Shape Modeling

Sketch-based interfaces and algorithms for shape creation and editing
Overview

- **Teddy**  
  [Igarashi et al. 1999]

- **Sketch based mesh editing**  
  [Nealen et al. 2005]

- **FiberMesh**  
  [Nealen et al. 2007]

- **SilSketch**  
  [Zimmermann et al. 2008]
<table>
<thead>
<tr>
<th></th>
<th>Machine</th>
<th>Human</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Video in</strong></td>
<td>![Machine Image]</td>
<td>![Human Image]</td>
</tr>
<tr>
<td><strong>Video out</strong></td>
<td>![Machine Image]</td>
<td>![Human Image]</td>
</tr>
</tbody>
</table>
The Main Question

- How can we convey a mental model of shape to a digital computer?

<table>
<thead>
<tr>
<th>Video in</th>
<th>Machine</th>
<th>Human</th>
</tr>
</thead>
<tbody>
<tr>
<td>Video out</td>
<td><img src="image1.png" alt="Projector" /> <img src="image2.png" alt="TIE fighter" /></td>
<td><img src="image3.png" alt="Sketch" /> <img src="image4.png" alt="Model" /></td>
</tr>
</tbody>
</table>
Motivation
Model Creation

- Digital 3D model creation from scratch is hard
- Common workflow(s)
  - Create clay model + scan
    - Laser scanner
    - Position tracking
  - Create coarse 3D model + refine
    - Subdivision surfaces
    - Parametric patches
    - Detail maps
Problem

- How to construct 3D surface from 2D sketch?
  - Infer the missing depth information
  - Construct a discrete surface
  - What rule(s) to use?
Background

- Study on Human perception of 3D shape from 2D drawings [Hoffman 2004]
Specific Algorithms

- Level-set method [Williams 91]
- Heuristic mesh construction [Igarashi 99]
- Volumetric construction [Owada 03]
- Convolution surface [Alexe 05]
- Blob tree (Implicit surface) [Schmidt 05]
- Mass-spring system [Karpenko 06]
- Optimization [Nealen 07]
Level-set Method

[Williams 91]

- Compute implicit representation of the contour
Teddy
Heuristic Mesh Construction

Teddy
Heuristic Mesh Construction

1. Find axes  
2. Elevate axes  
3. Wrap polygon and axes
Heuristic Mesh Construction

Input 2D polygon
Heuristic Mesh Construction

Constrained Delaunay Triangulation

Andrew Nealen, Rutgers, 2009
4/1/2009
Heuristic Mesh Construction

Discrete chordal axis [Prasad 97]
Heuristic Mesh Construction

Before trimming

After trimming
Lift the axes, put quarter ovals on the internal edges, and generate mesh.
Teddy

- Results
  - Fast and first of its kind
  - Meshes are of low quality
Sketch-based mesh editing
Motivation

Model Modification

- Digital 3D model modification is nontrivial
- Multiresolution mesh editing with subdivision
Motivation

Model Modification

- Digital 3D model modification is nontrivial
- Laplacian / Poisson mesh editing /w handle

[Sorkine et al. 2004]

Affine Handle

[Botsch and Kobbelt 2004]
Ideas and Contributions

• A sketch-based interface...
  – Feature modification
    (object-space silhouettes)
  – Feature creation
    (sharp features, ridges, ravines)

• ... For detail-preserving mesh editing
  – Adjust remaining geometry around the modified/created feature
    such that shape characteristics are preserved

Andrew Nealen, Rutgers, 2009
Ideas and Contributions

- Silhouette sketching
  - Sketching a shape can be interpreted as inverse Non-Photorealistic Rendering

- A sketch-based modeling interface which uses silhouettes and sketches as input, and produces contours, ridges and ravines.
Inspiration and Motivation

- The (affine) handle metaphor
  - Used in (almost) every editing tool
  - Nice, but can be unintuitive for specific editing tasks

- Laplacian Mesh Editing
  - Preserve local detail after imposing editing constraints

\[ \delta_i = \frac{1}{d_i} \sum_{j \in N(i)} (\mathbf{v}_i - \mathbf{v}_j) \]

[Sorkine et al. 04] [Zhou et al. 05]
Mesh Modeling Framework

- Discrete Laplacians

\[ \delta_i = x_i - \frac{1}{\sum_{(i,j) \in E} w_{ij}} \sum_{(i,j) \in E} w_{ij} x_j \]

\[ \delta_{\text{cotangent}} : w_{ij} = \cot \alpha_{ij} + \cot \beta_{ij} \]
Mesh Modeling Framework

