Lecture:
Edge Detection
What we will learn today

• Edge detection
• Image Gradients
• A simple edge detector
• Sobel edge detector
• Canny edge detector
• Hough Transform

Some background reading:
Forsyth and Ponce, Computer Vision, Chapter 8
What we will learn today

- Edge detection
- Image Gradients
- A simple edge detector
- Sobel edge detector
- Canny edge detector
- Hough Transform

Some background reading:
Forsyth and Ponce, Computer Vision, Chapter 8
Slide partially based on Stanford U. CS131
(A) Cave painting at Chauvet, France, about 30,000 B.C.;
(B) Aerial photograph of the picture of a monkey as part of the Nazca Lines geoglyphs, Peru, about 700 – 200 B.C.;
(C) Shen Zhou (1427-1509 A.D.): Poet on a mountain top, ink on paper, China;
(D) Line drawing by 7-year old I. Lleras (2010 A.D.).
A Experimental setup

Light bar stimulus projected on screen

Recording from visual cortex

Record

B Stimulus orientation

Stimulus presented

Hubel & Wiesel, 1960s

Slide partially based on Stanford U. CS131
We know edges are special from human (mammalian) vision studies.

Hubel & Wiesel, 1960s
Walther, Chai, Caddigan, Beck & Fei-Fei, *PNAS, 2011*
Edge detection

• **Goal:** Identify sudden changes (discontinuities) in an image
  – Intuitively, most semantic and shape information from the image can be encoded in the edges
  – More compact than pixels

• **Ideal:** artist’s line drawing (but artist is also using object-level knowledge)

Source: D. Lowe
Why do we care about edges?

- Extract information, recognize objects
- Recover geometry and viewpoint

Source: J. Hayes
Origins of edges

- surface normal discontinuity
- depth discontinuity
- surface color discontinuity
- illumination discontinuity

Source: D. Hoiem
Closeup of edges

Surface normal discontinuity

Source: D. Hoiem
Closeup of edges

Depth discontinuity

Source: D. Hoiem
Closeup of edges

Surface color discontinuity

Source: D. Hoiem
What we will learn today

- Edge detection
- Image Gradients
- A simple edge detector
- Sobel edge detector
- Canny edge detector
- Hough Transform
Derivatives in 1D

\[
\frac{df}{dx} = \lim_{\Delta x \to 0} \frac{f(x) - f(x - \Delta x)}{\Delta x} = f''(x) = f_x
\]
Derivatives in 1D - example

\[ y = x^2 + x^4 \]

\[ \frac{dy}{dx} = 2x + 4x^3 \]
Derivatives in 1D - example

\[ y = x^2 + x^4 \quad \frac{dy}{dx} = 2x + 4x^3 \]

\[ y = \sin x + e^{-x} \quad \frac{dy}{dx} = \cos x + (-1)e^{-x} \]
Discrete Derivative in 1D

\[
\frac{df}{dx} = \lim_{\Delta x \to 0} \frac{f(x) - f(x - \Delta x)}{\Delta x} = f'(x)
\]

\[
\frac{df}{dx} = \frac{f(x) - f(x - 1)}{1} = f''(x)
\]

\[
\frac{df}{dx} = f(x) - f(x - 1) = f'(x)
\]
Types of Discrete derivative in 1D

Backward  \[ \frac{df}{dx} = f(x) - f(x-1) = f'(x) \]

Forward  \[ \frac{df}{dx} = f(x) - f(x+1) = f'(x) \]

Central  \[ \frac{df}{dx} = f(x+1) - f(x-1) = f'(x) \]
1D discrete derivative filters

• Backward filter: \[ [0 \quad 1 \quad -1] \]
  \[ f(x) - f(x-1) = f'(x) \]

• Forward: \[ [-1 \quad 1 \quad 0] \]
  \[ f(x) - f(x+1) = f'(x) \]

• Central: \[ [1 \quad 0 \quad -1] \]
  \[ f(x+1) - f(x-1) = f'(x) \]
1D discrete derivative filters

- Backward filter: $[0 \quad 1 \quad -1]$

$$f(x) - f(x - 1) = f'(x)$$
1D discrete derivative filters

• Backward filter: \[ [0 \quad 1 \quad -1] \]

\[ f(x) - f(x - 1) = f'(x) \]

