Sampling, Resampling, and Warping

COS 426
Digital Image Processing

- Changing intensity/color
  - Linear: scale, offset, etc.
  - Nonlinear: gamma, saturation, etc.
  - Add random noise

- Filtering over neighborhoods
  - Blur
  - Detect edges
  - Sharpen
  - Emboss
  - Median

- Moving image locations
  - Scale
  - Rotate
  - Warp

- Combining images
  - Composite
  - Morph

- Quantization

- Spatial / intensity tradeoff
  - Dithering
Digital Image Processing

When implementing operations that move pixels, must account for the fact that digital images are sampled versions of continuous ones.
Sampling and Reconstruction

- Continuous function
- Discrete samples
- Sampling
Sampling and Reconstruction

Sampling

Continuous function

Discrete samples

Reconstruction

Continuous function
Sampling and Reconstruction

Figure 19.9 FvDFH
Sampling Theory

How many samples are enough?

- How many samples are required to represent a given signal without loss of information?
- What signals can be reconstructed without loss for a given sampling rate?
Sampling Theory

What happens when we use too few samples?

- **Aliasing**: high frequencies masquerade as low ones

Figure 14.17 FvDFH
Sampling Theory

What happens when we use too few samples?

- **Aliasing**: high frequencies masquerade as low ones
Sampling Theory

What happens when we use too few samples?

- **Aliasing**: high frequencies masquerade as low ones

(Barely) adequate sampling

Inadequate sampling
Sampling Theory

How many samples are enough to avoid aliasing?

- How many samples are required to represent a given signal without loss of information?
- What signals can be reconstructed without loss for a given sampling rate?
Sampling Theory

How many samples are enough to avoid aliasing?

- How many samples are required to represent a given signal without loss of information?
- What signals can be reconstructed without loss for a given sampling rate?
Sampling Theory

How many samples are enough to avoid aliasing?

- How many samples are required to represent a given signal without loss of information?
- What signals can be reconstructed without loss for a given sampling rate?
Sampling Theory

How many samples are enough to avoid aliasing?

- How many samples are required to represent a given signal without loss of information?
- What signals can be reconstructed without loss for a given sampling rate?
Sampling Theory

How many samples are enough to avoid aliasing?

- How many samples are required to represent a given signal without loss of information?
- What signals can be reconstructed without loss for a given sampling rate?
Spectral Analysis

- **Spatial domain:**
  - Function: $f(x)$
  - Filtering: convolution

- **Frequency domain:**
  - Function: $F(u)$
  - Filtering: multiplication

Any signal can be written as a sum of periodic functions.
Fourier Transform

Figure 2.6 Wolberg
Fourier Transform

• Fourier transform:

\[ F(u) = \int_{-\infty}^{\infty} f(x)e^{-i2\pi xu} \, dx \]

• Inverse Fourier transform:

\[ f(x) = \int_{-\infty}^{\infty} F(u)e^{+i2\pi ux} \, du \]
Sampling Theorem

• A signal can be reconstructed from its samples, iff the original signal has no content $\geq 1/2$ the sampling frequency - Shannon

• The minimum sampling rate for bandlimited function is called the “Nyquist rate”

A signal is bandlimited if its highest frequency is bounded. The frequency is called the bandwidth.
Image Processing

• Consider reducing the image resolution

Original image

1/4 resolution
Image Processing

- Image processing is a **resampling** problem
Sampling Theorem

- A signal can be reconstructed from its samples, iff the original signal has no content \( \geq 1/2 \) the sampling frequency - Shannon

**Aliasing** will occur if the signal is under-sampled
Aliasing

• In general:
  ○ Artifacts due to under-sampling or poor reconstruction

• Specifically, in graphics:
  ○ Spatial aliasing
  ○ Temporal aliasing

Figure 14.17 FvDFH
Spatial Aliasing

Artifacts due to limited spatial resolution
Spatial Aliasing

Artifacts due to limited spatial resolution

“Jaggies”
Temporal Aliasing

Artifacts due to limited temporal resolution

- Strobing
- Flickering
Temporal Aliasing

Artifacts due to limited temporal resolution

- Strobing
- Flickering
Temporal Aliasing

Artifacts due to limited temporal resolution

- Strobing
- Flickering
Temporal Aliasing

Artifacts due to limited temporal resolution

- Strobing
- Flickering
Antialiasing

- Sample at higher rate
  - Not always possible
  - Doesn’t always solve the problem

