Plan for today

• 1. Basics in feature extraction: filtering
• 2. Invariant local features
• 3. Specific object recognition methods

Basics in feature extraction
Image Formation

Digital camera

A digital camera replaces film with a sensor array
- Each cell in the array is light-sensitive diode that converts photons to electrons
Digital images

- **Sample** the 2D space on a regular grid
- **Quantize** each sample (round to nearest integer)
- Image thus represented as a matrix of integer values.

**FIGURE 2.17** (a) Continuous image projected onto a sensor array. (b) Result of image sampling and quantization.

Adapted from S. Seitz
Digital color images

Color images, RGB color space

Kristen Grauman

R  G  B
Images in Matlab

- Images represented as a matrix
- Suppose we have a NxM RGB image called “im”
  - im(1, 1, 1) = top-left pixel value in R-channel
  - im(y, x, b) = y pixels down, x pixels to right in the b\text{th} channel
  - im(N, M, 3) = bottom-right pixel in B-channel
- imread(filename) returns a uint8 image (values 0 to 255)
  - Convert to double format (values 0 to 1) with im2double

Main idea: image filtering

- Compute a function of the local neighborhood at each pixel in the image
  - Function specified by a “filter” or mask saying how to combine values from neighbors.

- Uses of filtering:
  - Enhance an image (denoise, resize, etc)
  - Extract information (texture, edges, etc)
  - Detect patterns (template matching)
Motivation: noise reduction

- Even multiple images of the same static scene will not be identical.

Kristen Grauman

Motivation: noise reduction

- Even multiple images of the same static scene will not be identical.
- How could we reduce the noise, i.e., give an estimate of the true intensities?
- What if there’s only one image?
First attempt at a solution

- Let’s replace each pixel with an average of all the values in its neighborhood
- Assumptions:
  - Expect pixels to be like their neighbors
  - Expect noise processes to be independent from pixel to pixel

Moving average in 1D:

Source: S. Marschner
Weighted Moving Average

Can add weights to our moving average

Weights \[1, 1, 1, 1, 1 \] / 5

Source: S. Marschner

Weighted Moving Average

Non-uniform weights \[1, 4, 6, 4, 1 \] / 16

Source: S. Marschner
Moving Average In 2D

\[ F[x, y] \]

\[ G[x, y] \]

Source: S. Seitz
Moving Average In 2D

\[ F[x, y] \quad G[x, y] \]

Source: S. Seitz
Moving Average In 2D

\[ F[x, y] \quad \text{and} \quad G[x, y] \]

Source: S. Seitz
Correlation filtering

Say the averaging window size is 2k+1 x 2k+1:

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

Attribute uniform weight to each pixel  Loop over all pixels in neighborhood around image pixel F[i,j]

Now generalize to allow different weights depending on neighboring pixel’s relative position:

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

Non-uniform weights

Correlation filtering

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

This is called cross-correlation, denoted \( G = H \otimes F \)

Filtering an image: replace each pixel with a linear combination of its neighbors.

The filter “kernel” or “mask” \( H[u,v] \) is the prescription for the weights in the linear combination.
Averaging filter

• What values belong in the kernel \( H \) for the moving average example?

\[
F[x, y] \otimes H[u, v] \quad \frac{1}{9} \quad G[x, y]
\]

\[
G = H \otimes F
\]

Smoothing by averaging

What if the filter size was 5 x 5 instead of 3 x 3?
Gaussian filter

- What if we want nearest neighboring pixels to have the most influence on the output?

This kernel is an approximation of a 2d Gaussian function:

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

- Removes high-frequency components from the image ("low-pass filter").

Smoothing with a Gaussian

Source: S. Seitz
Gaussian filters

- What parameters matter here?
- Variance of Gaussian: determines extent of smoothing

\[
\sigma = 2 \text{ with } 30 \times 30 \text{ kernel} \quad \sigma = 5 \text{ with } 30 \times 30 \text{ kernel}
\]

Kristen Grauman

Smoothing with a Gaussian

Parameter \( \sigma \) is the “scale” / “width” / “spread” of the Gaussian kernel, and controls the amount of smoothing.

\[
\text{for } \sigma = 1:3:10 \\
\text{h = fspecial('gaussian', fsize, sigma);} \\
\text{out = imfilter(im, h);} \\
\text{imshow(out);} \\
\text{pause;} \\
\text{end}
\]

Kristen Grauman
Properties of smoothing filters

- **Smoothing**
  - Values positive
  - Sum to 1
  - Amount of smoothing proportional to mask size
  - Remove "high-frequency" components; "low-pass" filter

Predict the outputs using correlation filtering

* = ?

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

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

\[
\begin{array}{cc}
0 & 0 \\
0 & 2 & 0 \\
0 & 0 & 0 \\
\end{array}
\] \times \frac{1}{9}

\[
\begin{array}{ccc}
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1 \\
\end{array}
\] = ?
Practice with linear filters

Original

\[
\begin{bmatrix}
0 & 0 & 0 \\
0 & 0 & 1 \\
0 & 0 & 0 \\
\end{bmatrix}
\]

Source: D. Lowe

Practice with linear filters

Original

\[
\begin{bmatrix}
0 & 0 & 0 \\
0 & 0 & 1 \\
0 & 0 & 0 \\
\end{bmatrix}
\]

Shifted left by 1 pixel with correlation

Source: D. Lowe
Practice with linear filters

Original

\[
\begin{array}{ccc}
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1 \\
\end{array}
\]

Blur (with a box filter)

Original

\[
\begin{array}{ccc}
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1 \\
\end{array}
\]

