“Paint by Numbers” and “Comprehensible Rendering of 3D Shapes”

Prof. Allison Klein

Announcements

- Sign up for 1st presentation at end of class today
- Undergrads:
  - Thinking about grad school?
  - Still here over the summer?
  - Do an undergrad research project
  - NSERC undergrad student research awards
    - Deadline is February 2nd!

Outline

- Haeberli, “Paint by Numbers”
- Saito and Takahashi, “Comprehensible Rendering of 3D Shapes”
- Comparison

Outline: Haeberli’s Paper

- Introduction/Motivation
- Brush Strokes
- Advanced Techniques
- Summary/Conclusions

Introduction

- SIGGRAPH ’90
- Motivated by quote from M. Hagen (artist)
  “The goal of effective representational image making, whether you paint in oil or in numbers, is to select and manipulate visual information in order to direct the viewer’s attention and determine the viewer’s perception.”
- Also motivated by the Impressionists

Introduction

“Impression Sunrise” Monet
Basic Idea
- Interactive technique for image processing

How it works:
- User drags mouse over source image
- Strokes colored by point-sampling image
- Other mouse data can control other qualities
  - Speed of mouse movement
  - Direction of mouse movement
  - Etc.

Varying parameters → different imagery

Outline: Haeberli’s Paper
- Introduction/Motivation
- Brush Strokes
- Advanced Techniques
- Summary/Conclusions

Brush Strokes
- Painting = ordered list of brush strokes
- Control characteristics of brush strokes:
  - Location
  - Color (RGBA)
  - Size (length and width)
  - Direction (angle)
  - Shape (rounded vs rectangular vs wavy etc.)
Brush Strokes: Location

- Clicking mouse button specifies location
- Strokes stochastically distributed around click point
- Attempt at “hand-craftedness”

Brush Strokes: Locations

[Figure 1 from Haeberli: Stochastic brush strokes based on mouse movement]

Brush Strokes: Color

- Strokes colored by point-sampling image
- Contribution: avoids explicit color selection
- Potential drawback: artist’s often don’t pick realistic colors

Brush Strokes: Size

Two Methods:
1. Average speed of cursor
   - Painting quickly = large brush strokes
   - Painting slowly = small brush strokes
   - Idea: quickly create rough representation of image, then add finer details later
2. Using up and down keys

Brush Strokes: Orientation

Two Methods (very similar):
1. Strokes oriented in direction of mouse movement
2. Use mouse gestures (i.e. user presses down on mouse button, moves in direction desired, releases button)

Brush Strokes: Shape

Now we’re getting to the interesting stuff!
- Geometry can be selected by pop-up menu
  - Circles
  - Rectangles
  - Lines
  - Scatterings of Points
  - Polygons
  - User-defined shape

[Figure 3 from Haeberli: Example stroke shapes]
**Outline: Haeberli’s Paper**

- **Introduction/Motivation**
- **Brush Strokes**
- **Advanced Techniques**
- **Summary/Conclusions**

**Brush Strokes: Shape**

- Strokes = circles = Pointillism
- Changing cones to hyperboloids = Voronoi Diagrams (Hoff et al. SIGGRAPH 99)
- Le Cirque” from Seurat

---

**Brush Strokes**

```
<table>
<thead>
<tr>
<th>Position</th>
<th>RGBA Color</th>
<th>Size</th>
<th>Dir Brush</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.847 0.941</td>
<td>241 129 173 255</td>
<td>50 99 5</td>
<td></td>
</tr>
<tr>
<td>0.844 0.931</td>
<td>222 197 35 255</td>
<td>50 99 5</td>
<td></td>
</tr>
<tr>
<td>0.441 0.524</td>
<td>172 29 1 255</td>
<td>50 99 5</td>
<td></td>
</tr>
<tr>
<td>0.444 0.553</td>
<td>230 100 75 255</td>
<td>50 99 5</td>
<td></td>
</tr>
<tr>
<td>0.447 0.526</td>
<td>220 122 162 255</td>
<td>50 99 5</td>
<td></td>
</tr>
<tr>
<td>0.456 0.354</td>
<td>245 189 137 255</td>
<td>50 99 5</td>
<td></td>
</tr>
<tr>
<td>0.503 0.522</td>
<td>245 183 237 255</td>
<td>50 99 5</td>
<td></td>
</tr>
<tr>
<td>0.479 0.345</td>
<td>228 161 92 255</td>
<td>50 99 5</td>
<td></td>
</tr>
<tr>
<td>0.498 0.337</td>
<td>245 181 230 255</td>
<td>50 99 5</td>
<td></td>
</tr>
</tbody>
</table>
```