- Pre-filter to form bandlimited signal
  - Use low-pass filter to limit signal to $< 1/2$ sampling rate
  - Trades blurring for aliasing
Image Processing

Real world

Sample

Discrete samples (pixels)

Reconstruct

Reconstructed function

Transform

Transformed function

Filter

Bandlimited function

Sample

Discrete samples (pixels)

Reconstruct

Display
Image Processing

Real world

Sample

Discrete samples (pixels)

Reconstruct

Reconstructed function

Transform

Transformed function

Filter

Bandlimited function

Sample

Discrete samples (pixels)

Reconstruct

Display

Continuous Function
Image Processing

Real world
Sample
Discrete samples (pixels)
Reconstruct
Reconstructed function
Transform
Transformed function
Filter
Bandlimited function
Sample
Discrete samples (pixels)
Reconstruct
Display

Discrete Samples
Image Processing

- Real world
  - Sample → Discrete samples (pixels)
  - **Reconstruct** → Reconstructed function
    - Transform → Transformed function
      - Filter → Bandlimited function
        - Sample → Discrete samples (pixels)
          - **Reconstruct** → Display
Image Processing

Real world

Sample
Discrete samples (pixels)
Reconstruct
Reconstructed function
Transform
Transformed function
Filter
Bandlimited function
Sample
Discrete samples (pixels)
Reconstruct
Display
Transformed Function
Image Processing

Real world

Sample

Discrete samples (pixels)

Reconstruct

Reconstructed function

Transform

Transformed function

Filter

Bandlimited function

Sample

Discrete samples (pixels)

Reconstruct

Display

Bandlimited Function
Image Processing

Real world

Sample
Discrete samples (pixels)

Reconstruct
Reconstructed function

Transform
Transformed function

Filter
Bandlimited function

Sample
Discrete samples (pixels)

Reconstruct
Display

Discrete samples
Image Processing

Real world

Sample
- Discrete samples (pixels)

Reconstruct
- Reconstructed function

Transform
- Transformed function

Filter
- Bandlimited function

Sample
- Discrete samples (pixels)

Reconstruct
- Display
Ideal Bandlimiting Filter

- Frequency domain

- Spatial domain

\[ Sinc(x) = \frac{\sin \pi x}{\pi x} \]

Figure 4.5 Wolberg
Practical Image Processing

- Finite low-pass filters
  - Point sampling (bad)
  - Box filter
  - Triangle filter
  - Gaussian filter

Real world

Sample

Discrete samples (pixels)

Reconstruct

Reconstructed function

Transform

Transformed function

Filter

Bandlimited function

Sample

Discrete samples (pixels)

Reconstruct

Display
Example: Scaling

- Resample with triangle or Gaussian filter

Original

1/4X resolution

4X resolution
General Image Warping

- Move pixels of an image

Source image  \[\rightarrow\]  Warp  \[\rightarrow\]  Destination image
General Image Warping

• Issues:
  ◦ Specifying where every pixel goes (mapping)
General Image Warping

• Issues:
  ◦ Specifying where every pixel goes (mapping)
  ◦ Computing colors at destination pixels (resampling)
General Image Warping

• Issues:
  - Specifying where every pixel goes (mapping)
    ◦ Computing colors at destination pixels (resampling)
Two Options

- **Forward mapping**

- **Reverse mapping**
Mapping

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

- Scale by \( \text{factor} \):
  - \( x = \text{factor} \times u \)
  - \( y = \text{factor} \times v \)

\[ \text{Scale 0.8} \]
Parametric Mappings

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

- Any function of $u$ and $v$:
  - $x = f_x(u,v)$
  - $y = f_y(u,v)$

Fish-eye

“Swirl”

“Rain”
COS426 Examples

Aditya Bhaskara

Wei Xiang
More COS426 Examples

Sid Kapur

Michael Oranato

Eirik Bakke
Point Correspondence Mappings

- Mappings implied by correspondences:
  - $A \leftrightarrow A'$
  - $B \leftrightarrow B'$
  - $C \leftrightarrow C'$
Line Correspondence Mappings

