A Signal Processing Approach to Fair Surface Design

Authors:
Gabriel Taubin

IBM T.J.Watson Research Center

Presentation By:
Charles Robertson

Overview

- Introduction
- Fairing
- Subdivision
- Constraints
- Conclusions

Introduction

- Motivation
  - Fairing surfaces
  - Medical data
- Goals
  - Quick, linear-time algorithm
  - Non-shrinking
  - Incorporate constraints

Closed Curve Fairing

- Extension of Fourier analysis to surfaces
  - Discrete Laplacian for surface
    - $\Delta x_i = \frac{1}{2}(x_{i-1} - x_i) + \frac{1}{2}(x_{i+1} - x_i)$
    - $\Delta x = -K\,x$
    - $K = \frac{1}{2}\begin{pmatrix} 2 & -1 & & & -1 \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ -1 & & & -1 & 2 \end{pmatrix}$
    - $x = \sum_{i=1}^{n} \xi_i\, u_i$

Closed Curve Fairing

- Unfortunately the decomposition of this summation is expensive even with FFT
- Instead, lets do this projection approximately
  - $x'_i = x_i + \lambda \Delta x_i$
  - $x' = (I - \lambda K) x$
- As Gaussian filtering, this produces shrinkage.
- They define a non-shrinking function to replace $I - \lambda K$

Surface Signal Fairing

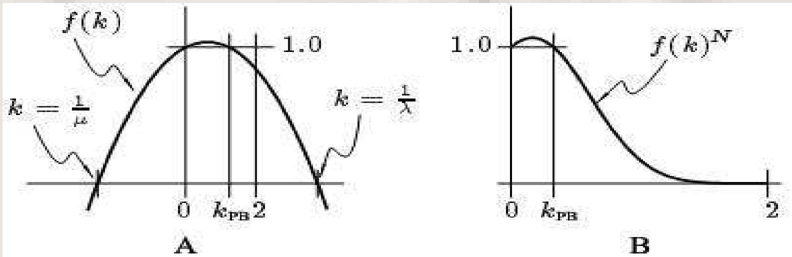
- Discrete surface signal
  - $\Delta x_i = \sum_{j \in i^*} w_{ij} (x_j - x_i)$
- Choosing weights
  - $w_{ij} = \dfrac{\phi(v_i, v_j)}{\sum_{h \in i^*} \phi(v_i, v_h)}$
- While Fourier descriptors are no longer feasible for large spaces of vertices, low-pass filtering still is.

Surface Signal Fairing

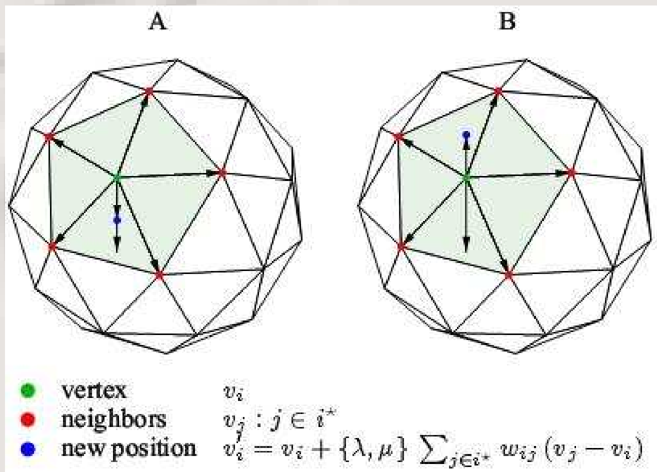
- Polynomial transfer function
  - $x' = f(K) x = \sum_{i=1}^{n} \xi_i f(k_i) u_i$

Surface Signal Fairing

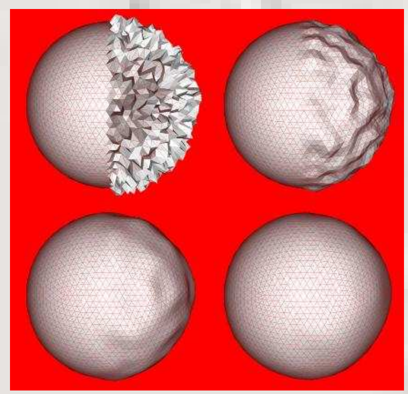
- Low-pass filter
  - $f(k) = (1 - \lambda k)(1 - \mu k)$
- Pass-band frequency
  - $k_{\mathrm{PB}} = \dfrac{1}{\lambda} + \dfrac{1}{\mu} > 0$