---

**Brush Strokes: Shape**

- Strokes = Z-buffered cones = Dirichlet domains
- Depth in Z

---

**Brush Strokes: Shape**

- 3D cones of differing colors
- Depth in Z

---

**An example ordered list**

```
- Position
- RGBA Color
- Size
- Dir Brush
```

---

**Figure 5 from Haeberli:**

An example ordered list

---

**Figure 4 from Haeberli:**

3D cones of differing colors

---

**Figure 4 from Haeberli:**

Strokes = Z-buffered cones = Dirichlet domains

---

**Figure 4 from Haeberli:**

Strokes = circles = Pointillism

---

**Figure 4 from Haeberli:**

Le Cirque” from Seurat

---

**Figure 4 from Haeberli:**

Changing cones to hyperboloids = Voronoi Diagrams (Hoff et al. SIGGRAPH 99)

---

**Figure 4 from Haeberli:**

3D cones of differing colors

---

**Figure 4 from Haeberli:**

Depth in Z
Post-Processing:
- Stored strokes can be manipulated later
- Options:
  - Adding noise to stroke parameter(s)
  - Rotating brush strokes
  - Simple animation:
    - Apply same stroke parameters to image sequence
    - Color still comes from underlying image
    - See Litwinowicz's paper on Thursday

Other Advanced Techniques
- Stroke direction:
  - 2nd input image
  - Aligned to image edges
- Stroke blending
- 3D scenes as input
- Relaxation for automatic painting

Advanced: Stroke Direction
- Figure 6 from Haeberli:
  Stroke directions based on 2nd input image
- We will see this sequence of steps over and over again!

Advanced: Stroke Blending
- Figure 8 from Haeberli:
  Note inset brush texture
Advanced: Painting 3D Scenes
- Raytrace scene to find color, surface-normal, and depth at each pixel
- Surface-normal controls stroke direction

Advanced: Relaxation
- Place brush strokes (randomly? thru UI?)
- Stochastically perturb stroke attributes
- Minimize root mean squared dist between painting and original image

Outline: Haeberli’s Paper
- Introduction/Motivation
- Brush Strokes
- Advanced Techniques
- Summary/Conclusions

Summary/Conclusions
- Interactive technique for image processing
- Also 3D rendering
- Varying parameters → different imagery
- Lots of ideas in this paper!

Summary/Conclusions
- Haeberli wants brush strokes to convey:
  - Surface color
  - Surface curvature
  - Center of focus
  - Location of edges
- Do they? How?
- What parts does system know?
- What parts does user provide?
Summary/Conclusions
Another way of thinking about it:
- How does this help/hinder a good artist?
- How does this help/hinder a talentless person?
- Artist’s tool vs. automated system?

Outline
- Haeberli, “Paint by Numbers”
- Saito and Takahashi, “Comprehensible Rendering of 3D Shapes”
- Comparison

Outline: Saito & Takahashi
- Introduction/Motivation
- Geometric Buffers (“G-Buffers”)
- Basic Effects
- Applications/Results
- Conclusions/Discussion

Introduction/Motivation
- SIGGRAPH ’90
- Comprehensible rendering of 3D shapes
- Applications:
  - Industrial Design
  - Medical Imaging
  - Topo/Contour Maps
  - Etc.

Introduction/Motivation
- SIGGRAPH ’90
- Comprehensible rendering of 3D shapes
- Applications:
  - Industrial Design
  - Medical Imaging
  - Topo/Contour Maps
  - Etc.

