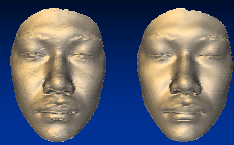


Bilateral Mesh Denoising

S. Fleishman, I. Drori, D. Cohen-Or

Tel Aviv University

Presented by Derek Bradley

Outline

- Motivation
- Previous Work
- Bilateral Mesh Denoising
  - Image Processing Background
  - Bilateral Image Filtering
  - Transforming from Images to Meshes
  - Mesh Denoising
- Results
- Discussion

(Unless otherwise noted, all images are from Fleishman et al.)

Motivation

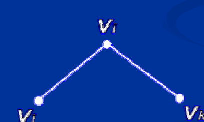
- 3D scanning creates noisy meshes
- Smoothing can reduce high frequency noise
- Challenge: how do you know what is noise and what are features?

Previous Work

- Laplacian Smoothing

$$v_i = v_i + \lambda \cdot \Delta v_i$$

$$\Delta v_i = \frac{1}{|(i,j)|} \sum_{(i,j)} (v_j - v_i)$$

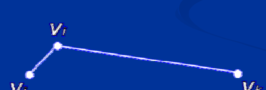


Previous Work

- Weighted Laplacian

$$v_i = v_i + \lambda \cdot \Delta v_i$$

$$\Delta v_i = \frac{1}{\sum_{(i,j)} w_{ij}} \sum_{(i,j)} w_{ij} (v_j - v_i)$$

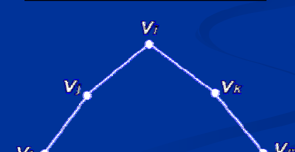


Previous Work

- Laplacian + Expansion

$$\Delta v_i^2 = \frac{1}{|(i,j)|} \sum_{(i,j)} (\Delta v_j - \Delta v_i)$$

$$v_i = v_i + (\mu - \lambda)\Delta v - \mu\lambda\Delta v_i^2$$



Previous Work

- Implicit Fairing (IF) [Desbrun 1999]
  - Implicit integration of the diffusion equation

$$X^{n+1} = (I + \lambda dtL)X^{n}$$
Explicit

$$(I + \lambda dtL)X^{n+1} = X^{n}$$
Implicit

- Anisotropic Feature-Preserving Denoising (AFP) [Desbrun 2000]
  - Features detected using local curvature
  - Denoise using weighted mean curvature smoothing
    - Penalize vertices with large ratio between principle curvatures

Bilateral Mesh Denoising

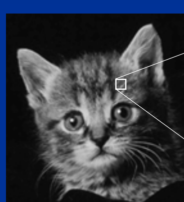
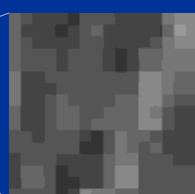
- Application of an image smoothing technique
- Vertices are moved along their normal direction

$$v_i = v_i + d \cdot n_i$$

- Scalar value ***d*** to be computed for each vertex
- Feature preserving
- Can be iterative or single-pass
- But first … some image processing basics

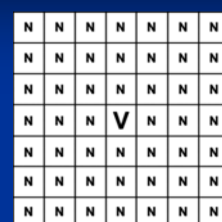
Image Processing Background

- An image is an array of integers (0-255)

[Tomasi and Manduchi]

Image Processing Background



- v is the current pixel
- N(v) is the set of neighbouring pixels of v
- I(v) is the intensity of v

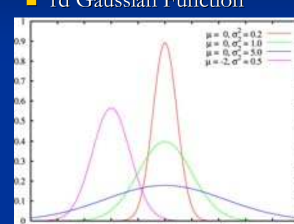
Bilateral Image Filtering

- Goal: Smooth the image intensities, but preserve strong edges (features)
- New intensity = weighted average of neighbours
- Two weights:
  - Geometric: Closer pixels weighted higher (closeness smoothing filter)
  - Photometric: Strong changes in intensity penalized (similarity weight function)

Bilateral Image Filtering

Closeness Smoothing Filter

- 1d Gaussian Function

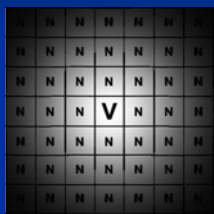


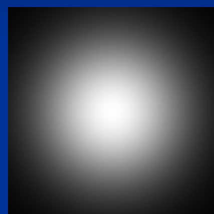
[wikipedia.org]

Bilateral Image Filtering

Closeness Smoothing Filter

- 2d Gaussian Filter





$$W_c(x) = e^{-x^2 / 2\sigma_c^2}$$

Bilateral Image Filtering

Similarity Weight Function

- Another Gaussian function

$$W_s(x) = e^{-x^2 / 2\sigma_s^2}$$

- **x** = absolute difference in intensity values
- Result: pixels with large changes in intensity are weighted lower