Source: D. Lowe
Practice with linear filters

Original

\[
\begin{bmatrix}
0 & 0 & 0 \\
0 & 2 & 0 \\
0 & 0 & 0 \\
\end{bmatrix} - \frac{1}{9}
\begin{bmatrix}
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1 \\
\end{bmatrix}
\]

Source: D. Lowe

Practice with linear filters

Original

Sharpening filter:
accentuates differences
with local average

\[
\begin{bmatrix}
0 & 0 & 0 \\
0 & 2 & 0 \\
0 & 0 & 0 \\
\end{bmatrix} - \frac{1}{9}
\begin{bmatrix}
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1 \\
\end{bmatrix}
\]

Source: D. Lowe
Filtering examples: sharpening

Main idea: image filtering

• Compute a function of the local neighborhood at each pixel in the image
  – Function specified by a “filter” or mask saying how to combine values from neighbors.

• Uses of filtering:
  – Enhance an image (denoise, resize, etc)
  – Extract information (texture, edges, etc)
  – Detect patterns (template matching)
Why are gradients important?

Derivatives and edges

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

Source: L. Lazebnik
Derivatives with convolution

For 2D function, \( f(x, y) \), the partial derivative is:

\[
\frac{\partial f(x, y)}{\partial x} = \lim_{\varepsilon \to 0} \frac{f(x + \varepsilon, y) - f(x, y)}{\varepsilon}
\]

For discrete data, we can approximate using finite differences:

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

To implement above as convolution, what would be the associated filter?

Kristen Grauman

Partial derivatives of an image

Which shows changes with respect to \( x \)?

Kristen Grauman

(showing filters for correlation)
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 change in intensity

$$\nabla f = \left[ \frac{\partial f}{\partial x}, 0 \right] \quad \rightarrow \quad \nabla f = \left[ 0, \frac{\partial f}{\partial y} \right]$$

The gradient direction (orientation of edge normal) is given by:

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

Effects of noise

Consider a single row or column of the image

- Plotting intensity as a function of position gives a signal

Where is the edge?
Solution: smooth first

- Where is the edge? Look for peaks in $\frac{\partial}{\partial x} (h \ast f)$

Derivative theorem of convolution

$$\frac{\partial}{\partial x} (h \ast f) = (\frac{\partial}{\partial x} h) \ast f$$

Differentiation property of convolution.

Slide credit Steve Seitz
Derivative of Gaussian filters

\[(I \otimes g) \otimes h = I \otimes (g \otimes h)\]

\[
\begin{bmatrix}
0.0030 & 0.0133 & 0.0219 & 0.0133 & 0.0030 \\
0.0133 & 0.0596 & 0.0983 & 0.0596 & 0.0133 \\
0.0219 & 0.0983 & 0.1621 & 0.0983 & 0.0219 \\
0.0133 & 0.0596 & 0.0983 & 0.0596 & 0.0133 \\
0.0030 & 0.0133 & 0.0219 & 0.0133 & 0.0030
\end{bmatrix} \otimes \begin{bmatrix} 1 & -1 \end{bmatrix}
\]

Source: L. Lazebnik
Laplacian of Gaussian

Consider \( \frac{\partial^2}{\partial x^2} (h \ast f) \)

Where is the edge? Zero-crossings of bottom graph

2D edge detection filters

Gaussian derivative of Gaussian

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

\[
\frac{\partial}{\partial u} h_\sigma(u, v)
\]

\[
\nabla^2 h_\sigma(u, v)
\]

- \( \nabla^2 \) is the Laplacian operator:

\[
\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}
\]
Smoothing with a Gaussian

Recall: parameter $\sigma$ is the "scale" / "width" / "spread" of the Gaussian kernel, and controls the amount of smoothing.

Effect of $\sigma$ on derivatives

The apparent structures differ depending on Gaussian’s scale parameter.

Larger values: larger scale edges detected
Smaller values: finer features detected
Mask properties

- **Smoothing**
  - Values positive
  - Sum to 1 $\rightarrow$ constant regions same as input
  - Amount of smoothing proportional to mask size
  - Remove “high-frequency” components; “low-pass” filter

- **Derivatives**
  - ___________ signs used to get high response in regions of high contrast
  - Sum to ___ $\rightarrow$ no response in constant regions
  - High absolute value at points of high contrast

Main idea: image filtering

- Compute a function of the local neighborhood at each pixel in the image
  - Function specified by a “filter” or mask saying how to combine values from neighbors.

- **Uses of filtering:**
  - Enhance an image (denoise, resize, etc)
  - Extract information (texture, edges, etc)
  - **Detect patterns (template matching)**
Template matching

• Filters as templates:
  Note that filters look like the effects they are intended to find --- “matched filters”

• Use normalized cross-correlation score to find a given pattern (template) in the image.
• Normalization needed to control for relative brightnesses.

Template matching

Scene

Template (mask)

A toy example
Template matching

Detected template

Template

Template matching

Detected template

Correlation map
Where’s Waldo?

Template

Scene

Where’s Waldo?

Template

Detected template
Where’s Waldo?

Detected template

Correlation map

Template matching

Scene

Template

What if the template is not identical to some subimage in the scene?
Template matching

Match can be meaningful, if scale, orientation, and general appearance is right.

…but we can do better!...

Summary so far

- Compute a function of the local neighborhood at each pixel in the image
  - Function specified by a “filter” or mask saying how to combine values from neighbors.

- Uses of filtering:
  - Enhance an image (denoise, resize, etc)
  - Extract information (texture, edges, etc)
  - Detect patterns (template matching)