Edge Detection and Corner Detection

Introduction to Computer Vision

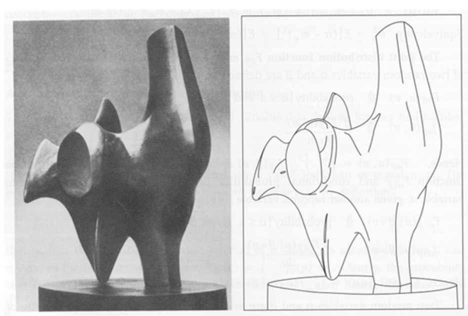
CSE 152

Lecture 7

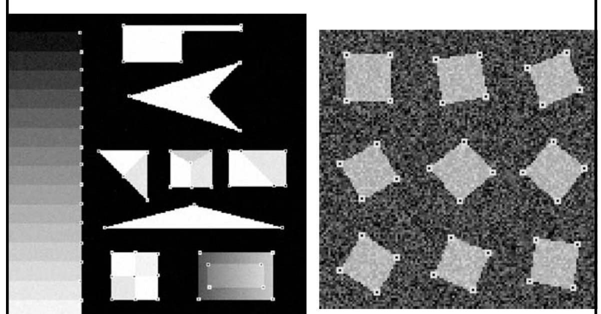
Announcements

- Homework 2 is due Apr 26, 11:59 PM
- Reading:
  – Chapter 5: Local Image Features

Edges



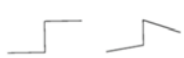
Corners



Edges

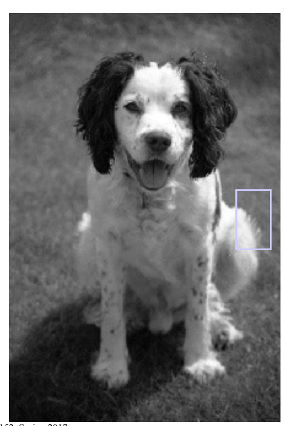
What is an edge?

A discontinuity in image intensity.

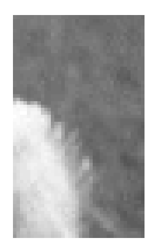


Physical causes of edges

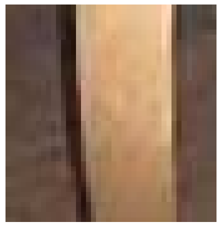
1. Object boundaries
2. Surface normal discontinuities
3. Reflectance (albedo) discontinuities
4. Lighting discontinuities (shadow boundaries)



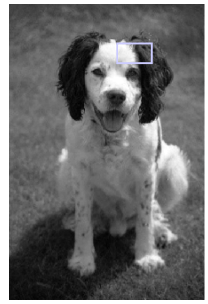
Object Boundaries

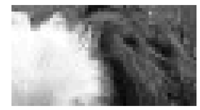


Surface normal discontinuities

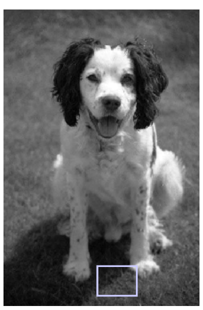


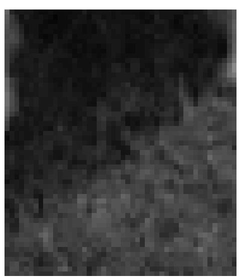
Boundaries of materials properties



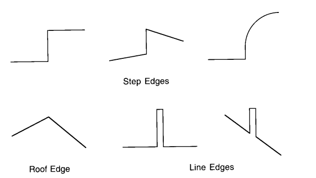


Boundaries of lighting





Profiles of image intensity edges

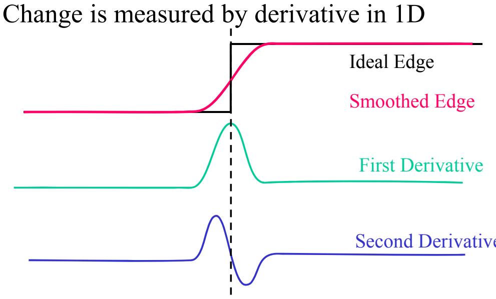


Noisy Step Edge

- Derivative is high everywhere.
- Must smooth before taking gradient.

