“Your x-ray showed a broken rib, but we fixed it with Photoshop.”
What’s on our plate today?

• Image formation
• Image sampling and quantization
• Image interpolation
• Domain transformations
  • Affine image transformations
• Range (intensity) transformations
  • Noise reduction through spatial filtering
  • Filtering as cross-correlation
  • Convolution
  • Nonlinear (median) filtering
Image Formation: Basics

(From Gonzalez & Woods, 2008)
Image Formation: Basics

Image f(x,y) is characterized by 2 components

1. **Illumination** \(i(x,y)\) = Amount of source illumination incident on scene

2. **Reflectance** \(r(x,y)\) = Amount of illumination reflected by objects in the scene

\[
f(x, y) = i(x, y)r(x, y)
\]

where

\[
0 < i(x, y) < \infty \text{ and } 0 < r(x, y) < 1
\]

\(r(x,y)\) depends on object properties
\(r = 0\) means total absorption and \(1\) means total reflectance
Image Formation: Basics

\[ f(x, y) = i(x, y) r(x, y) \]

where

\[ 0 < i(x, y) < \infty \text{ and } 0 < r(x, y) < 1 \]

Typical values of \( i(x,y) \):
- Sun on a clear day: 90,000 lm/m²
- Cloudy day: 10,000 lm/m²
- Inside an office: 1000 lm/m²

Typical values of \( r(x,y) \)
- Black velvet: 0.01, Stainless steel: 0.65, Snow: 0.93

Typical limits of \( f(x,y) \) in an office environment
- \( 10 < f(x,y) < 1000 \)
- Shifted to gray scale \([0, L-1]\); 0 = black, L-1 = 255 = white
Sampling and Quantization Process

(from Gonzalez & Woods, 2008)
Example of a Quantized 2D Image

Continuous image projected onto sensor array

Result of sampling and quantization

(from Gonzalez & Woods, 2008)
Suppose we want to zoom an image

Original image

Need to fill in values for new pixels

Zoomed image
Interpolation

Original

Zoomed

Need to fill in missing values *

Nearest Neighbor Interpolation

For each new pixel, copy nearest value
Nearest Neighbor Interpolation

Original image

Zoomed image

Can we do better?
Other image interpolation techniques

**Bilinear interpolation:**
Compute pixel value \( v(x,y) \) as:

\[
v(x, y) = ax + by + cxy + d
\]

\( a, b, c, d \) determined from **four nearest neighbors** of \((x,y)\)

**Bicubic interpolation:**
(Used in most commercial image editing programs, e.g., Photoshop)

\[
v(x, y) = \sum_{i=0}^{3} \sum_{j=0}^{3} a_{ij} x^i y^j
\]

\( a_{ij} \) determined from **16 nearest neighbors** of \((x,y)\)

(From http://www.cambridgeincolour.com/tutorials/image-interpolation.htm)

Comparison of Interpolation Techniques

Nearest Neighbor  Bilinear  Bicubic
Recall from Last Time

**Domain transformation**: $g(x, y) = f(t_x(x, y), t_y(x, y))$

(What is an example?)

Translation

Rotation

How are these done?
Geometric spatial transformations of images

Two steps:
1. Spatial transformation of coordinates \((x, y)\)
2. Interpolation of intensity value at new coordinates

We already know how to do (2), so focus on (1)

Example: What does the transformation 
\((x, y) = T((v, w)) = (v/2, w/2)\) do?

[Shrinks original image in half in both directions]
Affine Spatial Transformations

- Most commonly used set of transformations
- General form:

\[
\begin{bmatrix}
  x \\
  y \\
  1
\end{bmatrix} =
\begin{bmatrix}
  v \\
  w \\
  1
\end{bmatrix}
\begin{bmatrix}
  t_{11} & t_{12} & 0 \\
  t_{21} & t_{22} & 0 \\
  t_{31} & t_{32} & 1
\end{bmatrix}
\]

- \([x \ y \ 1]\) are called homogenous coordinates
- Can translate, rotate, scale, or shear based on values \(t_{ij}\)
- Multiple transformations can be concatenated by multiplying them to form new matrix \(T'\)
Example: Translation

\[
\begin{bmatrix} x & y & 1 \end{bmatrix} = \begin{bmatrix} v & w & 1 \end{bmatrix} T = \begin{bmatrix} v & w & 1 \end{bmatrix} \begin{bmatrix} t_{11} & t_{12} & 0 \\ t_{21} & t_{22} & 0 \\ t_{31} & t_{32} & 1 \end{bmatrix}
\]