- Surface reconstruction

\[ \delta_i = \mathbf{x}_i - \frac{1}{\sum_{(i,j) \in E} w_{ij}} \sum_{(i,j) \in E} w_{ij} \mathbf{x}_j \]

\[ \delta_{\text{cotangent}} : w_{ij} = \cot \alpha_{ij} + \cot \beta_{ij} \]
Mesh Modeling Framework

- Implicit transformations

Implicitly compute $T_i$
For the details see: Laplacian Mesh Editing
[Sorkine et al. 04]

deformed mesh
**Mesh Modeling Framework**

- **Surface reconstruction**

$$n \rightarrow \text{fix}$$

$$\mathbf{y} = \mathbf{c}_1$$

Andrew Nealen, Rutgers, 2009

4/1/2009
Mesh Modeling Framework

- Editing operations

\[ \begin{array}{c}
L/T_i & L/T_i \\
T_i & L/T_i \\
1 & 1 \\
1 & 1 \\
\end{array} \]

\[ \begin{align*}
x &= 0 \\
y &= c_1 \\
z &= c_2
\end{align*} \]
Mesh Modeling Framework

• Least-Squares solution

Normal Equations

\[ A^T A \mathbf{x} = A^T \mathbf{b} \]

\[ \mathbf{x} = (A^T A)^{-1} A^T \mathbf{b} \]

\[ A x = b \]

\[ \text{Normal Equations} \]

\[ \text{Mesh Modeling Framework} \]
• Using silhouettes as handles
  – Detect object space silhouette
  – Project to screen space and parametrize $[0,1]$
  – Parametrize sketch $[0,1]$
  – Find correspondences
Silhouette Sketching

• Using silhouettes as handles
  – Detect object space silhouette
  – Project to screen space and parametrize [0,1]
  – Parametrize sketch [0,1]
  – Find correspondences
  – Use as positional constraints while retaining depth value
Silhouette Sketching

- What is a good silhouette?
Silhouette Sketching

- What is a good silhouette?

Illustrating Smooth Surfaces [Hertzmann and Zorin 00]
Silhouette Sketching

- On edge constraints

\[ \lambda \mathbf{x}_i + (1-\lambda) \mathbf{x}_j \]
Silhouette Sketching

- Approximate sketching
  - Balance weighting between detail and positional constraints
Silhouette Sketching

- Approximate sketching
  - Balance weighting between detail and positional constraints
We wish to influence (discrete) differential properties of the mesh for arbitrary sketches

Possible solution
- Cut existing polygons along the sketch and add new edges

Our solution
- Adjust mesh geometry to lie under the sketch (as seen from the camera), while preserving mesh topology and ensuring well shaped triangles
First: min cost edge path (close to sketch)

Potentially *jaggy* appearance
Geometry Adjustment

- Second: projection onto sketch
Geometry Adjustment

- Second: **projection onto sketch**
  - Approximates the sketch very well
  - Can introduce badly shaped tri’s
Geometry Adjustment

- Third: **local mesh regularization**
  - Ask uniformly weighted Laplacian to become cotangent weighted Laplacian, while fixing path vertices

\[
L \ x = \delta
\]
Third: **local mesh regularization**
- Well shaped triangles and nice piecewise linear approximation of the users sketch

$$L \ x = \delta$$

Geometry Adjustment

Andrew Nealen, Rutgers, 2009
Feature Edit

- Edit: scale (or add to) Laplacians
Feature Edit

- Edit: scale (or add to) Laplacians
Feature Edit

\[ \mathbf{v}_1, \mathbf{v}_2, \delta_{\text{cotangent}} \]
Laplacian Constraints

- Scale (or add to) Laplacians

Normal Equations

\[ \begin{align*} 
A^T A x &= A^T b \\
(\text{Normal Equations}) \\
A^T A x &= (A^T A)^{-1} A^T b
\end{align*} \]
Contour Edit

radial plane
Contour Edit

radial plane

Andrew Nealen, Rutgers, 2009
Contour Edit

radial curvature
Inflection line
Contour Edit

Andrew Nealen, Rutgers, 2009
Editing Session Result
Problem: noisy Surface Silhouette
Discussion...