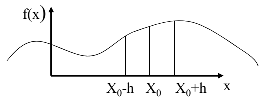
Edge is Where Change Occurs: 1-D

- Change is measured by derivative in 1D



- Biggest change, derivative has maximum magnitude
- Or 2nd derivative is zero.

Numerical Derivatives



Take Taylor series expansion of f(x) about x_0

$f(x) = f(x_0)+f'(x_0)(x-x_0) + ½ f''(x_0)(x-x_0)^2 + \cdots$

Consider samples taken at increments of h and first two terms of the expansion, we have

$f(x_0+h) = f(x_0)+f'(x_0)h+ ½ f''(x_0)h^2$

$f(x_0-h) = f(x_0)-f'(x_0)h+ ½ f''(x_0)h^2$

Subtracting and adding $f(x_0+h)$ and $f(x_0-h)$ respectively yields

$$f'(x_0) = \frac{f(x_0+h) - f(x_0-h)}{2h}$$

$$f''(x_0) = \frac{f(x_0+h) - 2f(x_0) + f(x_0-h)}{h^2}$$

Convolve with
First Derivative: [-1/2h 0 1/2h]
Second Derivative: [1/h² -2/h² 1/h²]

Numerical Derivatives

Convolution kernel
First Derivative: [-1/2h 0 1/2h]
Second Derivative: [1/h² -2/h² 1/h²]

- With images, units of h is pixels, so h=1
  - First derivative: [-1/2 0 1/2]
  - Second derivative: [1 -2 1]
- When computing derivatives in the x and y directions, use these convolution kernels:

$$\frac{d}{dx} = [-1/2 \quad 0 \quad 1/2] \qquad \frac{d}{dy} = \begin{bmatrix} -1/2 \\ 0 \\ 1/2 \end{bmatrix}$$

Implementing 1-D Edge Detection

1. Filter out noise: convolve with Gaussian
2. Take a derivative: convolve with [-1/2 0 1/2]
   – We can combine 1 and 2.
3. Find the peak: Two issues:
   – Should be a local maximum.
   – Should be sufficiently high.

Canny Edge Detector

1. Smooth image by filtering with a Gaussian
2. Compute gradient at each point in the image.
3. At each point in the image, compute the direction of the gradient and the magnitude of the gradient.
4. Perform non-maximal suppression to identify candidate edgels.
5. Trace edge chains using hysteresis tresholding.

2D Edge Detection: Canny

1. Filter out noise
   – Use a 2D Gaussian Filter.
2. Take a derivative
   – Compute the magnitude of the gradient:

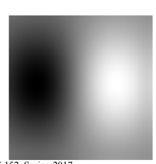
$$J = I \otimes G$$

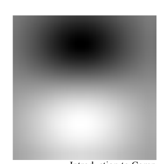
$$\nabla J = (J_x, J_y) = \left(\frac{\partial J}{\partial x}, \frac{\partial J}{\partial y}\right) \text{ is the Gradient}$$

$$\|\nabla J\| = \sqrt{J_x^2 + J_y^2}$$

Smoothing and Differentiation

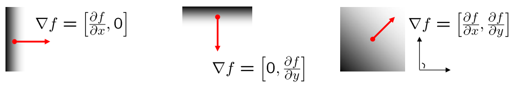
- Need two derivatives, in x and y direction.
- Filter with Gaussian and then compute Gradient, OR
- Use a derivative of Gaussian filter
  - because differentiation is convolution, and convolution is associative





Gradient

- Given a function f(x,y) -- e.g., intensity is f
- Gradient equation: $\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$
- Represents direction of most rapid change in intensity

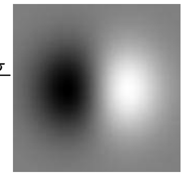


- Gradient direction: $\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$
- The *edge strength* is given by the gradient magnitude

