Edge Detection and Corner Detection

Computer Vision I

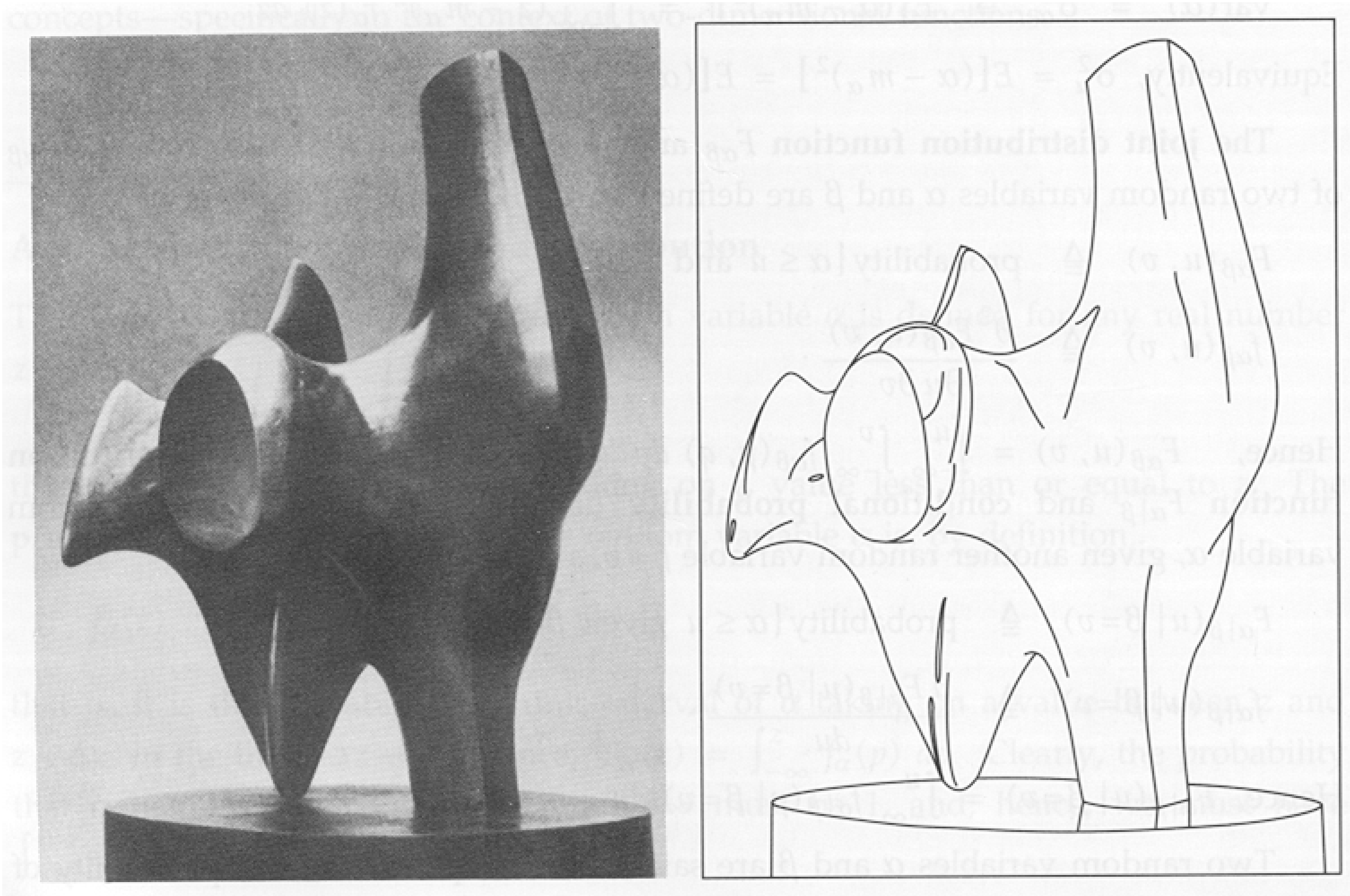
CSE 252A

Lecture 7

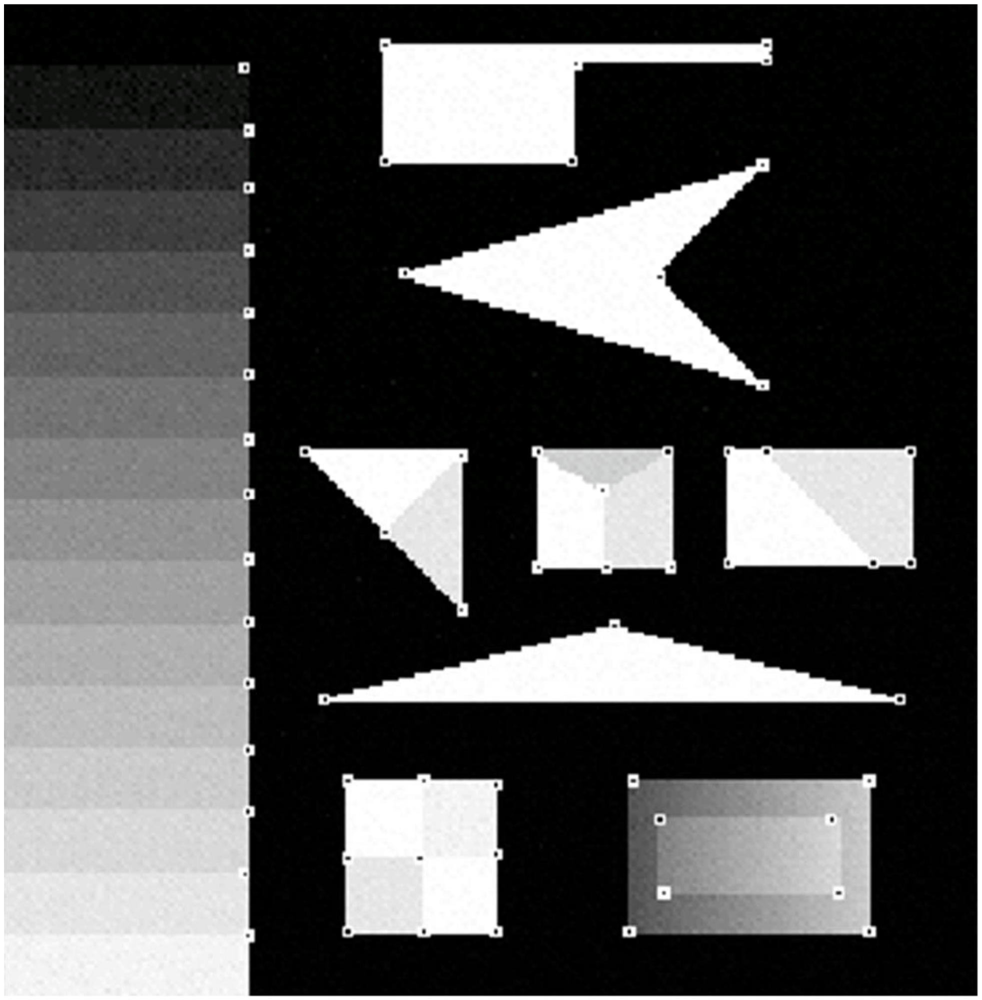
Announcements