- The good...
  - Intuitive, sketch-based User Interface for silhouette deformation and feature creation/modification
  - Fast model updates after sketch (Iterative Modeling)
  - Preserves surface detail as much as possible

- ... and the *not so good*
  - Object-Space sil’s useless in the presence of heavy noise
  - Editing differential properties can take time to learn
FiberMesh
Problem Statement

- 3D modeling from scratch is **difficult**
  
  **Parametric patches**
  
  **Subdivision surfaces**
  
  Initial patch layout can be tedious

**Sketching**

Produces simple, rough models
FiberMesh
Overview and Contributions

• **Curves as the user interface**
  • The user’s sketches are persistent, and used as a modeling handles
  • Topologically flexible = add / remove anywhere

• **Fast surface construction by (nonlinear) functional optimization**
  • Incorporates curve constraints
  • Runs at interactive rates
Algorithm Overview

- **3D curve deformation**

  Handle position \rightarrow Curve geometry

- **Surface optimization**

  Curve geometry \rightarrow Surface geometry

Please see paper for details...
Surface Optimization?
Inspirations
Curve and Surface Optimization

- Minimal energy networks [Moreton and Sequin 1991]
- Functional optimization [Moreton and Sequin 1992]
- Variational surface modeling [Welch and Witkin 1992]
- Shape design with triangulated surfaces [Welch and Witkin 1994]

**What defines „smooth“?**

\[ L^k (x) = 0 \]
Surface Optimization
Discrete Differential Geometry

See work from
Polthier, Desbrun, Meyer, Alliez, Grinspun, Schröder, etc.
Surface Optimization
Discrete Differential Geometry

- **Laplacian operators**
  - Uniform Laplacian $L_u(v_i)$
  - Cotangent Laplacian $L_c(v_i)$
  - Mean curvature normal

![Diagram showing a geometric configuration with vectors and angles labeled $v_i$, $v_j$, $\alpha$, $\beta$, and $A_i$.]
Surface Optimization

Two Important Observations!

- Laplacian operators
  - Uniform Laplacian $L_u(v_i)$
  - Cotangent Laplacian $L_c(v_i)$
  - Mean curvature normal
- Cotangent Laplacian $= \text{mean curvature normal} \times \text{vertex area} (A_i)$
- For nearly equal edge lengths $\text{Uniform} \approx \text{Cotangent}$
Surface Optimization

Two Important Observations!

- Laplacian operators
  - Uniform Laplacian $L_u(v_i)$
  - Cotangent Laplacian $L_c(v_i)$
  - Mean curvature normal
- Cotangent Laplacian = mean curvature normal $\times$ vertex area ($A_i$)
- For nearly equal edge lengths Uniform $\approx$ Cotangent
Surface Optimization
Linear- or Nonlinear Optimization?

- Linear variational methods [Botsch and Sorkine 2007]
- Instead: nonlinear optimization
  - Inspired by a discrete fairing algorithm
    - Geometric fairing of irregular meshes for free-form surface design [Schneider and Kobbelt 2002]
Surface Optimization
Overview of a Single Step

Current geometry

Iterate vertex positions

Linear system
But: matrix recomputation

Current curvatures (scalar)

Diffusion (smoothing)

Target curvatures (scalar)

Target mean curvature normals

Unit normals
\[ \mathbf{c} \cdot \mathbf{n} \]

Positional constraints

Scale unit normal vectors

\[
\mathbf{L}_c \mathbf{c} = \mathbf{0}
\]

Resulting geometry

Iterative procedure not designed for interactive response

• Linear system
• But: matrix recomputation

Unit normals

Target mean curvature normals

Positional constraints

Scale unit normal vectors

Iterate vertex positions

Resulting geometry

Iterative procedure not designed for interactive response
Surface Optimization
Our Optimizations and Approximations

Current geometry

Current curvatures (scalar)
Diffusion (smoothing)
Target curvatures (scalar)
Scale unit normal vectors
Target mean curvature normals

Positional constraints

Iterate vertex positions

Resulting geometry

Linear system

Reuse matrix factorization

$\mathbf{L}_u \mathbf{C} = \mathbf{0}$

$\mathbf{c} \cdot \mathbf{n} = c_0$
Surface Optimization
Our Optimizations and Approximations

Current geometry

Current curvatures (scalar)

Linear system

Reuse matrix factorization

Target curvatures (scalar)

Diffusion (smoothing)

Scale unit normal vectors

Target mean curvature normals

\[ \mathbf{L}_u \mathbf{x} = \mathbf{c} \cdot \mathbf{n} \]