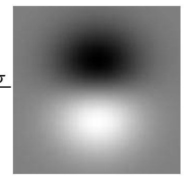
$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

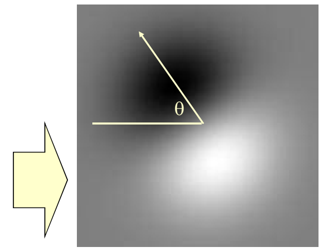
Directional Derivatives

$\frac{\partial G_\sigma}{\partial x}$



$\frac{\partial G_\sigma}{\partial y}$

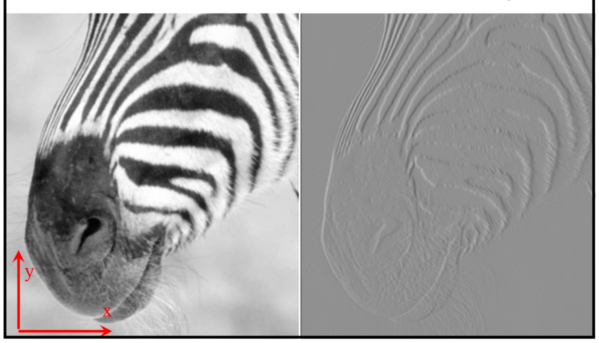


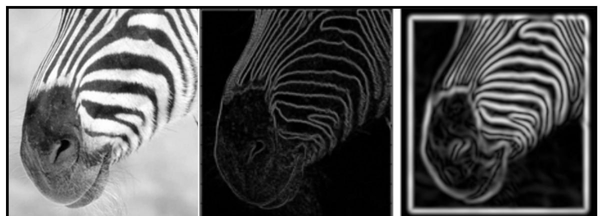


$$\cos\theta \frac{\partial G_\sigma}{\partial x} + \sin\theta \frac{\partial G_\sigma}{\partial y}$$

Finding derivatives

Is this dI/dx or dI/dy?





σ = 1 σ = 2

There are three major issues:

1. The gradient magnitude at different scales is different; which scale should we choose?
2. The gradient magnitude is large along a thick trail; how do we identify the significant points?
3. How do we link the relevant points up into curves?

There is ALWAYS a tradeoff between smoothing and good edge localization!

Image with Edge (No Noise)

Edge Location

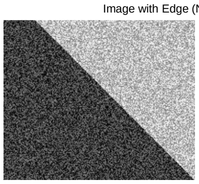
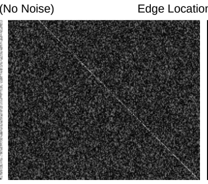
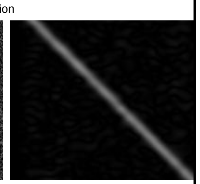


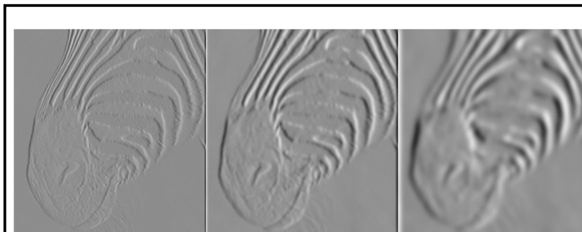
Image + Noise



Derivatives detect edge *and* noise

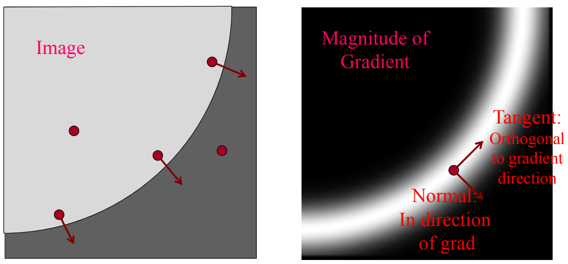


Smoothed derivative removes noise, but blurs edge



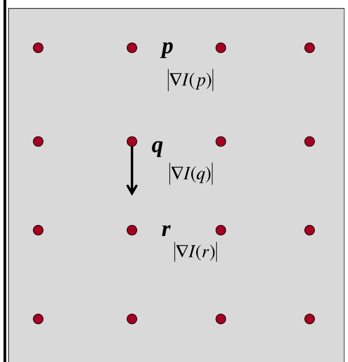
1 pixel 3 pixels 7 pixels

The scale of the smoothing filter affects derivative estimates



- We wish to mark points along a curve where the magnitude is biggest.
- We can do this by looking for a maximum along a slice orthogonal to the curve
- (non-maximum suppression).
- These points should form a curve.