What does $T$ look like for translation?

\[
x = v + t_x \\
y = w + t_y
\]
# Affine Transformations

<table>
<thead>
<tr>
<th>Transformation</th>
<th>Affine Matrix $T$</th>
<th>Coordinate Equations</th>
<th>Example</th>
</tr>
</thead>
</table>
| Translation   | $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ t_x & t_y & 1 \end{bmatrix}$ | $x = v + t_x$  
$y = w + t_y$ | ![T] |
| Scaling       | $\begin{bmatrix} c_x & 0 & 0 \\ 0 & c_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$ | $x = c_x v$  
$y = c_y w$ | ![T] |
| Rotation      | $\begin{bmatrix} \cos \theta & \sin \theta & 0 \\ -\sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$ | $x = v \cos \theta - w \sin \theta$  
$y = v \sin \theta + w \cos \theta$ | ![T] |
## Affine Transformations (cont.)

<table>
<thead>
<tr>
<th>Transformation</th>
<th>Affine Matrix $T$</th>
<th>Coordinate Equations</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shear (vertical)</td>
<td>$\begin{bmatrix} 1 &amp; 0 &amp; 0 \ s_v &amp; 1 &amp; 0 \ 0 &amp; 0 &amp; 1 \end{bmatrix}$</td>
<td>$x = v + s_v w$  \ $y = w$</td>
<td><img src="image" alt="Shear Vertical" /></td>
</tr>
<tr>
<td>Shear (horizontal)</td>
<td>$\begin{bmatrix} 1 &amp; s_h &amp; 0 \ 0 &amp; 1 &amp; 0 \ 0 &amp; 0 &amp; 1 \end{bmatrix}$</td>
<td>$x = v$  \ $y = s_h v + w$</td>
<td><img src="image" alt="Shear Horizontal" /></td>
</tr>
</tbody>
</table>
Example of Affine Transformation

Image rotated 21 degrees

Nearest Neighbor  Bilinear  Bicubic

(from Gonzalez & Woods, 2008)
Recall from last time

**Range transformation:** \( g(x, y) = t(f(x, y)) \)

(What is an example?)

---

Noise filtering
Image processing for noise reduction

Common types of noise:

- **Salt and pepper noise**: contains random occurrences of black and white pixels
- **Impulse noise**: contains random occurrences of white pixels
- **Gaussian noise**: variations in intensity drawn from a Gaussian normal distribution
How do we reduce the effects of noise?

\[ F[x, y] \]

\[ G[x, y] \]
How do we reduce the effects of noise?

$F[x, y]$  

$G[x, y]$
How do we reduce the effects of noise?

Idea: Compute mean value for each pixel from neighbors
Mean filtering

\[ F[x, y] \]

\[ G[x, y] \]
Filtering as cross-correlation

If the averaging window is \((2k+1)\times(2k+1)\):

\[
G[i, j] = \frac{1}{(2k + 1)^2} \sum_{u=-k}^{k} \sum_{v=-k}^{k} F[i + u, j + v]
\]

In our example in previous slide, \(k = 1\) for a 3x3 averaging window
Filtering as cross-correlation

Can generalize this by allowing *different weights for different neighboring pixels*:

\[
G[i, j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u, v] F[i + u, j + v]
\]

This is called **cross-correlation**, denoted by:

\[
G = H \otimes F
\]

H is called the “filter,” “kernel,” or “mask.”

Note: During implementation, we avoid the negative filter indices by using \(H[u+k,v+k]\) instead of \(H[u,v]\)
Kernel for mean filtering

What is the kernel for a 3x3 mean filter?

$$F[x, y]$$

$$H[u, v]$$
Kernel for mean filtering

What is the kernel for a 3x3 mean filter?

\[ F[x, y] \]

\[ H[u, v] = \frac{1}{9} \]

\[
\begin{bmatrix}
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1 \\
\end{bmatrix}
\]

\[
\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 90 & 90 & 90 & 90 & 90 & 0 \\
0 & 0 & 0 & 90 & 90 & 90 & 90 & 90 & 0 \\
0 & 0 & 0 & 90 & 90 & 90 & 90 & 90 & 0 \\
0 & 0 & 0 & 90 & 90 & 90 & 90 & 90 & 0 \\
0 & 0 & 0 & 90 & 90 & 90 & 90 & 90 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 90 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{bmatrix}
\]
Example of mean filtering

