Image Warping, Compositing & Morphing

Thomas Funkhouser
Princeton University
COS 426, Spring 2004

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

- 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 $v$:
  - $x = f(u,v)$
  - $y = f(u,v)$

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

Example Mappings

- Scale 0.8
- Rotate 30°

Example Mappings

- Shear X 1.3
- Shear Y 1.3

Other Mappings

- Fish-eye
- “Swirl”
- “Rain”
**Forward Mapping - NOT**

- Iterate over source image

Many source pixels can map to same destination pixel

**Reversing Mapping:**

- Iterate over source image

Some destination pixels may not be covered

**Image Warping Implementation II**

- Reverse mapping:
  
  ```c
  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:
  - \( \text{int} \ u = \text{trunc}(u+0.5); \)
  - \( \text{int} \ v = \text{trunc}(v+0.5); \)
  - \( \text{dst}(x,y) = \text{src}(u,v); \)

This method is simple, but it causes aliasing.

Filtering

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

Triangle Filtering

- Kernel is triangle function

Triangle Filtering (with width = 1)

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

Filter Width = 1

Filter Width = 2

filter width affects blurriness
Gaussian Filtering

• Kernel is Gaussian function

Image Warping Implementation

• Reverse mapping:
  for (int x = 0; x < xmax; x++) {
    for (int y = 0; y < ymax; y++) {
      float u = f⁻¹(x, y);
      float v = f⁻¹(y, x);
      dst(x, y) = resample_src(u, v, w);
    }
  }

Example: Scale

• Scale (src, dst, sx, sy):

Example: Rotate

• Rotate (src, dst, theta):

Filtering Methods Comparison

• Trade-offs
  ● Aliasing versus blurring
  ● Computation speed
Example: Fun

```
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);
    }
}
```

Swirl (src, dst, theta):  

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

Even CG folks Can Win an Oscar

Smith  Duff  Catmull  Porter

Overview: combining images

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

Image Compositing

- Separate an image into “elements”
  - Render independently
  - Composite together
- Applications
  - Cel animation
  - Chroma-keying
  - Blue-screen matting
**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.

  $\alpha = 0$

  $\alpha = 1$

  $0 < \alpha < 1$

**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 $(0, A, B, AB)$

**Color comonents are premultiplied by**

- Foreground and background pixels
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:
  - \( C' = A' + (1-\alpha_A) B' \)
  - \( \alpha = \alpha_A + (1-\alpha_A) \alpha_B \)

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

Image Composition Example

Jurassic Park

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

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

\[
blend(i,j) = (1-t) \text{src}(i,j) + t \text{dst}(i,j) \quad (0 \leq t \leq 1)
\]
**Image Morphing**
- Combines warping and cross-dissolving

H&B Figure 16.5

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

$u$ is a fraction
$v$ is a length (in pixels)

Source image: $u$ and $v$
Destination image: $u'$ and $v'$

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

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

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

  ![Image](image1.png)

**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](image2.png)

**Warping with Multiple Line Pairs**
- Use weighted combination of points defined by each pair of corresponding lines

  ![Image](image3.png)

**Warping Pseudocode**

```
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, L0[...], Image1, L1[...])
begin
    foreach intermediate frame time t do
        foreach line 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

Image_0

Warp_0

Result

Image_1

Warp_1

CS426 Examples

CS426 Class, Fall'98

Eric Schmidt, Fall'00

Beier & Neeley Example

Image_0

Warp_0

Result

Image_1

Warp_1

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