• Forward: \[ [-1 \quad 1 \quad 0] \]

\[ f(x) - f(x + 1) = f'(x) \]
1D discrete derivative example

\[ f(x) = 10 \ 15 \ 10 \ 10 \ 25 \ 20 \ 20 \ 20 \]

\[ f'(x) = 0 \ 5 \ -5 \ 0 \ 15 \ -5 \ 0 \ 0 \]
Discrete derivative in 2D

Given function $f(x, y)$
Discrete derivative in 2D

Given function

\[ f(x, y) \]

Gradient vector

\[
\nabla f(x, y) = \begin{bmatrix}
\frac{\partial f(x, y)}{\partial x} \\
\frac{\partial f(x, y)}{\partial y}
\end{bmatrix} = \begin{bmatrix}
x \\
y
\end{bmatrix}
\]
Discrete derivative in 2D

Given function

\[ f(x, y) \]

Gradient vector

\[ \nabla f(x, y) = \begin{bmatrix} \frac{\partial f(x, y)}{\partial x} \\ \frac{\partial f(x, y)}{\partial y} \end{bmatrix} = \begin{bmatrix} f_x \\ f_y \end{bmatrix} \]

Gradient magnitude

\[ |\nabla f(x, y)| = \sqrt{f_x^2 + f_y^2} \]

Gradient direction

\[ \theta = \tan^{-1} \left( \frac{\partial f}{\partial y} / \frac{\partial f}{\partial x} \right) \]
2D discrete derivative filters

What does this filter do?

\[
\frac{1}{3} \begin{bmatrix}
-1 & 0 & 1 \\
-1 & 0 & 1 \\
-1 & 0 & 1 \\
\end{bmatrix}
\]
2D discrete derivative filters

What about this filter?

\[
\frac{1}{3} \begin{bmatrix}
-1 & 0 & 1 \\
-1 & 0 & 1 \\
-1 & 0 & 1 \\
\end{bmatrix}
\]
2D discrete derivative - example

\[
I = \begin{bmatrix}
10 & 10 & 20 & 20 & 20 \\
10 & 10 & 20 & 20 & 20 \\
10 & 10 & 20 & 20 & 20 \\
10 & 10 & 20 & 20 & 20 \\
10 & 10 & 20 & 20 & 20 \\
\end{bmatrix}
\]
2D discrete derivative - example

What happens when we apply this filter?

\[ I = \begin{bmatrix}
10 & 10 & 20 & 20 & 20 \\
10 & 10 & 20 & 20 & 20 \\
10 & 10 & 20 & 20 & 20 \\
10 & 10 & 20 & 20 & 20 \\
10 & 10 & 20 & 20 & 20 \\
\end{bmatrix} \]

\[ \frac{1}{3} \begin{bmatrix}
1 & 1 & 1 \\
0 & 0 & 0 \\
-1 & -1 & -1 \\
\end{bmatrix} \]
2D discrete derivative - example

What happens when we apply this filter?

\[ I = \begin{bmatrix}
10 & 10 & 20 & 20 & 20 \\
10 & 10 & 20 & 20 & 20 \\
10 & 10 & 20 & 20 & 20 \\
10 & 10 & 20 & 20 & 20 \\
10 & 10 & 20 & 20 & 20 \\
\end{bmatrix} \]

\[ I_y = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
\end{bmatrix} \]
2D discrete derivative - example

Now let’s try the other filter!

\[ I = \begin{bmatrix}
10 & 10 & 20 & 20 & 20 \\
10 & 10 & 20 & 20 & 20 \\
10 & 10 & 20 & 20 & 20 \\
10 & 10 & 20 & 20 & 20 \\
10 & 10 & 20 & 20 & 20 \\
\end{bmatrix} \]

\[ \frac{1}{3} \begin{bmatrix}
-1 & 0 & 1 \\
-1 & 0 & 1 \\
-1 & 0 & 1 \\
\end{bmatrix} \]
2D discrete derivative - example

What happens when we apply this filter?

\[ I = \begin{bmatrix} 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \end{bmatrix} \]

\[ I_x = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 10 & 10 & 0 & 0 \\ 0 & 10 & 10 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \]
3x3 image gradient filters

\[
\frac{1}{3} \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}, \quad \frac{1}{3} \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}
\]
What we will learn today

• Edge detection
• Image Gradients
• A simple edge detector
• Sobel edge detector
• Canny edge detector
• Hough Transform
Characterizing edges

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

![Image of an edge in an image with corresponding intensity function and first derivative graphs.]
The gradient vector points in the direction of most rapid increase in intensity.