• Beier & Neeley use pairs of lines to specify warp
Image Warping

• Issues:
  ◦ Specifying where every pixel goes (mapping)
  ➢ Computing colors at destination pixels (resampling)
Image Warping

- Image warping requires resampling of image

Resampling
Point Sampling

- Possible (poor) resampling implementation:

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

- Use nearest sample
Point Sampling

Point Sampled: Aliasing!  Correctly Bandlimited
Image Resampling Pipeline

- Ideal resampling requires correct filtering to avoid artifacts.
  - Reconstruction filter especially important when magnifying.
  - Bandlimiting filter especially important when minifying.
In practice: Resampling with low-pass filter in order to reduce aliasing artifacts when minifying.
Resampling with Filter

- Output is weighted average of inputs:

```c
float Resample(src, u, v, k, w)
{
    float dst = 0;
    float ksum = 0;
    int ulo = u - w; etc.
    for (int iu = ulo; iu < uhi; iu++) {
        for (int iv = vlo; iv < vhi; iv++) {
            dst += k(u,v,iu,iv,w) * src(u,v)
            ksum += k(u,v,iu,iv,w);
        }
    }
    return dst / ksum;
}
```
Image Resampling

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

\[
dst(ix,iy) = 0; \\
\text{for (ix} = u-w; ix <= u+w; ix++) \\
\quad \text{for (iy} = v-w; iy <= v+w; iy++) \\
\quad \quad d = \text{dist (ix,iy)}\leftrightarrow(u,v) \\
\quad \quad dst(ix,iy) += k(ix,iy)^{*}\text{src(ix,iy)};
\]

\(k(ix,iy)\) represented by gray value
Image Resampling

• For isotropic Triangle and Gaussian filters, $k(ix, iy)$ is function of $d$ and $w$

Triangle filter

$$k(i, j) = \max(1 - d/w, 0)$$
Image Resampling

• For isotropic Triangle and Gaussian filters, $k(ix,iy)$ is function of $d$ and $w$
  ○ Filter width chosen based on scale factor (or blur)

![Diagram of Triangle filter](image)

Filter Width = 1

Triangle filter

Width of filter affects blurriness
Gaussian Filtering

- Kernel is Gaussian function

\[ G(d, \sigma) = e^{-d^2/(2\sigma^2)} \]

- Drops off quickly, but never gets to exactly 0
- In practice: compute out to \( w \approx 2.5\sigma \) or \( 3\sigma \)
Image Resampling

- What if width (w) is smaller than sample spacing?
Image Resampling (with width < 1)

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

Filter Width < 1
Image Resampling (with width < 1)

• Alternative: force width to be at least 1

Filter Width < 1
• Possible implementation of image blur:

\[
\text{Blur(src, dst, sigma) \{ } \\
\quad w \approx 3*\text{sigma}; \\
\quad \text{for (int ix = 0; ix < xmax; ix++) \{ } \\
\quad \quad \text{for (int iy = 0; iy < ymax; iy++) \{ } \\
\quad \quad \quad \text{float u = ix; } \\
\quad \quad \quad \text{float v = iy; } \\
\quad \quad \quad \text{dst(ix,iy) = Resample(src,u,v,k,w); } \\
\quad \quad \} \\
\quad \} \\
\text{\}}
\]
Putting it All Together

• Possible implementation of image scale:

\[
\text{Scale}(\text{src}, \text{dst}, sx, sy) \{
    w \approx \max(1/sx, 1/sy);
    \text{for} (\text{int} \ ix = 0; \ ix < \text{xmax}; \ ix++) \{
        \text{for} (\text{int} \ iy = 0; \ iy < \text{ymax}; \ iy++) \{
            \text{float} \ u = \ ix / \ sx;
            \text{float} \ v = \ iy / \ sy;
            \text{dst}(ix,iy) = \text{Resample}(\text{src},u,v,k,w);
        \}
    \}
}\]

Source image  \quad (u,v)  \quad f  \quad (ix,iy)  \quad \text{Destination image}
Putting it All Together

• Possible implementation of image rotation:

