### Image Warping, Compositing & Morphing

**Greg Humphreys**  
University of Virginia  
CS 445, Fall 2003

### Image Processing

<table>
<thead>
<tr>
<th><strong>Quantization</strong></th>
<th><strong>Filtering</strong></th>
<th><strong>Warping</strong></th>
<th><strong>Combining</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Uniform Quantization</td>
<td>Blur</td>
<td>Scale</td>
<td>Composite</td>
</tr>
<tr>
<td>Random dither</td>
<td>Detect edges</td>
<td>Rotate</td>
<td>Morph</td>
</tr>
<tr>
<td>Ordered dither</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Floyd-Steinberg dither</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Pixel operations</strong></th>
<th><strong>Warp</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Add random noise</td>
<td>Scale</td>
</tr>
<tr>
<td>Add luminance</td>
<td>Rotate</td>
</tr>
<tr>
<td>Add contrast</td>
<td>Warp</td>
</tr>
<tr>
<td>Add saturation</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Filtering</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Blur</td>
</tr>
<tr>
<td>Detect edges</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Combining</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Composite</td>
</tr>
<tr>
<td>Morph</td>
</tr>
</tbody>
</table>

### Image Warping

- Move pixels of image
  - Mapping
  - Resampling

![Source image](source-image.png) → **Warp** → ![Destination image](destination-image.png)
## 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:
  - $x = u + \text{factor} \times v$
  - $y = v$

- 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 = fx(u, v);
      float y = fy(u, v);
      dst(x, y) = src(u, v);
    }
  }
  ```

Forward Mapping

- Iterate over source image
Forward Mapping – Bad Idea

• Iterate over source image

Many source pixels can map to same destination pixel

Some destination pixels may not be covered

Reverse Mapping

• Iterate over destination image

  - Must resample source
  - May oversample, but much simpler!

Image Warping Implementation II

• 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) = src(u, v);
    }
}

Source image
Destination image
Resampling
• Evaluate source image at arbitrary (u,v)

(u,v) does not usually have integer coordinates

Source image  Destination image

Overview
• Mapping
  ▪ Forward
  ▪ Reverse
  ▪ Resampling
    ▪ Point sampling
    ▪ Triangle filter
    ▪ Gaussian filter

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

This method is simple, but it causes aliasing

Triangle Filtering
• Convolve with triangle filter

Input Output

Rotate -30
Scale 0.5

Figure 2.4 Wolberg
**Triangle Filtering**

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

**Gaussian Filtering**

- Convolve with Gaussian filter

**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

Figure 2.4 Wolberg
Image Warping Implementation

- Reverse mapping:

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

Example: Scale

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

```c
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):

```c
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

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

Gordon begins his political career

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.

$\alpha = 1$

$\alpha = 0$

$0 < \alpha < 1$
**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?
  - \((0, 1, 0, 1)\) = ? Full green, full coverage
  - \((0, 1/2, 0, 1)\) = ? Half green, full coverage
  - \((0, 1/2, 0, 1/2)\) = ? Full green, half coverage
  - \((0, 1/2, 0, 0)\) = ? No coverage

---

**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 \(B\)?
    \((1-\alpha_A)(1-\alpha_B)\)

---

**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

- Porter & Duff ‘84
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:
  - $C' = A' + (1-\alpha_A) B'$
  - $\alpha = \alpha_A + (1-\alpha_A) \alpha_B$

Assumption: coverages of $A$ and $B$ are uncorrelated for each pixel

Overview

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

- Image morphing
  - Specifying correspondences
  - Warping
  - Blending

Image Morphing

- Animate transition between two images

H&B Figure 16.9
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

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

<table>
<thead>
<tr>
<th>t = 0.0</th>
<th>t = 0.5</th>
<th>t = 1.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>src</td>
<td>blend</td>
<td>dst</td>
</tr>
</tbody>
</table>

Image Morphing

- Combines warping and cross-dissolving

<table>
<thead>
<tr>
<th>t = 0.0</th>
<th>t = 0.5</th>
<th>t = 1.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>src</td>
<td>warp</td>
<td>src</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?

[Diagram showing mapping with u and v coordinates]

<table>
<thead>
<tr>
<th>u is a fraction</th>
<th>v is a length (in pixels)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source image</td>
<td>Destination image</td>
</tr>
</tbody>
</table>

Image Morphing

- The warping step is the hard one
  - Aim to align features in images

[Diagram showing mapping with u and v coordinates]
Warping with One Line Pair

• What happens to the "F"?

<table>
<thead>
<tr>
<th>Translation!</th>
</tr>
</thead>
</table>

| Scale! |

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

![Image of Warping with Multiple Line Pairs](image)

Beier & Neeley, Figure 4

Weighting Effect of Each Line Pair

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

$$weight[i] = \left( \frac{length[i]^p}{a + dist[i]} \right)$$

Where:
- $length[i]$ is the length of $L[i]$
- $dist[i]$ is the distance from $X$ to $L[i]$
- $a$, $b$, $p$ are constants that control the warp

Warp Image

WarpImage(Image, L[...], L[...])

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[...], Image1, L[...])
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, L, L)
    Warp1 = WarpImage(Image1, L, L)
    foreach pixel p in FinalImage do
      Result(p) = (1-t) Warp0 + t Warp1
    end
  end
end

Beier & Neeley 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