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?

The edge strength is given by the gradient magnitude

$$\| \nabla f \| = \sqrt{\left( \frac{\partial f}{\partial x} \right)^2 + \left( \frac{\partial f}{\partial y} \right)^2}$$
Finite differences: example

Original Image

Gradient magnitude

x-direction

y-direction

• Which one is the gradient in the x-direction? How about y-direction?
Intensity profile

Slide partially based on Stanford U. CS131
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?

Source: S. Seitz
Effects of noise
Effects of noise

• Finite difference filters respond strongly to noise
  – Image noise results in pixels that look very different from their neighbors
  – Generally, the larger the noise the stronger the response

• What is to be done?
Effects of noise

• Finite difference filters respond strongly to noise
  – Image noise results in pixels that look very different from their neighbors
  – Generally, the larger the noise the stronger the response

• What is to be done?
  – Smoothing the image should help, by forcing pixels different to their neighbors (=noise pixels?) to look more like neighbors

Source: D. Forsyth
Smoothing with different filters

• Mean smoothing
  \[
  \begin{bmatrix}
  1 \\
  1 \\
  1 \\
  \end{bmatrix}
  \begin{bmatrix}
  1 & 1 & 1 \\
  \end{bmatrix}
  \]

• Gaussian (smoothing * derivative)
  \[
  \begin{bmatrix}
  1 \\
  2 \\
  1 \\
  \end{bmatrix}
  \begin{bmatrix}
  1 & 2 & 1 \\
  \end{bmatrix}
  \]
Smoothing with different filters

Mean

Gaussian

Median

3x3

5x5

7x7

Slide credit: Steve Seitz

Slide partially based on Stanford U. CS131
Solution: smooth first

- To find edges, look for peaks in $\frac{d}{dx}(f * g)$.

Source: S. Seitz

Slide partially based on Stanford U. CS131
Derivative theorem of convolution

- This theorem gives us a very useful property:

\[
\frac{d}{dx} (f * g) = f * \frac{d}{dx} g
\]

- This saves us one operation:

Source: S. Seitz
Derivative of Gaussian filter

\[ \text{2D-gaussian} \ast [1 \ 0 \ -1] = \text{x - derivative} \]
Derivative of Gaussian filter

x-direction

y-direction
Derivative of Gaussian filter
Tradeoff between smoothing at different scales

- Smoothed derivative removes noise, but blurs edge. Also finds edges at different “scales”.

Source: D. Forsyth
Designing an edge detector

- Criteria for an “optimal” edge detector:
  - **Good detection**: the optimal detector must minimize the probability of false positives (detecting spurious edges caused by noise), as well as that of false negatives (missing real edges)
Designing an edge detector

- Criteria for an “optimal” edge detector:
  - **Good detection:** the optimal detector must minimize the probability of false positives (detecting spurious edges caused by noise), as well as that of false negatives (missing real edges)
  - **Good localization:** the edges detected must be as close as possible to the true edges
Designing an edge detector

• Criteria for an “optimal” edge detector:
  – **Good detection:** the optimal detector must minimize the probability of false positives (detecting spurious edges caused by noise), as well as that of false negatives (missing real edges)
  – **Good localization:** the edges detected must be as close as possible to the true edges
  – **Single response:** the detector must return one point only for each true edge point; that is, minimize the number of local maxima around the true edge
What we will learn today

- Edge detection
- Image Gradients
- A simple edge detector
- Sobel Edge detector
- Canny edge detector
- Hough transform
Sobel Operator

- uses two 3×3 kernels which are convolved with the original image to calculate approximations of the derivatives
- one for horizontal changes, and one for vertical

\[ G_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} \quad G_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \]
Sobel Operation

• Smoothing + differentiation

\[ G_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \begin{bmatrix} +1 & 0 & -1 \end{bmatrix} \]

Gaussian smoothing differentiaion
Sobel Operation

• Magnitude:

\[ G = \sqrt{G_x^2 + G_y^2} \]