    Rotate(src, dst, Θ) {
        w ≈ 1
        for (int ix = 0; ix < xmax; ix++) {
            for (int iy = 0; iy < ymax; iy++) {
                float u = ix*cos(-Θ) – iy*sin(-Θ);
                float v = ix*sin(-Θ) + iy*cos(-Θ);
                dst(ix,iy) = Resample(src,u,v,k,w);
            }
        }
    }
Sampling Method Comparison

• Trade-offs
  ◦ Aliasing versus blurring
  ◦ Computation speed

Point  Triangle  Gaussian
Forward vs. Reverse Mapping

• Reverse mapping:

\[
\text{Warp}(\text{src}, \text{dst}) \{ \\
\quad \text{for (int } ix = 0; ix < xmax; ix++) \{ \\
\quad \quad \text{for (int } iy = 0; iy < ymax; iy++) \{ \\
\quad \quad \quad \text{float } w \approx 1 / \text{scale}(ix, iy); \\
\quad \quad \quad \text{float } u = f_x^{-1}(ix,iy); \\
\quad \quad \quad \text{float } v = f_y^{-1}(ix,iy); \\
\quad \quad \quad \text{dst}(ix,iy) = \text{Resample}(\text{src},u,v,w); \\
\quad \quad \} \\
\quad \} \\
\}
\]

Source image \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \ Quad
Forward vs. Reverse Mapping

- **Forward mapping:**

  ```
  Warp(src, dst) {
      for (int iu = 0; iu < umax; iu++) {
          for (int iv = 0; iv < vmax; iv++) {
              float x = f_x(iu,iv);
              float y = f_y(iu,iv);
              float w = 1 / scale(x, y);
              Splat(src(iu,iv),x,y,k,w);
          }
      }
  }
  ```

  ![Source image](source-image.png) ![Destination image](destination-image.png)

  - **Source image**
  - **Destination image**
Forward vs. Reverse Mapping

- Forward mapping:

  ```
  Warp(src, dst) {
    for (int iu = 0; iu < umax; iu++) {
      for (int iv = 0; iv < vmax; iv++) {
        float x = f_x(iu, iv);
        float y = f_y(iu, iv);
        float w ≈ 1 / scale(x, y);
        Splat(src(iu, iv), x, y, k, w);
      }
    }
  }
  ```

  (iu,iv) -> (x,y)

Source image  Destination image
Forward vs. Reverse Mapping

• Forward mapping:

```c
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);
        for (int ix = xlo; ix <= xhi; ix++) {
            for (int iy = ylo; iy <= yhi; iy++) {
                dst(ix, iy) += k(x, y, ix, iy, w) * src(iu, iv);
            }
        }
    }
}
```

Problem?
Forward vs. Reverse Mapping

- Forward mapping:

```c
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);
        for (int ix = xlo; ix <= xhi; ix++) {
            for (int iy = ylo; iy <= yhi; iy++) {
                dst(ix,iy) += k(x,y,ix,iy,w) * src(iu,iv);
                ksum(ix,iy) += k(x,y,ix,iy,w);
            }
        }
    }
}

for (ix = 0; ix < xmax; ix++)
    for (iy = 0; iy < ymax; iy++)
        dst(ix,iy) /= ksum(ix,iy)
```

Destination image $(x,y)$
Forward vs. Reverse Mapping

• Tradeoffs?
Forward vs. Reverse Mapping

- **Tradeoffs:**
  - **Forward mapping:**
    - Requires separate buffer to store weights
  - **Reverse mapping:**
    - Requires inverse of mapping function, random access to original image
Summary

• Mapping
  ○ Forward vs. reverse
  ○ Parametric vs. correspondences

• Sampling, reconstruction, resampling
  ○ Frequency analysis of signal content
  ○ Filter to avoid undersampling: point, triangle, Gaussian
  ○ Reduce visual artifacts due to aliasing
    » Blurring is better than aliasing
Next Time…

• Changing intensity/color
  - Linear: scale, offset, etc.
  - Nonlinear: gamma, saturation, etc.
  - Add random noise

• Filtering over neighborhoods
  - Blur
  - Detect edges
  - Sharpen
  - Emboss
  - Median

• Moving image locations
  - Scale
  - Rotate
  - Warp

• Combining images
  - Composite
  - Morph

• Quantization
  - Spatial / intensity tradeoff
  - Dithering