Input image

Filtered Images

- Salt and pepper noise

Kernel size:
- 3 x 3
- 5 x 5
- 7 x 7
Gaussian Filtering

A Gaussian kernel gives less weight to pixels further from the center of the window.

\[ F[x, y] \]

Kernel approximates Gaussian function:

\[ h(u, v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2 + v^2}{2\sigma^2}} \]

What happens if you increase \( \sigma \)?
Mean versus Gaussian filtering

Input Image

Mean filtered

Gaussian filtered
Filtering an impulse

$F[x, y]$  

$H[u, v]$  

$G = H \otimes F$  

Output = ?
Filtering an impulse

Impulse signal

Filter Kernel

\[ F[x, y] \]

\[ G = H \otimes F \]

\[ G[x, y] \]

Output is equal to filter kernel flipped horizontally & vertically

\[ H[u, v] \]
What if we want to get an output that looks exactly like the filter kernel?
Flipping kernels

Impulse signal

\[
\begin{array}{cccccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]

\( F[x, y] \)

Filter Kernel

\[
\begin{array}{ccc}
a & b & c \\
d & e & f \\
g & h & i \\
\end{array}
\]

Flipped Kernel

\[
\begin{array}{ccc}
i & h & g \\
f & e & d \\
c & b & a \\
\end{array}
\]

\( G = H \otimes F \)

Output is equal to filter kernel!

\[
\begin{array}{cccccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & a & b & c & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & d & e & f & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & g & h & i & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]

\( G[x, y] \)
Convolution

A **convolution** is a cross-correlation where the *filter is flipped both horizontally and vertically* before being applied to the image:

\[
G[i, j] = \sum_{u=\text{-}k}^{k} \sum_{v=\text{-}k}^{k} H[u, v]F[i - u, j - v]
\]

Written as:  \( G = H \ast F \)

Compare with cross-correlation:

\[
G[i, j] = \sum_{u=\text{-}k}^{k} \sum_{v=\text{-}k}^{k} H[u, v]F[i + u, j + v]
\]

If \( H \) is a Gaussian or mean kernel, how does convolution differ from cross-correlation?
Why convolution?

- Convolution is associative (cross-corr. is not):
  \[ F * (G * I) = (F * G) * I \]

- Important for efficiency:
  To apply two filters \( F \) and \( G \) sequentially to incoming images \( I \), pre-compute \( (F * G) \) and perform only 1 convolution (with pre-computed filter)

- Convolution also allows effects of filtering to be analyzed using Fourier analysis (will touch on this later)
Cross-correlation and template matching

Cross-correlation is useful for template matching (locating a given pattern in an image)

Image

Template (pattern)

\[ H[u, v] \]

\[ G = H \otimes F \]

Highest value yields location of pattern in image
Nonlinear filters: Median filter

- A **Median Filter** replaces the value of a pixel by the median of intensity values of neighbors
  - Recall: m is the median of a set of values iff half the values in the set are \( \leq m \) and half are \( \geq m \).
  - Median filtering of image I: For each location \( (x,y) \), sort intensity values in its neighborhood, determine median intensity value, and assign that value to \( I(x,y) \)

- Is a median filter better than a mean filter?
- Is median filtering a convolution?
Comparison of filters (salt-and-pepper noise)

3x3

7x7
Comparison of filters (Gaussian noise)

<table>
<thead>
<tr>
<th>Mean</th>
<th>Gaussian</th>
<th>Median</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="3x3 Mean" /></td>
<td><img src="image2" alt="3x3 Gaussian" /></td>
<td><img src="image3" alt="3x3 Median" /></td>
</tr>
<tr>
<td><img src="image4" alt="7x7 Mean" /></td>
<td><img src="image5" alt="7x7 Gaussian" /></td>
<td><img src="image6" alt="7x7 Median" /></td>
</tr>
</tbody>
</table>
Next Time: Edge detection

Things to do:

- Read Chap. 5: Secs. 5.6 - 5.8, 5.11 and online article by Cipolla & Gee on edge detection
- Mailing list: cse455@cs.washington.edu
  - Did you receive the first message? Otherwise, sign up
- Prepare for C/C++ programming
- Visit Vision and Graphics Lab (Sieg 327)
  - Your ID card should open Sieg 327
  - Check to make sure ASAP

Have a good weekend!