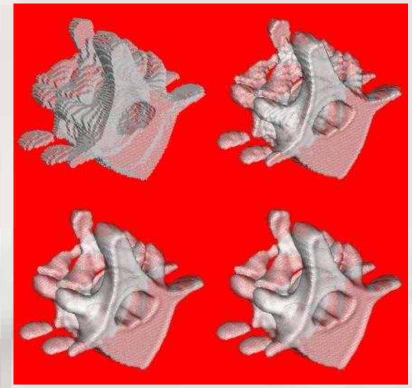


Surface Signal Fairing

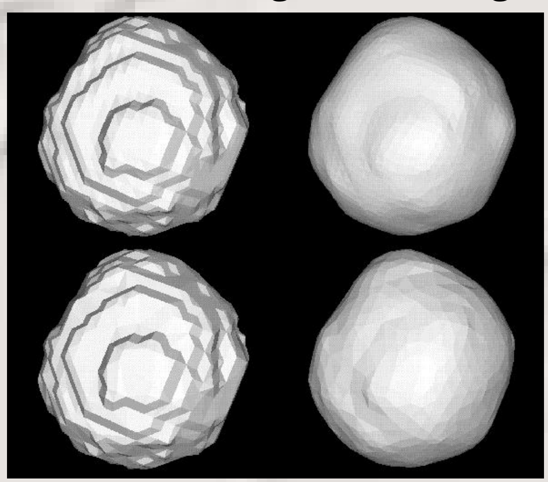


Surface Signal Fairing

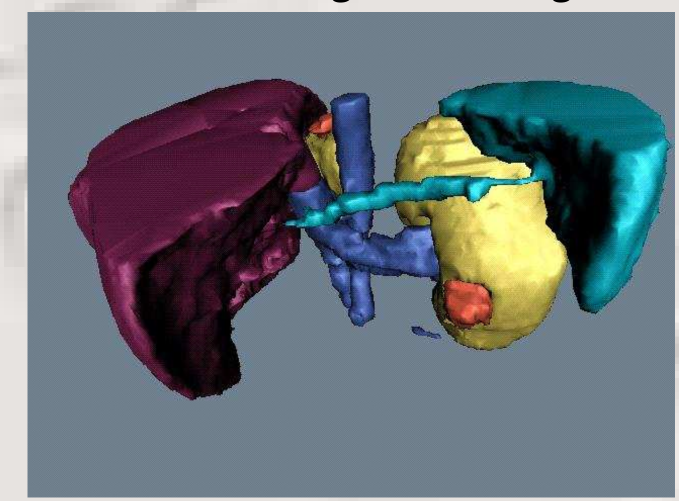




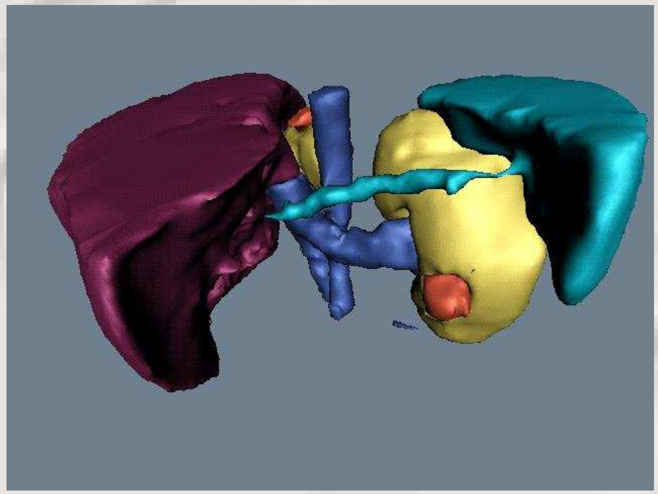
Surface Signal Fairing



Surface Signal Fairing

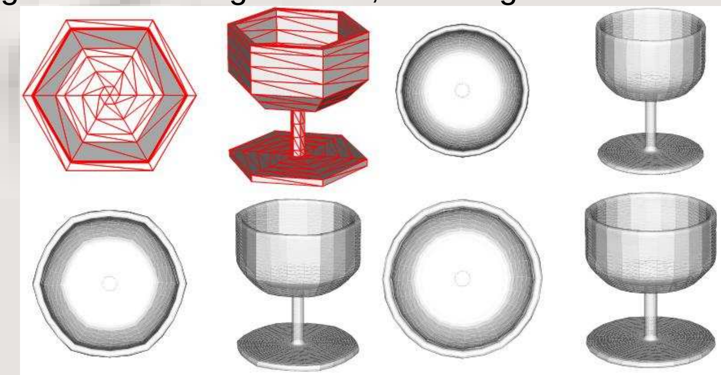


Surface Signal Fairing



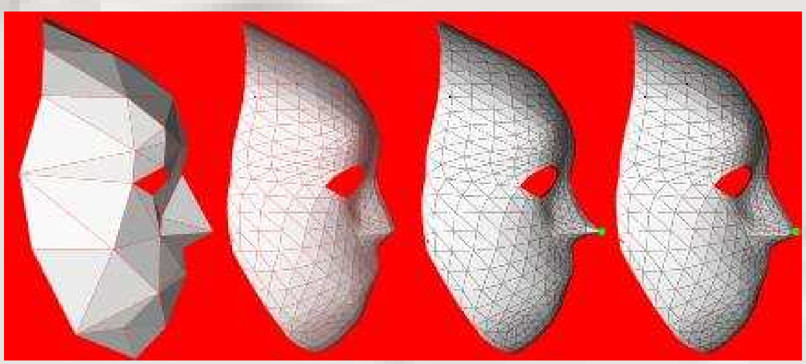
Subdivision

- Subdivision schemes suffer from shrinkage.
- Using this smoothing method, shrinkage is reduced.



Interpolatory Contraints

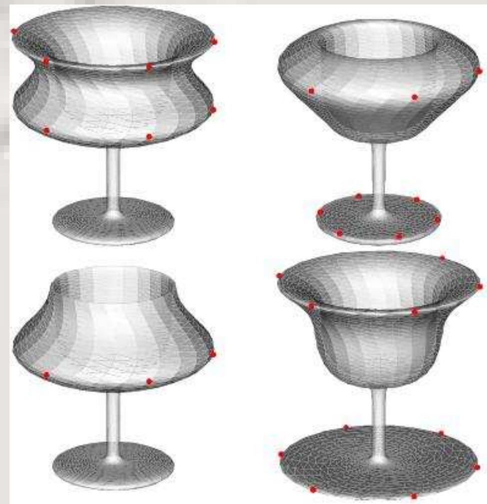
- Easy constraint is non-symmetric neighborhoods



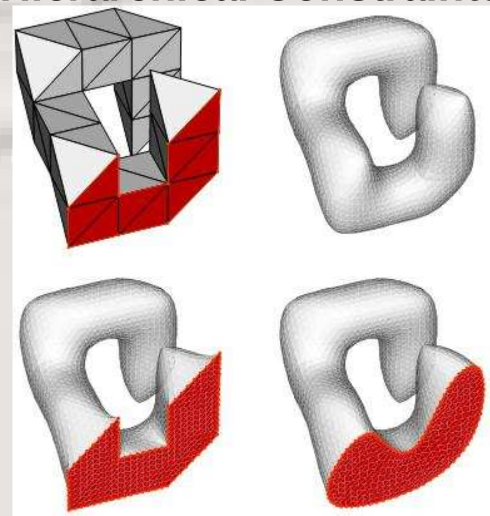
Smooth Interpolation

- Instead of simply editing the neighborhood, they add in a new deformation that returns a point to its original location
  - $x_C^N = x^N + (x_1 - x_1^N)\, d_1$
- More generally:
  - $x_C^N = x^N + F_{nm}\, F_{mm}^{-1} \begin{pmatrix} x_1 - x_1^N \\ \vdots \\ x_m - x_m^N \end{pmatrix}$
- Computation can be sped up by precomputing