Solve least squares

Resulting geometry

Positional constraints

\[ \mathbf{L}_u \mathbf{x} = \mathbf{c} \cdot \mathbf{n} \]
Surface Optimization
Our Optimizations and Approximations

Current geometry

Current curvatures (scalar)

Diffusion (smoothing)

Target curvatures (scalar)

Scale unit normal vectors

Target mean curvature normals

Solve least squares

Resulting geometry

\[ \mathbf{L}_u \mathbf{x} = \delta_c \]

Positional constraints

- Linear system
- Reuse matrix factorization

Scale unit normal vectors:

\[ \mathbf{L}_u \mathbf{x} = \delta_c \]

Target mean curvature normals:

\[ \mathbf{n} = c \cdot \mathbf{n} \]
Surface Optimization
Our Optimizations and Approximations

**Current geometry**

- Current curvatures (scalar)
- Diffusion (smoothing)
- Target curvatures (scalar)

**Linear system**

\[ L_u C = 0 \]

- Solve least squares
- Scale unit normal vectors

**Target cotan Laplacians**

\[ \delta_c = c \cdot n \cdot A \]

- Positional constraints

**Resulting geometry**

\[ L_u x = \delta_c \]
Surface Optimization
Requirements for our Approximations

- We replace $L_c$ with $L_u$ in both steps
- This a viable approximation for:
Surface Optimization

Requirements for our Approximations

- We replace $L_c$ with $L_u$ in both steps
- This a viable approximation for:
  - (A) nearly equal edge lengths (smooth metric)
  - (B) nearly equal vertex areas
Surface Optimization
How to Enforce (A) and (B)

Setting $L_u(x)$ equal to $\delta_c$ improves inner fairness

$[\text{Nealen et al. 2006}]$

Smooth the edge lengths
Constrain edge vectors
Surface Optimization
Adding Edge Vector Constraints to LS

Current geometry
Current edge lengths (scalar)

Current geometries
Current curvatures (scalar)

Diffusion (smoothing)
Target curvatures (scalar)

Target edge lengths (scalar)

Scale vectors

Target $\delta_c$

Target edge vectors

Solve least squares

Positional constraints

Resulting geometry

Unit edge vectors

$= c \cdot n \cdot A$
Surface Optimization
A Single Step of the Optimization

1. Current geometry
2. Current curvatures (scalar)
3. Current edge lengths (scalar)
4. Diffusion (smoothing)
5. Target curvatures (scalar)
6. Target edge lengths (scalar)
7. Target edge vectors
8. Solve least squares
9. Scale vectors
10. Target $\delta_c$
11. $= c \cdot n \cdot A$
12. Positional constraints
13. Resulting geometry
Results
15 min instructions + 20 minutes tryout
Results
2D Artist

15 min instructions + 10 minutes tryout
Results

2D Artist

... + 20 minutes tryout
Results
Base Model Creation
- Takes a bit to learn the interface operations and their combinations
- Larger meshes would be nice
  - Perhaps „freeze“ part of the mesh
  - In general: entire mesh is optimized
  - Multigrid / Multiresolution acceleration techniques
- Add more intuitive modeling operations: symmetry, vertex snapping, merging, etc.
SilSketch
SilSketch
Demo
Sil Detection and Segmentation
Image Space
Which Lines to Match Against?

We Use

Depth Discontinuities

Normal Discontinuities

Flat Shaded

Combined Discontinuities
Some Segmentation Results

Resolve depth ambiguity
Partial Matching
Image Space
Partial Matching

Input
- Silhouette polyline(s)
- User stroke (polyline as well)

Criteria
- Proximity
- Shape similarity

Shape Matching
- Partial Matching of Polylines under Similarity Transformations Cohen and Guibas [1997]
Handle deformation and ROI

Object Space
Handle Vertices
Moving from 2D to 3D
Results
Editing Sequence (1)
Editing Sequence (2)
Editing Sequence Result
The human visual system uses silhouettes as the first index into its memory of shapes
  - SilSketch leverages this human “shape database”
  - This enables the creation of diverse shapes for non professionals

- Extends affine handle transformations to general handle warps
- Was added to the content creation pipeline at Disney Animation Studios
Many other works out there...

- **SKETCH**  
  [Zeleznik et al. 1996]

- **Variational implicits**  
  [Karpenko et al. 2002]

- **ShapeShop**  
  [Schmidt et al. 2005]
Conclusions

- Further step(s) towards a human “video out”