaries and Creases

- Use hierarchical neighborhoods
- Assign label 1 to boundary and crease vertices
- Assign label 0 to all internal vertices
- The graph defined by the boundary and crease edges and vertices is smoothed independently of the rest of the mesh
- The rest of the mesh “follows” the graph defined by the boundary and crease edges and vertices
- Eigenvalues of \( K \) are complex, but \( 1 - k_i \leq 1 \)

Boundaries and Creases

Hierarchical neighborhoods and weights

- \( W_{ij} > 0 \)
- \( W_{ji} > 0 \)
- \( W_{ik} = 0 \) but \( W_{ki} > 0 \)

Vertex Constraints and Surface Design

- Hard vs. soft constraints
- Hard vertex position constraints are easy to impose but produce artifacts because of lack of normal control
- Kobbelt-et-al Variational Fairing (SG’98)
  - Minimize square norm of Laplacian operator
- Yamada-et-al Discrete Spring Model (PCCGA’98)
  - Impose soft normal constraints with a spring model that adds an extra term to the smoothing step
- Slow convergence and/or high computational cost
Variational Fairing

- Minimize \( \sum_j \| \Delta v_j \|^2 \)
- Under linear constraints

The Boundary Shrinkage Problem

- Laplacian operator approximates
  - Mean curvature \( \times \) normal vector \( \times \) mean edge length
- Not for boundary vertices!
  - Has a strong tangential component
- Fix: project onto normal direction

Modified Laplacian for Boundary Vertices

- Project onto normal direction
  \( \Delta v_i = \sum_j w_{ij} \eta_i (v_j - v_i) \)
- Define weights as 3x3 matrices
  \( W_{ij} = w_{ij} \eta_i \)
- Linear Anisotropic Laplacian Operator

Anisotropic Laplacian Operators

\( \Delta v_i = \sum_j W_{ij} (v_j - v_i) \)
\( W_{ij} = C_i^j C_{ij} \)
\( C_{ij} \) Symmetric non-negative definite
\( C_i = \sum_j C_{ij} \)

Preventing Tangential Drift

- Use Laplacian Operator that fixes boundary shrinkage
- But, how to define the vertex normals?
- Use smooth face normal field instead

Smoothing Normal Fields

- Signal is defined on dual graph with values in the unit sphere
- Only need to define Laplacian Operator
- Then can apply any Linear Filter
- Displacement \( \eta_j - \eta_i \) is the Rotation defined by the vector product \( \eta_j \times \eta_i \)
- Laplacian Operator \( \Delta \eta_i \) is the average Rotation
Rodrigues Formula

• Local parameterization of Rotations
  \( \{u : |u| \leq 1\} \rightarrow SO(3) \)
  \( R(u) = cl : (1 - c)r^t + s \Lambda \)

• If \( \mathbf{n}_1 \) and \( \mathbf{n}_2 \) are two unit vectors, then
  \( R(\mathbf{n}_1 \times \mathbf{n}_2) \mathbf{n}_1 = \mathbf{n}_2 \)
  \( R(\mathbf{n}_1 \times \mathbf{n}_2) \mathbf{n}_1 \times \mathbf{n}_2 = \mathbf{n}_1 \times \mathbf{n}_2 \)

Laplacian for Normal Fields

• Definition
  \( \lambda \Delta \mathbf{n}_1 = R \left( \mathbf{n}_1 \times \left( \lambda \sum_j w_{ij} \mathbf{n}_j \right) \right) \)
  \( \mathbf{n}_1^* = R \left( \mathbf{n}_1 \times \left( \lambda \sum_j w_{ij} \mathbf{n}_j \right) \right) \mathbf{n}_1 \)

Constrained Normal Filtering

• Like vertex position constraints in the Euclidean case
• Just do not update the constrained values
• Face normals are filtered independently of vertex positions
• Then vertex positions are filtered with the linear anisotropic filter defined by the face normals
• Can impose both face normal constraints and vertex position constraints

Application: Hole filling

• Triangulate hole with internal vertices
• Smooth normal field in the graph defined by the hole faces and the incident faces
• Fix normals on incident faces
• Filter normals with boundary constraints
• Filter vertices with boundary constraints
• Use dynamic connectivity rules to resample if needed, and iterate

What Next?

• Combine with Dynamic Connectivity Rules for adaptive resampling
• Ridge detection and enhancement
• Non-linear isotropic and anisotropic filtering

Irregular Mesh Resampling


  - Define Min–Max target edge lengths
  - Collapse short edges
  - Optimize vertex valences by flipping edges
  - Smooth mesh
  - Split long edges
Non-Linear Anisotropic Diffusion