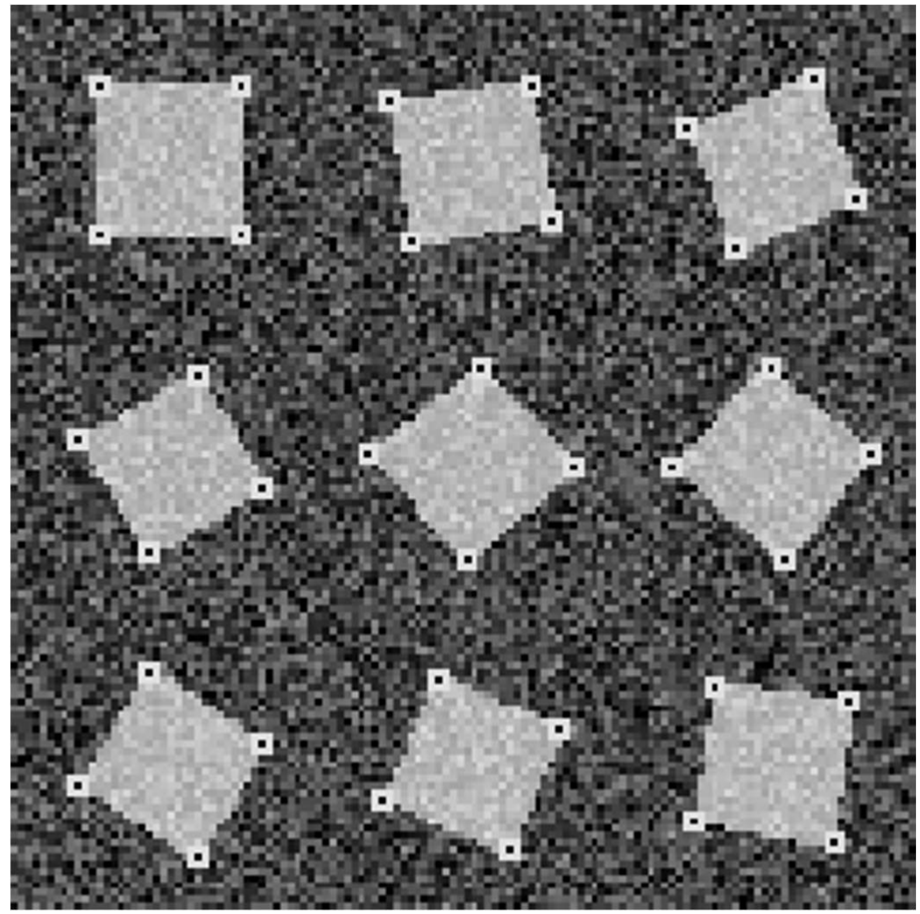
- Homework 2 is due Oct 22, 11:59 PM
- Homework 3 will be assigned on Oct 22
- Reading:
  - Chapter 5: Local Image Features