Non-maximum suppression

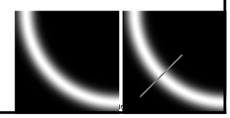


Loop over every point **q** in the image, decide whether **q** is a candidate edge point

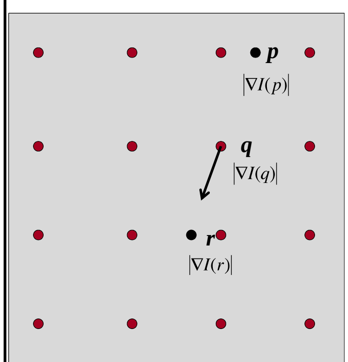
Using gradient direction at **q**, find two points **p** and **r** on adjacent rows (or columns).

If $|\nabla I(q)| > |\nabla I(p)|$ and $|\nabla I(q)| > |\nabla I(r)|$

then q is a candidate edgel



Non-maximum suppression

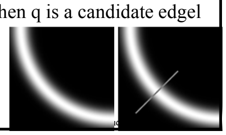


Loop over every point **q** in the image, decide whether **q** is a candidate edge point

Using gradient direction at **q**, find two points **p** and **r** on adjacent rows (or columns).
p & r are found by interpolation

If $|\nabla I(q)| > |\nabla I(p)|$ and $|\nabla I(q)| > |\nabla I(r)|$

then q is a candidate edgel



The Canny Edge Detector

original image (Lena)

(Example from Srinivasa Narasmihan)

The Canny Edge Detector

magnitude of the gradient

(Example from Srinivasa Narasmihan)

The Canny Edge Detector

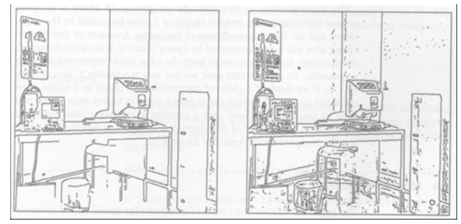
After non-maximum suppression

(Example from Srinivasa Narasmihan)

An Idea: Single Threshold

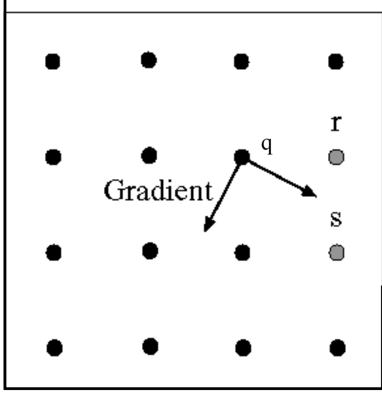
1. Smooth Image
2. Compute gradients & Magnitude
3. Non-maximal suppression
4. Compare to a threshold: T

An OK Idea: Single Threshold

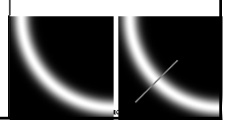
T=15  T=5

1. Smooth Image
2. Compute gradients & Magnitude
3. Non-maximal suppression
4. Compare to a threshold: T

A Better Idea: Linking + Two Tresholds

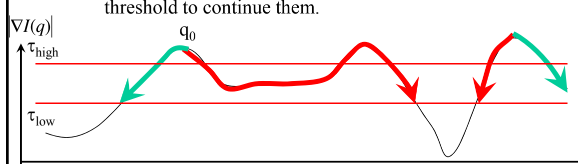


Linking: Assume the marked point q is an edge point. Then we construct the tangent to the edge curve (which is normal to the gradient at that point) and use this to predict the next points (either r or s).