Introduction/Motivation
- SIGGRAPH ’90
- Comprehensible rendering of 3D shapes
- Applications:
  - Industrial Design
  - Medical Imaging
  - Topo/Contour Maps
  - Etc.

Introduction/Motivation
- SIGGRAPH ’90
- Comprehensible rendering of 3D shapes
- Applications:
  - Industrial Design
  - Medical Imaging
  - Topo/Contour Maps
  - Etc.
Introduction/Motivation

- Comprehensibility (according to S&T):
  - Comes from image enhancement
  - Not from accurately simulating optical phenomena

- Paper focuses on line-drawing techniques:
  - Outlining silhouettes and creases
  - Hatching to convey curvature
  - Done as 2D image processing for speed

Introduction/Motivation

Process Overview:
- Input = 3D Model
- Rendering
  - Traditional shaded image
  - G-buffers for additional info
- Image Processing
  - Combine above for final image

Note that we can make images at the image processing and combining stages without having to re-render!
Outline: Saito & Takahashi

- Introduction/Motivation
- Geometric Buffers ("G-Buffers")
- Basic Effects
- Applications/Results
- Conclusions/Discussion

G-Buffers

- Intermediate rendering result
- Stores info about the visible object at that pixel location
- Used as input for enhancement effects
- Typical G-buffer contents *might* be:
  - Id: object/patch identifier
  - Z-depth
  - Nx: normal vector x
  - Ny: normal vector y
  - Nz: normal vector z
  - Etc.

Basic Effects: Edge Drawing

Two kinds of edges:
- Profile (aka silhouette):
  - Object border on screen
  - 0th order discontinuity in depth
- Internal edge (aka crease):
  - Line where two faces meet
  - 1st order discontinuity in depth

Both can be found with 3x3 box kernels provided in paper...

Basic Effects: Contour Lines

- Popular visualization technique
- Path of constant value (height, temp, etc.)

...and with a little extra normalization of values, you get silhouettes and creases.
Basic Effects: Contour Lines

Let:
- \( p \) be the scalar value of the desired contour
- \( s \) be the value of an input image pixel
- \( g \) be the gradient value of input image pixel
- \( d \) be the contour width, in pixels
- \( c_b \) be desired background intensity (i.e. 1.0)
- \( c_c \) be desired contour intensity (i.e. 0.0)

We want to calculate contour value \( c \) at each pixel

\[
c = c_b + f_i \left( \frac{s - p}{g} \right) \cdot (c_c - c_b)
\]

\[f_i(t) = \begin{cases} 0 & 0 \leq t < \frac{d}{2} \\ 1 & \frac{d}{2} \leq t \end{cases}\]

These equations work for single value of \( p \)
See paper for boundary cases

Basic Effects: Curved Hatching

Like contour lines except drawn at regular pixel intervals
So S&T modify contour approach to get uniform density
See paper for details
Outline: Saito & Takahashi

- Introduction/Motivation
- Geometric Buffers (“G-Buffers”)
- Basic Effects
- Applications/Results
- Conclusions/Discussion

Applications/Results

Line Drawings (Figure 9 from S&T)

Applications/Results

Topo Map Process (Figure 11 from S&T)

Applications/Results

Figure 8: Edge Enhancement Of Reflected Objects

Applications/Results

Figure 13: Sumi-e?

Outline: Saito & Takahashi

- Introduction/Motivation
- Geometric Buffers (“G-Buffers”)
- Basic Effects
- Applications/Results
- Conclusions/Discussion
Conclusions/Discussion

- Process for *comprehensible* images
  - Input = 3D Model
  - Rendering
    - Traditional shaded image
    - G-buffers for additional info
  - Image Processing
  - Combine above for final image

- Mainly line drawing, but some shading
- Written when *rendering* was slow

Outline

- Haeberli, “Paint by Numbers”
- Saito and Takahashi, “Comprehensible Rendering of 3D Shapes”

Comparison

Some ways to think about these papers:

- 2D versus 3D input
- Artistic stylization versus visualization
- How much user input does each require?
- When do they require user input?
- What styles can you do with one vs the other?
- What styles can’t you do with one vs the other?
- Possibilities for future work?
- Other ways to compare?