Image Warping, Compositing & Morphing

Adam Finkelstein
Princeton University
COS 426, Spring 2005

Image Processing

- Quantization
  - Uniform Quantization
  - Random dither
  - Ordered dither
  - Floyd-Steinberg dither
- Pixel operations
  - Add random noise
  - Add luminance
  - Add contrast
  - Add saturation
- Filtering
  - Blur
  - Detect edges
- Warping
  - Scale
  - Rotate
  - Warp
- Combining
  - Morph
  - Composite

Image Warping

- Move pixels of image
  - Mapping
  - Resampling
- Warping
  - Scale
  - Rotate
  - Warp

Overview

- Mapping
  - Forward
  - Reverse
- Resampling
  - Point sampling
  - Triangle filter
  - Gaussian filter

Mapping

- Define transformation
  - Describe the destination \((x, y)\) for every location \((u, v)\) in the source (or vice-versa, if invertible)
Example Mappings

- Scale by factor:
  - \( x = \text{factor} \times u \)
  - \( y = \text{factor} \times v \)

- Rotate by \( \Theta \) degrees:
  - \( x = u \cos \Theta - v \sin \Theta \)
  - \( y = u \sin \Theta + v \cos \Theta \)

- Shear in X by factor:
  - \( x = u + \text{factor} \times v \)
  - \( y = v \)

- Shear in Y by factor:
  - \( x = u \)
  - \( y = v + \text{factor} \times u \)

Other Mappings

- Any function of \( u \) and \( v \):
  - \( x = f_x(u,v) \)
  - \( y = f_y(u,v) \)

Image Warping Implementation I

- Forward mapping:
  ```
  for (int u = 0; u < umax; u++) {
    for (int v = 0; v < vmax; v++) {
      float x = f_x(u,v);
      float y = f_y(u,v);
      dst(x,y) = src(u,v);
    }
  }
  ```

Forward Mapping

- Iterate over source image
Forward Mapping - NOT

- Iterate over source image

Many source pixels can map to same destination pixel

Image Warping Implementation II

- Reverse mapping:
  ```
  for (int x = 0; x < xmax; x++) {
    for (int y = 0; y < ymax; y++) {
      float u = \lf(x,y);
      float v = \fy^{-1}(x,y);
      dst(x,y) = src(u,v);
    }
  }
  ```

Source image
Destination image

Reverse Mapping

- Iterate over destination image
  - Must resample source
  - May oversample, but much simpler!

Resampling

- Evaluate source image at arbitrary (u,v)

(u,v) does not usually have integer coordinates

Overview

- Mapping
  - Forward
  - Reverse
  - Resampling
    - Point sampling
    - Triangle filter
    - Gaussian filter
Point Sampling

• Take value at closest pixel:
  o int iu = trunc(u+0.5);
  o int iv = trunc(v+0.5);
  o dst(x,y) = src(iu,iv);

Filtering

• Compute weighted sum of pixel neighborhood
  o Weights are normalized values of kernel function
  o Equivalent to convolution at samples

\[
s = 0;
\text{for } (i = -w; i <= w; i++)
\text{for } (j = -w; j <= w; j++)
\text{s += k(i,j)*I(u+i, v+j)};
\]

\[
\sum_{k(i, j) = 1} k(i, j) \text{ represented by gray value}
\]

Triangle Filtering

• Kernel is triangle function

\[
\text{Bilinearly interpolate four closest pixels}
\]

\[
(a = \text{linear interpolation of src}(u_1,v_2) \text{ and src}(u_2,v_3))
\text{ and src}(u_1,v_1) \text{ and src}(u_2,v_1))
\]

\[
\text{dst}(x,y) = \text{linear interpolation of "a" and "b"}
\]

Triangle Filtering (with width = 1)

• Kernel is triangle function

\[
\text{Width of filter affects blurriness}
\]
**Gaussian Filtering**

- Kernel is Gaussian function

    ![Gaussian Function](image)

    Filter Width = 2

**Filtering Methods Comparison**

- Trade-offs
  - Aliasing versus blurring
  - Computation speed

**Image Warping Implementation**

- Reverse mapping:

  ```
  for (int x = 0; x < xmax; x++) {
    for (int y = 0; y < ymax; y++) {
      float u = fx^{-1}(x,y);
      float v = fy^{-1}(x,y);
      dst(x,y) = resample_src(u,v,w);
    }
  }
  ```

**Example: Scale**

- Scale (src, dst, sx, sy):

  ```
  float w = max(1.0/sx,1.0/sy);
  for (int x = 0; x < xmax; x++) {
    for (int y = 0; y < ymax; y++) {
      float u = x / sx;
      float v = y / sy;
      dst(x,y) = resample_src(u,v,w);
    }
  }
  ```

**Example: Rotate**

- Rotate (src, dst, theta):

  ```
  for (int x = 0; x < xmax; x++) {
    for (int y = 0; y < ymax; y++) {
      float u = x*cos(-theta) - y*sin(-theta);
      float v = x*sin(-theta) + y*cos(-theta);
      dst(x,y) = resample_src(u,v,w);
    }
  }
  ```
Example: Fun

- Swirl (src, dst, theta):
  ```java
  for (int x = 0; x < xmax; x++) {
      for (int y = 0; y < ymax; y++) {
          float u = rot(dist(x, xcenter)*theta);
          float v = rot(dist(y, ycenter)*theta);
          dst(x,y) = resample_src(u,v,w);
      }
  }
  ```