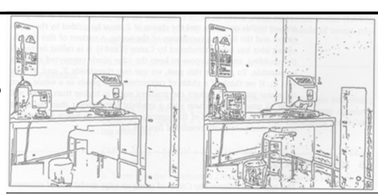


A Better Idea: HysteresisThresholding

- Define two thresholds τ_{low} and τ_{high}.
- Starting with output of nonmaximal supression, find a point q_0 where $|\nabla I(q_o)| > \tau_{high}$, and $|\nabla I(q_o)|$ is a local maximum.
- Start tracking an edge chain at pixel location q_0 in one of the two directions
- Stop when gradient magnitude < τ_{low}.
  - i.e., use a high threshold to start edge curves and a low threshold to continue them.



Single Threshold

T=15  T=5

Hysteresis T_h=15 T_l = 5

Hysteresis thresholding

fine scale, high threshold

coarse scale, high high threshold

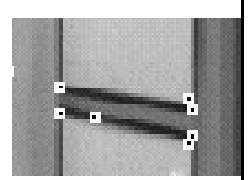
coarse scale, low high threshold

Why is Canny so dominant?

- Still widely used after 30 years.
  1. Theory is nice
  2. Details good (magnitude of gradient, non-max suppression).
  3. Hysteresis an important heuristic.
  4. Code was distributed.

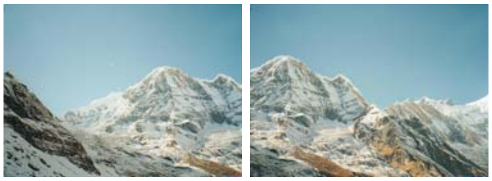
Corner Detection

Feature extraction: Corners



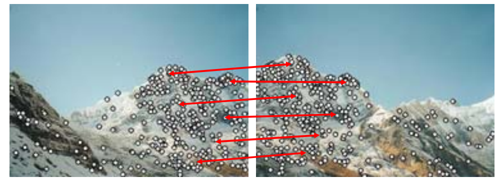
Why extract features?

- Motivation: panorama stitching
  - We have two images – how do we combine them?



Why extract features?

- Motivation: panorama stitching
  - We have two images – how do we combine them?



Step 1: extract features

Step 2: match features

Why extract features?

- Motivation: panorama stitching
  - We have two images – how do we combine them?

Step 1: extract features

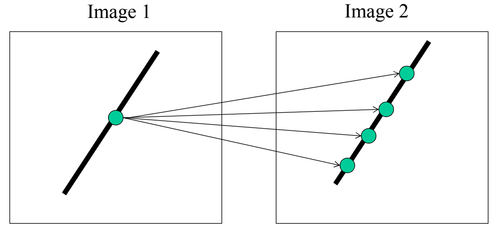
Step 2: match features

Step 3: align images

Corners contain more info than lines.

- A point on a line is hard to match.

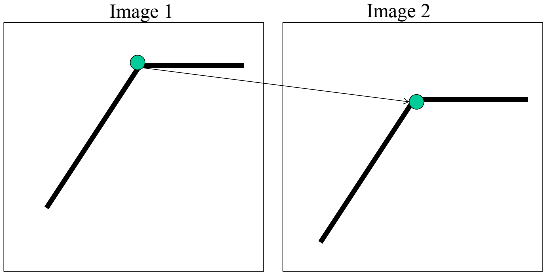
Image 1 Image 2



Corners contain more info than lines.

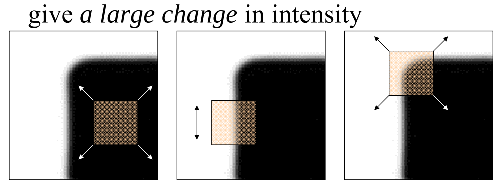
- A corner is easier to match

Image 1 Image 2



The Basic Idea

- We should easily recognize the point by looking through a small window
- Shifting a window in *any direction* should give *a large change* in intensity