Edges



Corners





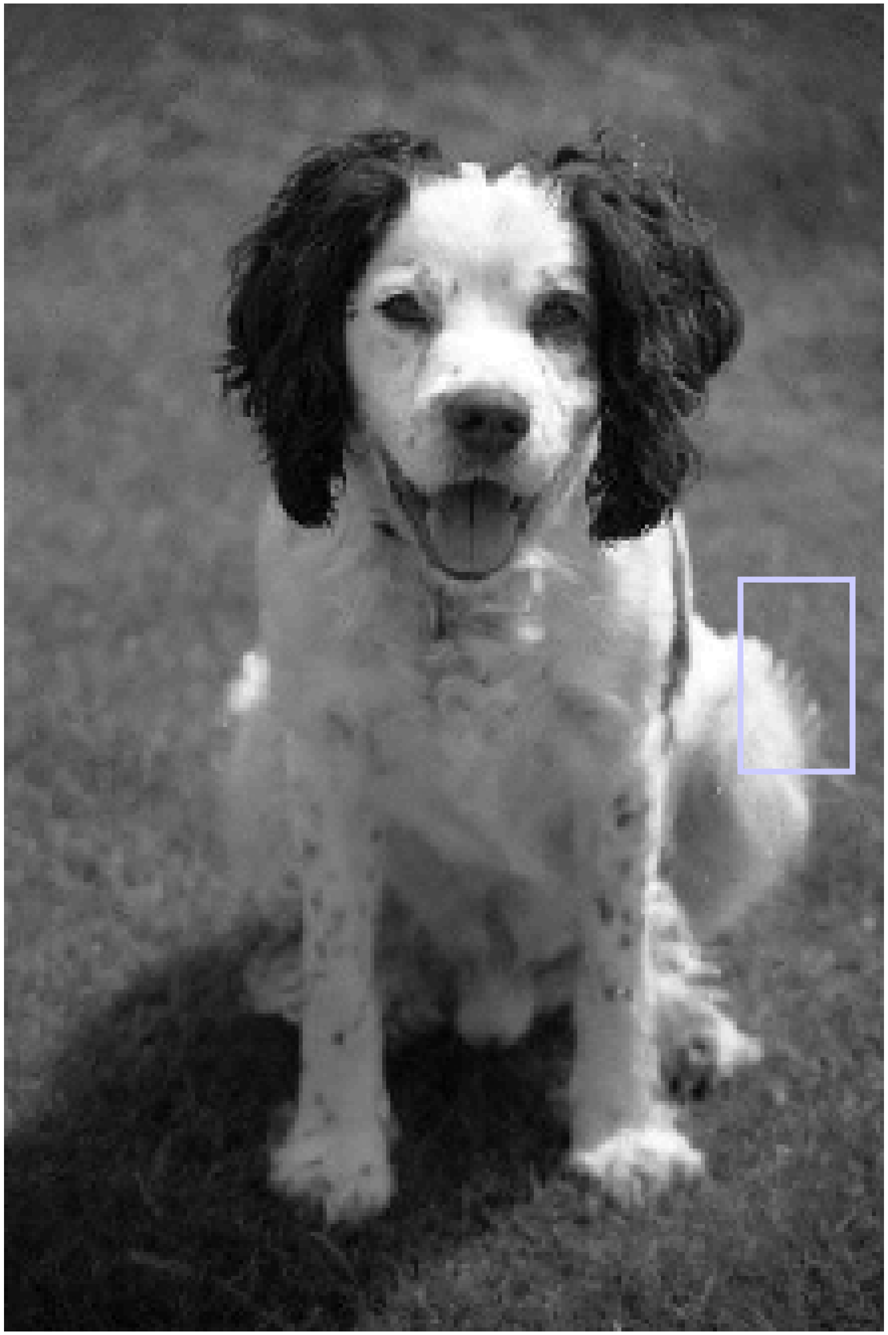
Edges

What is an edge?

