Digital Image Processing

• Changing pixel values
  ◦ Linear: scale, offset, etc.
  ◦ Nonlinear: gamma, saturation, etc.
  ◦ Histogram equalization

• Filtering over neighborhoods
  ◦ Blur & sharpen
  ◦ Detect edges
  ◦ Median
  ◦ Bilateral filter

• Moving image locations
  ◦ Scale
  ◦ Rotate
  ◦ Warp

• Combining images
  ◦ Composite
  ◦ Morph

• Quantization

• Spatial / intensity tradeoff
  ◦ Dithering
Image Warping

• Move pixels of an image
Image Warping

• **Issues:**
  - Specifying where every pixel goes (**mapping**)

![Diagram showing source and destination images with warping](image.png)
Image Warping

• Issues:
  ◦ Specifying where every pixel goes (mapping)
  ◦ Computing colors at destination pixels (resampling)
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 \((u,v)\) (vice-versa, if reverse mapping)
Parametric Mappings

- Scale by factor:
  - $x = \text{factor} \times u$
  - $y = \text{factor} \times v$
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*:**
  - $x = u + \textit{factor} \times v$
  - $y = v$

- **Shear in Y by *factor*:**
  - $x = u$
  - $y = v + \textit{factor} \times u$

Shear X
- $\textit{factor} = 0.3$

Shear Y
- $\textit{factor} = 0.3$
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

• Alternatively, Beier & Neeley [92] use pairs of *lines* to specify warp (more on this next time)
Image Warping

• Issues:
  ○ Specifying where every pixel goes (mapping)
  ○ Computing colors at destination pixels (resampling)
When implementing operations that move pixels, must account for the fact that digital images are \textit{sampling} 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

Original signal

Sampling

Sampled signal

Reconstruction

Reconstructed signal

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?

• What happens when we use too few samples?
Sampling Theory

• What happens when we use too few samples?
  ○ **Aliasing:** high frequencies masquerade as low ones

• 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

(Barely) adequate sampling

Inadequate sampling
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
  - Flickering
  - Strobing ("Backwards spinning wheel" effect)
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?
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 it has no content $\geq \frac{1}{2}$ the sampling frequency – Shannon

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

A signal is *bandlimited* if its highest frequency is bounded. That frequency is called the bandwidth.
Why $>?$

- Sampling rate must be $> 2$ bandwidth

Adequate?
• Sampling rate must be $> 2$ bandwidth
Anti-aliasing

- 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

- Consider scaling the image (or, equivalently, reducing resolution)
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

Reconstructed Function
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

1. Real world
2. Sample
   - Discrete samples (pixels)
3. Reconstruct
   - Reconstructed function
4. Transform
   - Transformed function
5. Filter
   - Bandlimited function
6. Sample
   - Discrete samples (pixels)
7. 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

- Ideal resampling requires correct filtering to avoid artifacts

- **Reconstruction** filter especially important when **magnifying**

- **Bandlimiting** filter especially important when **minifying**
Ideal Image Processing Filter

• Frequency domain (multiplication)

• Spatial domain (convolution)

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

Retain these frequencies
Remove these frequencies

Figure 4.5 Wolberg
Practical Image Processing

Resampling

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

- **Resampling**: effectively (discrete) convolution to prevent artifacts
- **Finite low-pass filters**
  - Point sampling (bad)
  - Box filter
  - Triangle filter
  - Gaussian filter
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);
}
```

![Diagram](image.png)
Point Sampling

• Use nearest sample
Point Sampling

Point Sampled: Aliasing!  Correctly Bandlimited
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;
}
```

(f(u,v), (ix,iy))
Image Resampling

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

\[ k(ix,iy) \text{ represented by gray value} \]

\[ (u,v) \]

\[ (iu,iv) \]
Image Resampling

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

\[ k(i,j) = \max(1 - \frac{d}{w}, 0) \]

Triangle filter

Filter Width = 2
Image Resampling

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

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

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

- Filter width chosen based on scale factor (or blur)

Filter Width = 1

Triangle filter

Width of filter affects blurriness
Image Resampling

- What if width (w) is smaller than sample spacing?

Filter Width < 1

Triangle filter
Image Resampling (with width < 1)

- Reconstruction filter: 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”
Image Resampling (with width < 1)

• Alternative: force width to be at least 1

Filter Width = 1
Putting it All Together

- Possible implementation of image scale:

```plaintext
Scale(src, dst, sx, sy) {
    w ≈ max(1/sx, 1/sy, 1);
    for (int ix = 0; ix < xmax; ix++) {
        for (int iy = 0; iy < ymax; iy++) {
            float u = ix / sx;
            float v = iy / sy;
            dst(ix,iy) = Resample(src,u,v,k,w);
        }
    }
}
```

(u,v) (ix,iy)

Source image Destination image
Putting it All Together

- Possible implementation of image rotation:

\[
\text{Rotate}(src, \text{dst}, \Theta) \{
\begin{align*}
  w & \approx 1; \\
  & \text{for (int } \text{ix} = 0; \text{ix} < \text{xmax}; \text{ix}++) \{ \\
  & \text{for (int } \text{iy} = 0; \text{iy} < \text{ymax}; \text{iy}++) \{ \\
  & \quad \text{float } u = \text{ix} \ast \cos(-\Theta) - \text{iy} \ast \sin(-\Theta); \\
  & \quad \text{float } v = \text{ix} \ast \sin(-\Theta) + \text{iy} \ast \cos(-\Theta); \\
  & \quad \text{dst}(\text{ix},\text{iy}) = \text{Resample}(src, u, v, k, w); \\
  & \} \\
  & \} \\
\end{align*}
\]
Sampling Method Comparison

• Trade-offs
  ○ Aliasing versus blurring
  ○ Computation speed

Point  Triangle  Gaussian
Forward vs. Reverse Mapping

- Reverse mapping:

```
Warp(src, dst) {
    for (int ix = 0; ix < xmax; ix++) {
        for (int iy = 0; iy < ymax; iy++) {
            float w = 1 / scale(ix, iy);
            float u = f_x^{-1}(ix, iy);
            float v = f_y^{-1}(ix, iy);
            dst(ix,iy) = Resample(src,u,v,w);
        }
    }
}
```
Forward vs. Reverse Mapping

- Forward mapping:

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

\[(iu,iv) \rightarrow (x,y)\]

Source image \rightarrow Destination image
Forward vs. Reverse Mapping

- Forward mapping:

```c
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);
            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);
                }
            }
        }
    }
}
```
Forward vs. Reverse 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);
            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)
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 pixel values
  - Linear: scale, offset, etc.
  - Nonlinear: gamma, saturation, etc.
  - Histogram equalization

- Filtering over neighborhoods
  - Blur & sharpen
  - Detect edges
  - Median
  - Bilateral filter

- Moving image locations
  - Scale
  - Rotate
  - Warp

- Combining images
  - Composite
  - Morph

- Quantization

- Spatial / intensity tradeoff
  - Dithering