Image Warping

Tom Funkhouser
Princeton University
COS 426, Spring 2007

Image Processing

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

Issues:
- How do we specify where every pixel goes? (mapping)
- How do we compute colors at dest pixels? (resampling)
Example

- Image scaling:
  \( (x', y') = (sx, sy) \)
  \( I(x', y') = ?? \)

Image Resampling Pipeline

- Resampling requires bandlimiting function in order to avoid aliasing artifacts

Image Resampling

- Convolution with a triangle filter:

Image Warping

- Image warping requires resampling of image

Image Resampling Pipeline

- Resampling requires convolution with low-pass filter in order to reduce aliasing artifacts (in practice)
**Image Resampling**

- Convolution with a triangle filter:

![Image of convolution with a triangle filter](image)

**Input**

**Output**

- What if the convolution runs off the end?

![Image of convolution running off the end](image)

**Input**

**Output**

- What if output sample is between input samples?

![Image of output sample between input samples](image)

**Input**

**Output**
Image Resampling

• What if scale factor is smaller (e.g., 1/3)?

Input
Output

Figure 2.4 Wolberg

Image Resampling

• What if scale factor is greater than one?

Input
Output

Figure 2.4 Wolberg

Image Resampling

• What is we use a Gaussian filter?

Input
Output

Figure 2.4 Wolberg

Image Resampling

• What if we are resampling a 2D image?
  - Same ideas

same ideas

Figure 2.4 Wolberg

Image Resampling

• Compute weighted sum of pixel neighborhood
  - Output is weighted average of input, where weights are normalized values of filter kernel (k)

\[
d_{st}(ix,iy) = 0; \\
\text{for } (ix = u-w; ix < u+w; ix++) \\
\text{for } (iy = v-w; iy < v+w; iy++) \\
d = dist((ix,iy)*pix(x,y)); \\
dst(ix,iy) += k(ix,iy)*src(ix,iy); \\
\text{filter values represented by gray value}
\]

[df(x,y)] represented by gray value

Figure 2.4 Wolberg
**Image Resampling**

- For isotropic Triangle and Gaussian filters, \( k(i_x, i_y) \) is function of \( r \) and \( w \)

```
Point Sampling
for (int iy = 0; iy < ymax; iy++) {
    1 2
    1 2
    1 2
1 2 1 2 1

1 2 1 2 1

1 2 1 2 1
```

**Triangle Filtering (with width <= 1)**

- 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 \( "b" \)

```
Triangle filter
k(i_j) = 1 - d / w
```

**Gaussian Filtering**

- Kernel is Gaussian function

```
\begin{align*}
G_d(d) &= 2^{-d^2/2} \\
\end{align*}
```

**Image Scale**

- Possible scale implementation:

```
float Resample(src, u, v, w) {
    int iu = round(u);
    int iv = round(v);
    return src(iu, iv);
}
```

```
\begin{align*}
A 1 2 3 4 5 \\
1 2 3 4 5 \\
2 3 4 5 6 \\
3 4 5 6 7 \\
4 5 6 7 8 \\
5 6 7 8 9 \\
\end{align*}
```

**Point Sampling**

- Possible (poor) resampling implementation:

```
float Resample(src, u, v, w) {
    int iu = round(u);
    int iv = round(v);
    return src(iu, iv);
}
```

```
\begin{align*}
0 & \quad 0 \quad 0 \quad 0 \quad 0 \\
0 & \quad 0 \quad 0 \quad 0 \quad 0 \\
0 & \quad 0 \quad 0 \quad 0 \quad 0 \\
0 & \quad 0 \quad 0 \quad 0 \quad 0 \\
0 & \quad 0 \quad 0 \quad 0 \quad 0 \\
\end{align*}
```
Point Triangle Gaussian Image Warping (in General)

- Better resampling implementation:
  ```
  float Resample(src, u, v, k, w)
  {
    float dat = 0;
    float ksum = 0;
    int iu = u - w; iv = v - w;
    for (int iu = u0; iu < uh; iu++)
      for (int iv = v0; iv < vh; iv++)
        dat += k(u, v, iu, iv, w) * src(u, v);
    ksum += k(u, v, iu, iv, w);
  }
  return dat / ksum;
  }
  ```

Image Blur

- Possible blur implementation:
  ```
  float src01[32][32];
  float dst01[32][32];
  float w = 3 * sigma;
  for (int ix = 0; ix < xmax; ix++)
    for (int iy = 0; iy < ymax; iy++)
      float u = ix;
      float v = iy;
      float u1 = f(ix + 1, iy);
      float v1 = f(ix, iy + 1);
      float u2 = f(ix - 1, iy);
      float v2 = f(ix, iy - 1);
      float x = -0.5 + 0.5 * u;
      float y = -0.5 + 0.5 * v;
      float dx = x - u;
      float dy = y - v;
      float dsrc = dst[ix, iy] = Resample(src, u, v, k, w);
  ```

Image Rotation

- Possible rotation implementation:
  ```
  float u = f(x, y);
  float v = f(y, x);
  Rotate(src, dst, @)
  ```

Triangle and Gaussian Sampling

- Possible warp implementation (reverse mapping):
  ```
  Warp(src, dst)
  {
    for (int ix = 0; ix < xmax; ix++)
      for (int iy = 0; iy < ymax; iy++)
        float u = f[ix, iy];
        float v = f[y, ix];
        float w = 1 / scale;
        dat[ix, iy] = Resample(src, u, v, w);
  }
  ```