Image Processing

- Quantization
  - Uniform Quantization
  - Random dither
  - Ordered dither
  - Floyd-Steinberg dither
- Pixel operations
  - Add random noise
  - Add luminance
  - Add contrast
  - Add saturation
- Filtering
  - Blur
  - Detect edges
- Warping
  - Scale
  - Rotate
  - Warp
- Combining
  - Morph
  - Composite

Overview: combining images

- Image morphing
  - Specifying correspondences
  - Warping
  - Blending
- Image compositing
  - Blue-screen mattes
  - Alpha channel
  - Porter-Duff compositing algebra

Overview: combining images

- Image morphing
  - Specifying correspondences
  - Warping
  - Blending
- Image compositing
  - Blue-screen mattes
  - Alpha channel
  - Porter-Duff compositing algebra

Image Morphing

- Animate transition between two images

Cross-Dissolving

- Blend images with “over” operator
  - alpha of bottom image is 1.0
  - alpha of top image varies from 0.0 to 1.0
  ```plaintext
  blend(i,j) = (1-t) src(i,j) + t dst(i,j)   (0 ≤ t ≤ 1)
  ```
**Image Morphing**
• Combines warping and cross-dissolving

```
<table>
<thead>
<tr>
<th>src</th>
<th>warp</th>
<th>cross-dissolve</th>
<th>warp</th>
<th>dst</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>t = 0.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>t = 0.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>t = 1.0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

**Feature-Based Warping**
• Beier & Neeley use pairs of lines to specify warp
  - Given p in dst image, where is p' in source image?

```
Source image
\( u \) is a fraction
\( v \) is a length (in pixels)

Destination image
```

**Warping with One Line Pair**
• What happens to the “F”?

- Translation!
- Scale!
- Rotation!
Warping with One Line Pair

• What happens to the “F”?

In general, similarity transformations

What types of transformations can’t be specified?

Warping with Multiple Line Pairs

• Use weighted combination of points defined by each pair of corresponding lines

Beier & Neeley, Figure 4

Warping with Multiple Line Pairs

• Use weighted combination of points defined by each pair of corresponding lines

Source image

Destination image

p’ is a weighted average

Warping Pseudocode

WarpImage(Image, L[1], L[n])
begin
  foreach destination pixel p do
    psum = (0,0)
    wsum = 0
    foreach line L[i] in destination do
      p’[i] = p transformed by (L[i], L’[i])
      psum = psum + p’[i] * weight[i]
      wsum += weight[i]
    end
    p’ = psum / wsum
    Result(p) = Image(p’)
  end
end

Morphing Pseudocode

GenerateAnimation(Image0, L[1], ..., Image1, L[n])
begin
  foreach intermediate frame time t do
    for i = 1 to number of line pairs do
      L[i] = line t-th of the way from L0[i] to L1[i]
    end
    Warp0 = WarpImage(Image0, L0, L)
    Warp1 = WarpImage(Image1, L1, L)
    foreach pixel p in FinalImage do
      Result(p) = (1-t) Warp0 + t Warp1
    end
  end
end
Beier & Neeley Example

Overview

• Image compositing
  o Blue-screen mattes
  o Alpha channel
  o Porter-Duff compositing algebra

• Image morphing
  o Specifying correspondences
  o Warping
  o Blending

Image Compositing

• Separate an image into “elements”
  o Render independently
  o Composite together

• Applications
  o Cel animation
  o Chroma-keying
  o Blue-screen matting

Blue-Screen Matting

• Composite foreground and background images
  o Create background image
  o Create foreground image with blue background
  o Insert non-blue foreground pixels into background

Problem: no partial coverage!

Even CG folks Can Win an Oscar

Smith Porter Catmull Duff
**Alpha Channel**

- Encodes pixel coverage information
  - $\alpha = 0$: no coverage (or transparent)
  - $\alpha = 1$: full coverage (or opaque)
  - $0 < \alpha < 1$: partial coverage (or semi-transparent)

- Example: $\alpha = 0.3$

![Partial Coverage](image)

![Semi-Transparent](image)

**Compositing with Alpha**

Control the linear interpolation of foreground and background pixels when elements are composited.

- $\alpha = 0$
- $0 < \alpha < 1$
- $\alpha = 1$

**Semi-Transparent Objects**

- Suppose we put A over B over background G
- How much of B is blocked by A?
  - $\alpha_A$
- How much of B shows through A
  - $(1-\alpha_A)$
- How much of G shows through both A and B?
  - $(1-\alpha_A)(1-\beta_B)$

**Opaque Objects**

- How do we combine 2 partially covered pixels?
  - 3 possible colors (0, A, B)
  - 4 regions (0, A, B, AB)

**Composition Algebra**

- 12 reasonable combinations

**Example: C = A Over B**

- Consider the areas covered:
  - $C = \alpha_A A + (1-\alpha_A) \alpha_B B$
  - $\alpha = \alpha_A + (1-\alpha_A) \alpha_B$

Assumption: coverages of A and B are uncorrelated for each pixel
Image Composition Example

- Image Composition Example

- Image Composition Example

Image Processing

- Quantization
  - Uniform Quantization
  - Random dither
  - Ordered dither
  - Floyd-Steinberg dither

- Pixel operations
  - Add random noise
  - Add luminance
  - Add contrast
  - Add saturation

- Filtering
  - Blur
  - Detect edges

- Warping
  - Scale
  - Rotate
  - Warp

- Combining
  - Composite
  - Morph

Next Time: 3D Rendering

Misha Kazhdan, CS426, Fall99