  - $F_{mm} \begin{pmatrix} y_1 \\ \vdots \\ y_m \end{pmatrix} = \begin{pmatrix} x_1 - x_1^N \\ \vdots \\ x_m - x_m^N \end{pmatrix}$

Smooth Deformations



Hierarchical Constraints



Tangent Plane Constraints

- Shading
  - $\Delta v_i = \sum_{j \in i^\star} w_{ij} (v_j - v_i)$
  - $\frac{1}{|\gamma|} \int_{v \in \gamma} (v - v_i)\, dl(v)$
  - $\lim_{\epsilon \to 0} \frac{1}{|\gamma_\epsilon|} \int_{v \in \gamma_\epsilon} (v - v_i)\, dl(v) = \bar{\kappa}(v_i) N_i$
  - $S_i^t \Delta v_i^N = T_i^t \Delta v_i^N = 0$

General Linear Constraints

- Generalization of the smooth interpolation problem seen before.
  - $C x_C^N = c$
- To see it is equivalent to smooth interpolation we look at:
  - $\begin{cases} x_{(1)} & = & y_{(1)} - C_{(1)}^{-1} C_{(2)}\, y_{(2)} \\ x_{(2)} & = & y_{(2)} \end{cases}$
  - $y_{(1)} = C_{(1)}^{-1} c$

Conclusions

- Linear time and space fairing system
- Constraints allowed
  - Differing network structure
  - Differing vertex handling
  - Subdivision

Questions?