Image Warping (in General)

- Possible warp implementation (forward mapping):
  ```
  Warp(src, dst)
  {
    for (int ix = 0; ix < xmax; ix++)
      for (int iy = 0; iy < ymax; iy++)
        float x = f[ix, iy];
        float y = f[y, ix];
        float w = 1 / scale;
        Splat(src[ix, iy], x, y, w);
  }
  ```

Sampling Method Comparison

- Trade-offs
  - Aliasing versus blurring
  - Computation speed

Point Triangle Gaussian
**Image Warping (in General)**

- Alternative implementation (forward mapping):
  
  ```
  Warp(src, dst) {
      for (int iu = 0; iu < umax; iu++) {
         for (int iv = 0; iv < vmax; iv++) {
            float x = fx(iu, iv);
            float y = fy(iu, iv);
            float w = 1 / scale(x, y);
            Splat(src(iu, iv), x, y, w);   // weighting ???
         }
      }
  }
  ```

**Image Processing**

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

**Mapping**

- Define transformation
  - Describe the destination (x,y) for every source (u,v) (actually vice-versa, if reverse mapping)

**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 + factor * v
  - y = v
  [Shear X 1.3]

• Shear in Y by factor:
  - x = u
  - y = v + factor * u
  [Shear Y 1.3]

Other Parametric Mappings

• Any function of u and v:
  - x = f_1(u,v)
  - y = f_2(u,v)
  [Fish-eye “Swirl”]

Point Correspondence Mappings

• Mappings implied by correspondences:
  - A ↔ A'
  - B ↔ B'
  - C ↔ C'

  [Warp]  [Warp]

Point Correspondence Mappings

• How compute P' from:
  - A ↔ A'
  - B ↔ B'
  - C ↔ C'
  - P

  [Warp]  [Warp]

Point Correspondence Mappings

• How compute P' from:
  - A ↔ A'
  - B ↔ B'
  - C ↔ C'
  - P

  [Warp]  [Warp]
Point Correspondence Mappings

• How compute $P'$ from:
  - $X' \leftrightarrow X' \rightarrow P = \sum w_i X_i$
  - $P$  

Radial Basis Functions
Thin-Plate Splines

Line Correspondence Mappings

• Beier & Neeley use pairs of lines to specify warp

Warping with One Line Pair

• What happens to the “F”?  

Translation!

Scale!

Rotation!
Warping with One Line Pair

- What happens to the “F”?

```
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

```
Mapping
```

```
Source image
```

```
Destination image
```

```
p’ is a weighted average
```

```
Weighting Effect of Each Line Pair
```

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

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

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

Putting It All Together

- Warping with correspondences

```
Warped(src, dst, correspondences) {
    for (int ix = 0; ix < xmax; ix++) {
        for (int iy = 0; iy < ymax; iy++) {
            float u = f / scale(ix, iy);
            float v = f / scale(ix, iy);
            float w = \frac{1}{\text{scale}(ix, iy)};
            float \( u' \) = f / \text{scale}(ix, iy);
            float \( v' \) = f / \text{scale}(ix, iy);
            float \( w' \) = \frac{1}{\text{scale}(ix, iy)};
            \text{dat}(ix, iy) = \text{Resample}(src, u, v, w);
        }
    }
}
```

- Other fun warps:

```
Swirl(src, dst, \theta) {
    for (int ix = 0; ix < xmax; ix++) {
        for (int iy = 0; iy < ymax; iy++) {
            float \( u \) = \text{rot}(\text{dist}(ix, xcenter) * \theta);
            float \( v \) = \text{rot}(\text{dist}(iy, ycenter) * \theta);
            \text{dat}(ix, iy) = \text{Resample}(src, u, v, \theta);
        }
    }
}
```

```
Swirl
```

```
45
```

```
x
```

```
y
```

```
u
```

```
v
```

```
Warping with Multiple Line Pairs
```

```
Mapping
```

```
Source image
```

```
Destination image
```

```
p’ is a weighted average
```

```
Putting It All Together
```

```
Warped(src, dst, correspondences) {
    for (int ix = 0; ix < xmax; ix++) {
        for (int iy = 0; iy < ymax; iy++) {
            float u = f / scale(ix, iy);
            float v = f / scale(ix, iy);
            float w = \frac{1}{\text{scale}(ix, iy)};
            float \( u' \) = f / \text{scale}(ix, iy);
            float \( v' \) = f / \text{scale}(ix, iy);
            float \( w' \) = \frac{1}{\text{scale}(ix, iy)};
            \text{dat}(ix, iy) = \text{Resample}(src, u, v, w);
        }
    }
}
```

```
Swirl(src, dst, \theta) {
    for (int ix = 0; ix < xmax; ix++) {
        for (int iy = 0; iy < ymax; iy++) {
            float \( u \) = \text{rot}(\text{dist}(ix, xcenter) * \theta);
            float \( v \) = \text{rot}(\text{dist}(iy, ycenter) * \theta);
            \text{dat}(ix, iy) = \text{Resample}(src, u, v, \theta);
        }
    }
}
```

```
Swirl
```

```
45
```

```
x
```

```
y
```

```
u
```

```
v
```
More COS426 Examples

- Randy Carnivale
- Sid Kapur
- Wei Xiang
- Paul Nelson

Summary

- Resampling
  - Triangle filter
  - Gaussian filter
- Mapping
  - Parametric
  - Correspondences
- Image processing
  - Reverse mapping
  - Forward mapping