Image Warping, Compositing & Morphing
Adam Finkelstein
Princeton University
COS 426, Fall 2001

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

Image Warping

• Move pixels of image
  - Mapping
  - Resampling

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:
  - $x = \text{factor} \times u$
  - $y = \text{factor} \times v$

Example Mappings

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

Example Mappings

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

Example Mappings

- 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 = f_x^{-1}(x,y);
      float v = f_y^{-1}(x,y);
      dst(x,y) = src(u,v);
    }
  }
  ```

**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:
  - \[ \text{int } i_u = \text{trunc}(u+0.5); \]
  - \[ \text{int } i_v = \text{trunc}(v+0.5); \]
  - \[ \text{dst}(x,y) = \text{src}(i_u,i_v); \]

**Triangle Filtering**

- Convolve with triangle filter

- Bilinearly interpolate four closest pixels
  - \[ a = \text{linear interpolation of } \text{src}(u_1,v_2) \text{ and } \text{src}(u_2,v_2) \]
  - \[ b = \text{linear interpolation of } \text{src}(u_1,v_1) \text{ and } \text{src}(u_2,v_1) \]
  - \[ \text{dst}(x,y) = \text{linear interpolation of } "a" \text{ and } "b" \]

**Gaussian Filtering**

- Compute weighted sum of pixel neighborhood:
  - Weights are normalized values of Gaussian function

**Filtering Methods Comparison**

- Trade-offs
  - Aliasing versus blurring
  - Computation speed
Image Warping Implementation

- Reverse mapping:

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

Example: Scale

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

```java
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):

```java
for (int x = 0; x < xmax; x++) {
    for (int y = 0; y < ymax; y++) {
        float u = x*cos(-θ) - y*sin(-θ);
        float v = x*sin(-θ) + y*cos(-θ);
        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
  - Composite
  - Morph
Overview: combining images

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

Even CG folks can win an Oscar

Image Compositing

- Separate an image into "elements"
  - Render independently
  - Composite together
- Applications
  - Cel animation
  - Chroma-keying
  - Blue-screen matting

Dobkin meets the King

Blue-Screen Matting

- Composite foreground and background images
  - Create background image
  - Create foreground image with blue background
  - Insert non-blue foreground pixels into background
  - Problem: no partial coverage!

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: $\alpha = 0.3$

Compositing with Alpha

Controls the linear interpolation of foreground and background pixels when elements are composited.
**Pixels with Alpha**

- Alpha channel convention:
  - \((r, g, b, \alpha)\) represents a pixel that is \(\alpha\) covered by the color \(C = (r/\alpha, g/\alpha, b/\alpha)\)
  - Color components are premultiplied by \(\alpha\)
  - Can display \((r,g,b)\) values directly
  - Closure in composition algebra

- What is the meaning of the following?
  - \((0, 1, 0, 1)\) = ?
  - \((0, 1/2, 0, 1)\) Full green, full coverage
  - \((0, 1/2, 0, 1/2)\) Half green, full coverage
  - \((0, 1/2, 0, 0)\) = ? Full green, half coverage

**Semi-Transparent Objects**

- Suppose we put A over B over background G

**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**

- For colors that are not premultiplied:
  - \(C = \alpha_A A + (1-\alpha_A) \alpha_B B\)
  - \(\alpha = \alpha_A + (1-\alpha_A) \alpha_B\)

- For colors that are premultiplied:
  - \(C' = A' + (1-\alpha_A) B'\)
  - \(\alpha = \alpha_A + (1-\alpha_A) \alpha_B\)

**Image Composition Example**

- Jurassic Park
Overview

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

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
  \[ \text{blend}(i,j) = (1-t) \text{src}(i,j) + t \text{dst}(i,j) \quad (0 \leq t \leq 1) \]

Image Morphing

- Animate transition between two images

Cross-Dissolving

- Specifying correspondences
- Warping
- Blending

Feature-Based Warping

- Beier & Neeley use pairs of lines to specify warp
  - Given \( p \) in dst image, where is \( p' \) in source image?

\[ u \text{ is a fraction} \]
\[ v \text{ is a length (in pixels)} \]
Warping with One Line Pair

• What happens to the “F”?

Translation!

Warping with One Line Pair

• What happens to the “F”?

Scale!

Warping with One Line Pair

• What happens to the “F”?

Rotation!

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

$p'$ is a weighted average
Weighting Effect of Each Line Pair

To weight the contribution of each line pair, Beier & Neeley use:

\[ \text{weight}[i] = \frac{\text{length}[i]^p}{a + \text{dist}[i]} \]

Where:
- \(\text{length}[i]\) is the length of \(L[i]\)
- \(\text{dist}[i]\) is the distance from \(X\) to \(L[i]\)
- \(a, b, p\) are constants that control the warp

Warping Pseudocode

\[
\text{WarpImage}(\text{Image}, L[\ldots], L'[\ldots])
\begin{align*}
\text{begin} & \\
\quad & \text{foreach destination pixel} \ p \ \text{do} \\
\quad & \quad \text{psum} = (0,0) \\
\quad & \quad \text{wsum} = 0 \\
\quad & \quad \text{foreach line} \ L[i] \ \text{in destination do} \\
\quad & \quad \quad \text{p}'[i] = p \ \text{transformed by} \ (L[i], L'[i]) \\
\quad & \quad \quad \text{psum} = \text{psum} + p'[i] \times \text{weight}[i] \\
\quad & \quad \quad \text{wsum} += \text{weight}[i] \\
\quad & \quad \text{end} \\
\quad & \quad p' = \text{psum} / \text{wsum} \\
\quad & \quad \text{Result}(p) = \text{Image}(p') \\
\text{end} \\
\end{align*}
\]

Morphing Pseudocode

\[
\text{GenerateAnimation}(\text{Image}_0, L_0[\ldots], \text{Image}_1, L_1[\ldots])
\begin{align*}
\text{begin} & \\
\quad & \text{foreach intermediate frame} \ t \ \text{do} \\
\quad & \quad \text{for} \ i = 1 \ \text{to number of line pairs do} \\
\quad & \quad \quad L[i] = \text{line} \ t-\text{th of the way from} \ L_0[i] \ \text{to} \ L_1[i] \\
\quad & \quad \end{align*}
\]

\[
\text{Warp}_0 = \text{WarpImage}(\text{Image}_0, L_0, L) \\
\text{Warp}_1 = \text{WarpImage}(\text{Image}_1, L_1, L) \\
\text{foreach pixel} \ p \ \text{in} \ \text{FinalImage} \ \text{do} \\
\quad \text{Result}(p) = (1-t) \ \text{Warp}_0 + t \ \text{Warp}_1 \\
\text{end} \\
\text{end}
\]

Beier & Neeley Example

Beier & Neeley Example

\[
\text{Image}_0 \quad \text{Warp}_0 \quad \text{Result} \\
\text{Image}_1 \quad \text{Warp}_1
\]

CS426 Examples

CS426 Class, Fall98

Robert Osada, Fall00
### 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, Fall 99