Image Warping, Compositing & Morphing

Tom Funkhouser
Princeton University
COS 426, Spring 2006

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

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 \( v \):
  - \( x = f_1(u,v) \)
  - \( y = f_2(u,v) \)

Other Mappings

- Shear in X by factor
- Shear in Y by factor
- Any function of \( u \) and \( v \)
- Any function of \( u \) and \( v \)

Image Warping Implementation I

- Forward mapping
  - Iterate over source image

Image Warping Implementation I

- Forward mapping:
  ```java
  for (int iu = 0; iu < umax; iu++) {
    for (int iv = 0; iv < vmax; iv++) {
      float x = f1(iu, iv);
      float y = f2(iu, iv);
      // ???
    }
  }
  ```

Source image  Destination image
Image Warping Implementation I

- Forward mapping:
  
  ```
  for (int iu = 0; iu < umax; iu++) {
    float x = f[iu,iv];
    float y = f[iu,iv];
    for (int ix = xlo; ix <= xhi; ix++) {
      for (int iy = ylo; iy <= yhi; iy++) {
        dst(ix,iy) += k(x,y,ix,iy); // acc(iu,iv);
      }
    }
  }
  ```

- Source image
- Destination image

Image Warping Implementation II

- Reverse mapping
  
  Iterate over destination image

```python
for (int iu = 0; iu < umax; iu++) {
    for (int iv = 0; iv < vmin; iv++) {
        float x = f[iu,iv];
        float y = f[iu,iv];
        for (int ix = xlo; ix <= xhi; ix++) {
            for (int iy = ylo; iy <= yhi; iy++) {
                dst(ix,iy) += k(x,y,ix,iy); // acc(iu,iv);
            }
        }
    }
}
```

Value at every dst pixel is weighted sum of src values
Image Warping Implementation II

• Reverse mapping:
  
  ```c
  float v = f(ix, iy);
  ```

  ```c
  for (int iv = vhi; iv < vhi; iv++) {
      dst(ix, iy) /= ksum;
  }
  ```

  ```c
  for (int iv = vhi; iv < vhi; iv++) {
      ksum += k(u, v, iu, iv);
  }
  ```

  ```c
  for (int iu = ulo; iu < uhi; iu++) {
      for (int iv = vhi; iv < vhi; iv++) {
          dat(ix, iy) += k(u, v, iu, iv) * src(u, v);
          ksum += k(u, v, iu, iv);
          src(u, v) = 0;
      }
      for (int iv = vhi; iv < vhi; iv++) {
          dat(ix, iy) /= ksum;
          ksum = 0;
      }
  }
  ```

  ```c
  for (int ly = 0; ly < ymax; ly++) {
      float ulo = u - w;
      for (int iu = ulo; iu < uhi; iu++) {
          dst(ix, iy) = 0;
          for (int iv = vhi; iv < vhi; iv++) {
              dat(ix, iy) += k(u, v, iu, iv) * src(u, v);
              ksum += k(u, v, iu, iv);
              src(u, v) = 0;
          }
          for (int iv = vhi; iv < vhi; iv++) {
              dat(ix, iy) /= ksum;
              ksum = 0;
          }
      }
  }
  ```

  ```c
  for (int ly = 0; ly < ymax; ly++) {
      float ulo = u - w;
      for (int iu = ulo; iu < uhi; iu++) {
          dst(ix, iy) = 0;
          for (int iv = vhi; iv < vhi; iv++) {
              dat(ix, iy) += k(u, v, iu, iv) * src(u, v);
              ksum += k(u, v, iu, iv);
          }
          for (int iv = vhi; iv < vhi; iv++) {
              dat(ix, iy) /= ksum;
              ksum = 0;
          }
      }
  }
  ```

  ```c
  float ReSample(src, u, v, w)
  ```

  ```c
  return k(u, v, iu, iv) * src(u, v);
  ```
Image Warping Implementation II

- Reverse mapping:
  
  ```
  Warp(src, dst) {
    for (int ix = 0; ix < xmax; ix++) {
      for (int iy = 0; iy < ymax; iy++) {
        float u = f_x(ix, iy);
        float v = f_y(ix, iy);
        dst(ix, iy) = Resample(src, u, v, w);
      }
    }
  }
  ```

Point Sampling

- Take value at closest pixel:
  
  ```
  float Resample(src, u, v, w) {
    int iu = round(u);
    int iv = round(v);
    return src(iu, iv);
  }
  ```

Filtering

- Compute weighted sum of pixel neighborhood
  - Weights are normalized values of kernel function
  - Equivalent to convolution at samples
  
  ```
  s = 0;
  for (i = -w; i <= w; i++)
    for (j = -w; j <= w; j++)
      s += k(i,j)*I(u+i, v+j);
  ```

Triangle Filtering

- Kernel is triangle function
  
  ```
  Tent Function
  ```

- Compute weighted sum of pixel neighborhood
  
  ```
  weights are normalized values of kernel function
  ```

- Equivalent to convolution at samples

  ```
  s = 0;
  for (i = -w; i <= w; i++)
    for (j = -w; j <= w; j++)
      s += k(i,j)*I(u+i, v+j);
  ```
Filtering Methods Comparison

- Trade-offs
  - Aliasing versus blurring
  - Computation speed

Point Bilinear Gaussian
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);
    }
  }
  ```
  
  - Combine images
  - Separate image into “elements”
  - Generate independently
  - Composite together
  
  - Applications
    - Gel animation
    - Chroma-keying
    - Blue-screen matting

Example: Fun

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

COS426 Examples

- Randy Carnevale
- Jonathan Heinberg
- Sid Kapur
- Philip Wei

Image Processing

- Quantization
  - Uniform Quantization
  - 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 Compositing

- Combine images

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.

**Semi-Transperent 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

**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


Jurassic Park

Even CG folks Can Win an Oscar

COS426 Examples

Darin Sleiter

Kenrick Kin

Smith  Duff  Catmull  Porter
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

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

- Combines warping and cross-dissolving

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

Feature-Based Warping

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

H&B Figure 16.9
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:

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

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

```
WarpImage(Image, L[i], L[j])
begin
    foreach destination pixel p do
        psun = (0,0)
        wsum = 0
        foreach line L[i] in destination do
            p[i] = p transformed by (L[i], L[j])
            psun = psun + p[i] * weight[i]
            wsum += weight[i]
        end
        p' = psun / 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 i-th of the way from L_0[i] to L_1[i]
        end
        Warp_0 = WarpImage(Image0, L)
        Warp_1 = WarpImage(Image_1, L)
        foreach pixel p in FinalImage do
            Result(p) = (1-t) Warp_0 + t Warp_1
        end
    end
end
```

Beier & Neeley Example

- Image0
- Warp_0
- Result
- Image1
- Warp_1

COS426 Examples

- CS426 Class, Fall98
- Jon Beyer
Summary

- Image warping
  - Mapping
  - Resampling
- Image compositing
  - Alpha channel
  - Porter-Duff compositing algebra
- Image morphing
  - Specifying correspondences
  - Warping
  - Compositing

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