• Angle or direction of the gradient:

\[ \Theta = \tan^{-1}\left(\frac{G_y}{G_x}\right) \]
Sobel Filter example
Sobel Filter Problems

- Poor Localization (Trigger response in multiple adjacent pixels)
- Thresholding value favors certain directions over others
  - Can miss oblique edges more than horizontal or vertical edges
  - False negatives
What we will learn today

• Edge detection
• Image Gradients
• A simple edge detector
• Sobel Edge detector
• Canny edge detector
• Hough Transform
Canny edge detector

• This is probably the most widely used edge detector in computer vision
• Theoretical model: step-edges corrupted by additive Gaussian noise
• Canny has shown that the first derivative of the Gaussian closely approximates the operator that optimizes the product of signal-to-noise ratio and localization

Canny edge detector

• Suppress Noise
• Compute gradient magnitude and direction
• Apply Non-Maximum Suppression
  – Assures minimal response
• Use hysteresis and connectivity analysis to detect edges
Example

• original image
Derivative of Gaussian filter

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

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

Slide partially based on Stanford U. CS131
Compute gradients (DoG)

X-Derivative of Gaussian   Y-Derivative of Gaussian   Gradient Magnitude

Source: J. Hayes
Get orientation at each pixel

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

X - Derivative of Gaussian
Y - Derivative of Gaussian
Gradient Magnitude
Canny edge detector

- Suppress Noise
- Compute gradient magnitude and direction
- Apply Non-Maximum Suppression
  - Assures minimal response
Non-maximum suppression

- Edge occurs where gradient reaches a maxima
- Suppress non-maxima gradient even if it passes threshold
- Only eight angle directions possible
  - Suppress all pixels in each direction which are not maxima
  - Do this in each marked pixel neighborhood
Remove spurious gradients

$|\nabla G|(x, y)$ is the gradient at pixel $(x, y)$

$$M(x, y) = \begin{cases} 
|\nabla G|(x, y) & \text{if } |\nabla G|(x, y) > |\nabla G|(x', y') \\
& \text{and } |\nabla G|(x, y) > |\nabla G|(x'', y'') \\
0 & \text{otherwise}
\end{cases}$$

$x'$ and $x''$ are the neighbors of $x$ along normal direction to an edge.
Non-maximum suppression

• Edge occurs where gradient reaches a maxima
• Suppress non-maxima gradient even if it passes threshold
• Only eight angle directions possible
  – Suppress all pixels in each direction which are not maxima
  – Do this in each marked pixel neighborhood
Non-maximum suppression

At q, we have a maximum if the value is larger than those at both p and at r. Interpolate to get these values.

Source: D. Forsyth
Non-max Suppression

Before

After
Canny edge detector

- Suppress Noise
- Compute gradient magnitude and direction
- Apply Non-Maximum Suppression
  - Assures minimal response
- Use hysteresis and connectivity analysis to detect edges
Hysteresis thresholding

- Define two thresholds: Low and High
  - If less than Low, not an edge
  - If greater than High, strong edge
  - If between Low and High, weak edge
Hysteresis thresholding

If the gradient at a pixel is
• above High, declare it as a ‘strong edge pixel’
• below Low, declare it as a “non-edge-pixel”
• between Low and High
  – Consider its neighbors iteratively then declare it an “edge pixel” if it is connected to an ‘strong edge pixel’ directly or via pixels between Low and High
Hysteresis thresholding

strong edge pixel

weak but connected edge pixels

strong edge pixel
Final Canny Edges
Canny edge detector

1. Filter image with x, y derivatives of Gaussian
2. Find magnitude and orientation of gradient
3. Non-maximum suppression:
   – Thin multi-pixel wide “ridges” down to single pixel width
4. Thresholding and linking (hysteresis):
   – Define two thresholds: low and high
   – Use the high threshold to start edge curves and the low threshold to continue them
Effect of $\sigma$ (Gaussian kernel spread/size)

The choice of $\sigma$ depends on desired behavior

- large $\sigma$ detects large scale edges
- small $\sigma$ detects fine features

Source: S. Seitz
Gradients (e.g. Canny)
Color
Texture
Combined
Human
45 years of boundary detection

Source: Arbelaez, Maire, Fowlkes, and Malik. TPAMI 2011 (pdf)