Image processing
Today

- Consequences of image formation
- Some basic primitives needed for computer vision problems
  - Edge detection
  - Image resizing
- Convolution as a basic operation
- Image pyramids as a basic structure
Recap

- Geometry: \( \bar{x}_{img} \equiv K \begin{bmatrix} R & t \end{bmatrix} \bar{x}_w \)

- Color (Lambertian assumption): 
  \[
  I(\bar{x}_{img}) = \rho(\bar{x}_w) \int L(\bar{x}_w, \Omega) \cos \theta(\bar{x}_w, \Omega) d\Omega
  \]
Consequences of image formation

• Nearby objects appear larger
• Parallel lines and planes converge
• Information lost: distance from camera
• Pixel color depends on light intensity, light direction and surface normal and paint on object
• So objects in images
  • can appear in many different sizes and many positions
  • can have very different color
Consequence 1: nearby pixels are similar
Consequence 1: nearby pixels are similar

• Why?
  • Nearby pixels in pinhole camera lead to nearby rays
  • Nearby rays *mostly* fall on the same object
  • Objects have *mostly* smooth surfaces and *mostly* uniform color
  • Lighting is *mostly* uniform
Consequence 1: nearby pixels are similar

- Nearby pixels that are *not* similar tend to have different depth, surface normal, paint or lighting

- Idea: *Abrupt changes in color can delineate objects, be a clue to shape, or be distinctive marks*

Depth discontinuities

Changes in albedo

Normal discontinuities
Key primitive: edge detection
Consequence 2: Farther away objects appear smaller
Key primitive: Image resizing

• May need to match objects/patches across different scales.
Some primitives

• Edge detection: identifying where pixels change color
  • Cue to object boundary
  • Cue to shape
  • More resilient to lighting than pixel color

• Image resizing: downsizing or upscaling images
  • Allows searching over scales

• Basic operation: convolution
Convolution
Image denoising
What is an image?

• A grid (matrix) of intensity values: 1 color or 3 colors
Mean filtering: replace pixel by mean of neighborhood

\[
(0 + 0 + 0 + 10 + 40 + 0 + 10 + 0 + 0 + 0)/9 = 6.66
\]
Noise reduction using mean filtering
A more general version

Local image data

\[
S[f](m, n) = \sum_{i=-k}^{k} \sum_{j=-k}^{k} w(i, j) f(m + i, n + j)
\]
Convolution and cross-correlation

• Cross correlation

\[ S[f] = w \otimes f \]

\[ S[f](m, n) = \sum_{i=-k}^{k} \sum_{j=-k}^{k} w(i, j) f(m + i, n + j) \]

• Convolution

\[ S[f] = w \ast f \]

\[ S[f](m, n) = \sum_{i=-k}^{k} \sum_{j=-k}^{k} w(i, j) f(m - i, n - j) \]
Convolution

Adapted from F. Durand
Properties: Linearity

\[(w \otimes f)(m, n) = \sum_{i=-k}^{k} \sum_{j=-k}^{k} w(i, j) f(m + i, n + j)\]

\[f' = af + bg\]

\[w \otimes f' = a(w \otimes f) + b(w \otimes g)\]
Properties: Linearity

\[(w \otimes f)(m, n) = \sum_{i=-k}^{k} \sum_{j=-k}^{k} w(i, j)f(m + i, n + j)\]

\[w' = aw + bv\]

\[w' \otimes f = a(w \otimes f) + b(v \otimes f)\]
Properties: Shift invariance

\[(w \otimes f)(m, n) = \sum_{i=-k}^{k} \sum_{j=-k}^{k} w(i, j) f(m + i, n + j)\]

\[f'(m, n) = f(m - m_0, n - n_0)\]

\(f\)

\(f'\)
Shift invariance

\[(w \otimes f)(m, n) = \sum_{i=-k}^{k} \sum_{j=-k}^{k} w(i, j) f(m + i, n + j)\]

\[f'(m, n) = f(m - m_0, n - n_0)\]

\[(w \otimes f')(m, n) = \sum_{i=-k}^{k} \sum_{j=-k}^{k} w(i, j) f'(m + i, n + j)\]

\[= \sum_{i=-k}^{k} \sum_{j=-k}^{k} w(i, j) f(m + i - m_0, n + j - n_0)\]

\[= (w \otimes f)(m - m_0, n - n_0)\]
Shift invariance

\[
f'(m, n) = f(m - m_0, n - n_0)
\]

\[
(w \otimes f')(m, n) = (w \otimes f)(m - m_0, n - n_0)
\]

• Shift, then convolve = convolve, then shift

• Convolution does not depend on where the pixel is
Why is convolution important?

• Shift invariance is a crucial property
Why is convolution important?

• **We *like* linearity**
  • Linear functions behave predictably when input changes
  • Lots of theory just easier with linear functions
• **All linear shift-invariant systems can be expressed as a convolution**
• Basic primitive in computer vision
Image resizing
Why is resizing hard?

- E.g., consider reducing size by a factor of 2
- Simple solution: subsampling
- Example: subsampling by a factor of 2
Why is resizing hard?

• Dropping pixels causes problems
Aliasing in time
Aliasing in time
Why does aliasing happen?

- We "miss" things between samples
- High frequency signals might appear as low frequency signals
- Called “aliasing”
What about the general case?