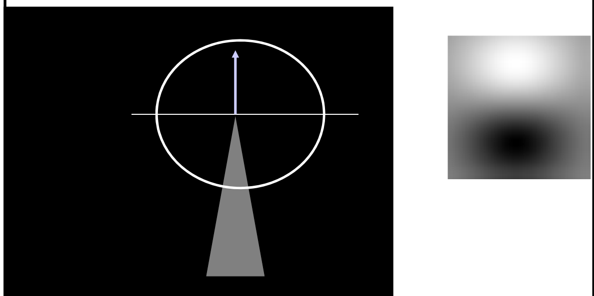
"flat" region: no change in all directions

"edge": no change along the edge direction

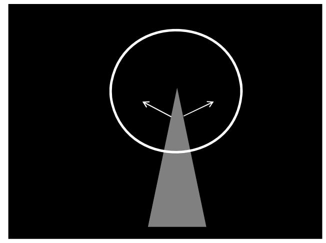
"corner": significant change in all directions

Source: A. Efros

Edge Detectors Tend to Fail at Corners



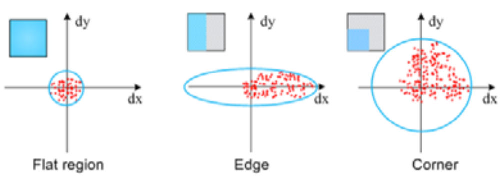
Finding Corners



Intuition:

- Right at corner, gradient is ill-defined.
- Near corner, gradient has two different values.

Distribution of gradients for different image patches



Derivative distribution of different regions

Finding Corners

For each image location (x,y), we create a matrix C(x,y):

Sum over a small region

Gradient with respect to x, times gradient with respect to y

$$C(x,y) = \begin{bmatrix} \sum\sum I_x^2 & \sum\sum I_x I_y \\ \sum\sum I_x I_y & \sum\sum I_y^2 \end{bmatrix}$$

Matrix is symmetric

WHY THIS?

Because C is a symmetric positive semidefinite matrix, it can be factored as:

$$C = R^{-1}\begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} R = R^T \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} R$$

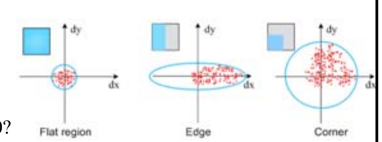
where R is a 2x2 rotation matrix and λ_1 and λ_2 are non-negative.

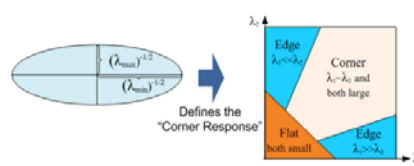
1. λ_1 and λ_2 are the Eigenvalues of C.
2. The columns of R are the Eigenvectors of C.
3. Eigenvalues can be found by solving the characteristic equation det(C-λ I) = 0 for λ.

Example: Assume R=Identity (axis aligned)

What is region like if:

1. $\lambda_1 = 0, \lambda_2 > 0$?
2. $\lambda_2 = 0, \lambda_1 > 0$?
3. $\lambda_1 = 0$ and $\lambda_2 = 0$?
4. $\lambda_1 >> 0$ and $\lambda_2 >> 0$?





So, to detect corners

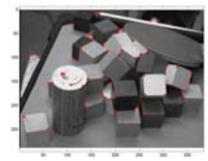
- Filter image with a Gaussian.
- Compute the gradient everywhere.
- Move window over image, and for each window location:
  1. Construct the matrix C over the window.
  2. Use linear algebra to find λ_1 and λ_2.
  3. If they are both big, we have a corner.
     1. Let $e(x,y) = \min(\lambda_1(x,y), \lambda_2(x,y))$
     2. (x,y) is a corner if it's local maximum of $e(x,y)$ and $e(x,y) > \tau$

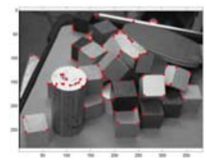
Parameters: Gaussian std. dev, window size, threshold

Corner Detection Sample Results

Threshold=25,000

Threshold=10,000





Threshold=5,000

Next Lecture

- Early vision: multiple images
  - Stereo
- Reading:
  - Chapter 7: Stereopsis