Bilateral Image Filtering

- Combining the weights and normalizing:

$$\hat{I}(v) = \frac{\sum_{p \in N(v)} W_c(\|p - v\|) \cdot W_s(|I(v) - I(p)|) \cdot I(p)}{\sum_{p \in N(v)} W_c(\|p - v\|) \cdot W_s(|I(v) - I(p)|)}$$

- In practice, **N(v)** is defined by the set of points:

$$\{q_i\}, \text{ where } \|v - q_i\| < |2\sigma_c|$$

Bilateral Image Filtering

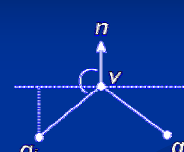
- Results:

[Tomasi and Manduchi]

Transforming from Images to Meshes

- Vertices instead of pixels
- Neighbourhood **N(v)**, defined the same
- Closeness smoothing filter:
  - 3D Euclidean distance instead of 2D
- Similarity weight function:
  - Heights of neighbouring vertices = pixel intensities

Transforming from Images to Meshes

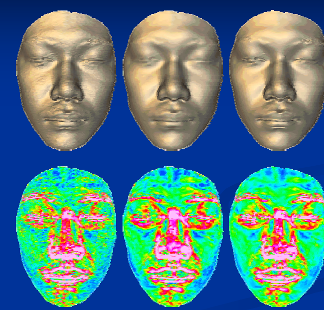


- Dot product between normal and (**v**-$\mathbf{q}_i$) used instead of computing the height at $\mathbf{q}_i$

Mesh Denoising

DenoisePoint(Vertex v, Normal n)
- $\{q_i\}$ = neighbourhood(v)
- $K = |\{q_i\}|$
- sum = 0, normalizer = 0
- **for** i := 1 **to** K
  - $t = ||v - q_i||$
  - $h = \langle n, v - q_i \rangle$
  - $W_c = \exp(-t^2/(2\sigma_c^2))$
  - $W_s = \exp(-h^2/(2\sigma_s^2))$
  - sum += $(W_c * W_s) * h$
  - normalizer += $W_c * W_s$
- **end**
- **return** Vertex $v = v + n * (\text{sum}/\text{normalizer})$

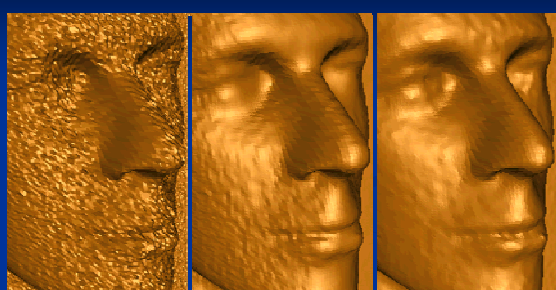
Results



Mean Curvature

Input | Implicit Fairing | Bilateral Denoising

Results



Input | Anisotropic Denoising of Height Fields (AFP) | Bilateral Denoising

Discussion

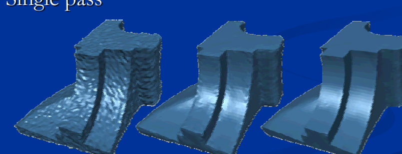
- Issues when using an image-based technique on a mesh:
  - Only applies to manifold meshes
  - Irregularity of meshes
  - Shrinkage
  - Vertex drift
- Handling boundaries
  - Mirror neighbours at boundary vertices
  - Virtual vertices at infinity (used in this algorithm)

Discussion

- Setting the parameters (σ_c, σ_s, # iterations)
  - User-assisted method
  - σ_c and σ_s :
    - User selects smooth point and radius on the mesh
  - Large σ_c = few iterations, small σ_c = more iterations
  - Small σ_c makes sense
    - large values can cross features
    - smaller neighbourhood leads to faster iterations
  - < 6 iterations for all results in the paper

Discussion

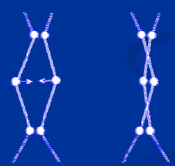
- Independently, Jones et al. present the same algorithm with minor differences:
  - Surface predictor
  - Single pass



Input | Jones et al. | Bilateral Denoising

Discussion

- Disadvantages
  - Assumes well-behaved meshes
  - Can result in self-intersection



Conclusion

- Simple, effective and fast algorithm for denoising meshes
- Easy to implement
- Takes advantage of the success of an image processing technique
- Would I implement this algorithm? Yes

References

- S. Fleishman, I. Drori, D. Cohen-Or. Bilateral mesh denoising. SIGGRAPH 2003.
- C. Tomasi, R. Manduchi. Bilateral filtering for gray and color images. ICCV 1998.
- T. Jones, F. Durand, M. Desbrun. Non-iterative feature-preserving mesh smoothing. SIGGRAPH 2003.
- M. Desbrun, M. Meyer, P. Schroder, A.H. Barr. Implicit fairing of irregular meshes using diffusion and curvature flow. SIGGRAPH 1999.
- M. Desbrun, M. Meyer, P. Schroder, A.H. Barr. Anisotropic feature-preserving denoising of height fields and bivariate data. Graphics Interface 2000.
- www.wikipedia.org