- Every signal (doesn’t matter what it is)
  - Sum of sine/cosine waves
  - Fourier transform
Fourier transform

• Represent each signal as a linear combination of sines and cosines
• Equivalent to a change of basis
• Fourier transform = representation of signal in Fourier basis
Fourier transform for images

• Images are 2D arrays
• Fourier basis elements are indexed by 2 spatial frequencies
• \((i,j)\)th Fourier basis for \(N \times N\) image
  • Has period \(N/i\) along \(x\)
  • Has period \(N/j\) along \(y\)
• \(B_{k,l}(x, y) = e^{\frac{2\pi ikx}{N} + \frac{2\pi lly}{N}}\)
  \(= \cos \left(\frac{2\pi kx}{N} + \frac{2\pi lly}{N}\right) + i \sin \left(\frac{2\pi kx}{N} + \frac{2\pi lly}{N}\right)\)
Visualizing the Fourier basis for images

$B_{1,1}$

$B_{3,20}$

$B_{0,0}$

$B_{10,1}$
Visualizing the Fourier transform

• Given $N \times N$ image, there are $N \times N$ basis elements
• Fourier coefficients can be represented as an $N \times N$ image

High frequency in $X$
Aliasing
Aliasing

• Image = linear combination of high frequency and low frequency components
• Subsampling: high frequency components alias as low frequency
• First smooth the image to remove high frequency components
• How should we smooth?
  • Mean filtering?
Convolution and Fourier transforms

- Image: Spatial domain
- Fourier Transform: Frequency domain
  - Amplitudes are called spectrum
- For any transformations we do in spatial domain, there are corresponding transformations we can do in the frequency domain
- And vice-versa
Convolution and Fourier transforms

• **Convolution** in spatial domain = *Point-wise multiplication* in frequency domain
  
  • \( h = f \ast g \Rightarrow h(m, n) = \sum_{ij} f(i, j)g(m - i, n - j) \)
  
  • \( H = F \cdot G \Rightarrow H(k, l) = F(k, l) \cdot G(k, l) \)

• **Convolution** in frequency domain = *Point-wise multiplication* in spatial domain
Smoothing and Fourier transforms

• Mean filter = convolving with a “box” filter
Subsampling before and after smoothing

Before

After
Gaussian pre-filtering

- Solution: filter the image, *then* subsample
Gaussian pyramid

\{ F_0 \}

\begin{align*}
  & F_0 \\
  & \quad \downarrow \text{blur} \quad \downarrow \text{subsample} \quad \downarrow \text{blur} \quad \downarrow \text{subsample} \\
  & F_0 \ast H \\
  & \quad \downarrow \text{blur} \quad \downarrow \text{subsample} \\
  & F_1 \ast H \\
  & \quad \downarrow \text{blur} \\
  & \cdots
\end{align*}

\text{Gaussian pyramid}
Anti-aliasing circa 2019

Figure 2. Anti-aliasing common downsampling layers. (Top) Max-pooling, strided-convolution, and average-pooling can each be better antialiased (bottom) with our proposed architectural modification. An example on max-pooling is shown below.

Edge detection
Edges

- Edges are curves in the image, across which the brightness changes “a lot”
- Corners/Junctions
Closeup of edges

Source: D. Hoiem
Closeup of edges

Source: D. Hoiem
Closeup of edges

Source: D. Hoiem
Closeup of edges

Source: D. Hoiem
Characterizing edges

- An edge is a place of rapid change in the image intensity function

Source: L. Lazebnik
Intensity profile
Derivatives and convolution

- Differentiation is \textit{linear}

\[
\frac{\partial (af(x) + bg(x))}{\partial x} = a \frac{\partial f(x)}{\partial x} + b \frac{\partial g(x)}{\partial x}
\]

- Differentiation is \textit{shift-invariant}
  - Derivative of shifted signal is shifted derivative

- Hence, differentiation can be represented as convolution!
Image derivatives

- How can we differentiate a digital image \( F[x,y] \)?
  - Option 1: reconstruct a continuous image, \( f \), then compute the derivative
  - Option 2: take discrete derivative (finite difference)

\[
\frac{\partial f}{\partial x}[x, y] \approx F[x + 1, y] - F[x, y]
\]

How would you implement this as a linear filter?

\[
\frac{\partial f}{\partial x} \quad H_x
\]

\[
\frac{\partial f}{\partial y} \quad H_y
\]
Image gradient

- The gradient of an image: \( \nabla f = \left[ \frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right] \)

The gradient points in the direction of most rapid increase in intensity

\[
\nabla f = \left[ \frac{\partial f}{\partial x}, 0 \right]
\]

\[
\nabla f = \left[ 0, \frac{\partial f}{\partial y} \right]
\]

The edge strength is given by the gradient magnitude:

\[
\| \nabla f \| = \sqrt{\left( \frac{\partial f}{\partial x} \right)^2 + \left( \frac{\partial f}{\partial y} \right)^2}
\]

The gradient direction is given by:

\[
\theta = \tan^{-1} \left( \frac{\partial f}{\partial y} / \frac{\partial f}{\partial x} \right)
\]

- how does this relate to the direction of the edge?

Source: Steve Seitz
Image gradient
With a little Gaussian noise
Effects of noise

Noisy input image

$f(x)$

$rac{d}{dx} f(x)$

Where is the edge?

Source: S. Seitz
Solution: smooth first

To find edges, look for peaks in \( \frac{d}{dx}(f \ast h) \)

Source: S. Seitz
Associative property of convolution

- Differentiation is a convolution
- Convolution is associative:
  \[ \frac{d}{dx}(f \ast h) = f \ast \frac{d}{dx}h \]
- This saves us one operation:

\[ f \]

Source: S. Seitz
2D edge detection filters

Gaussian

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

derivative of Gaussian (x)

\[ \frac{\partial}{\partial x} h_\sigma(u, v) \]
Derivative of Gaussian filter

$x$-direction

$y$-direction