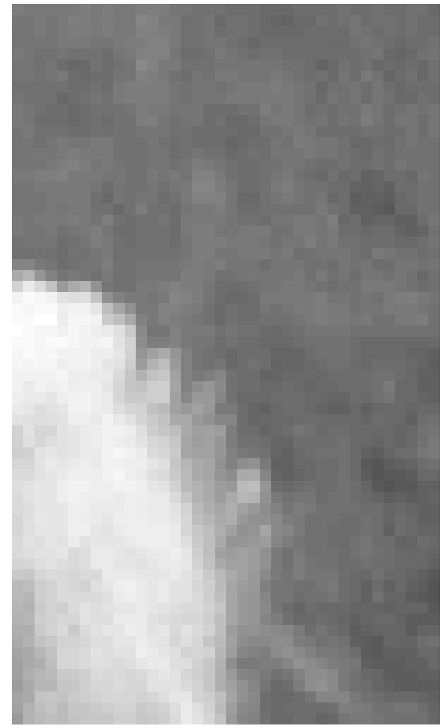
A discontinuity in image intensity.

Physical causes of edges

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

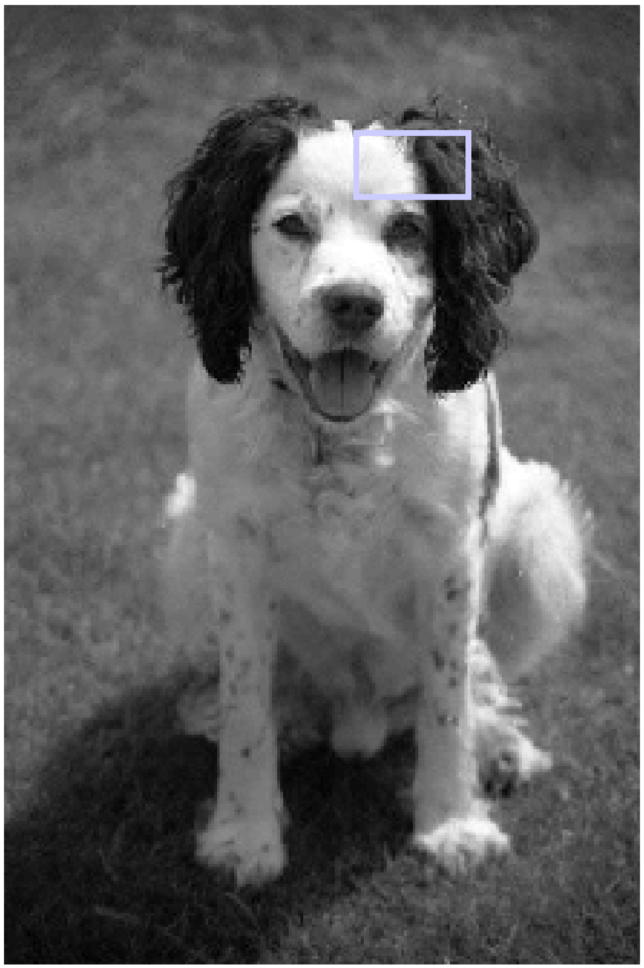


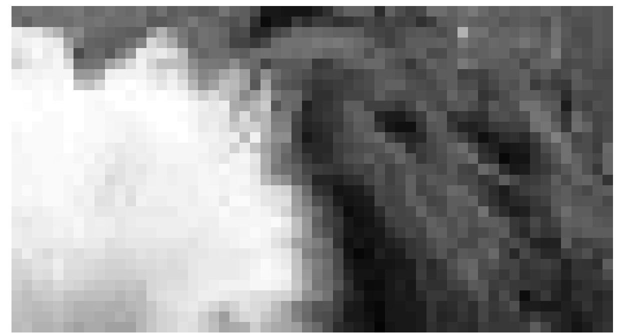
Object Boundaries



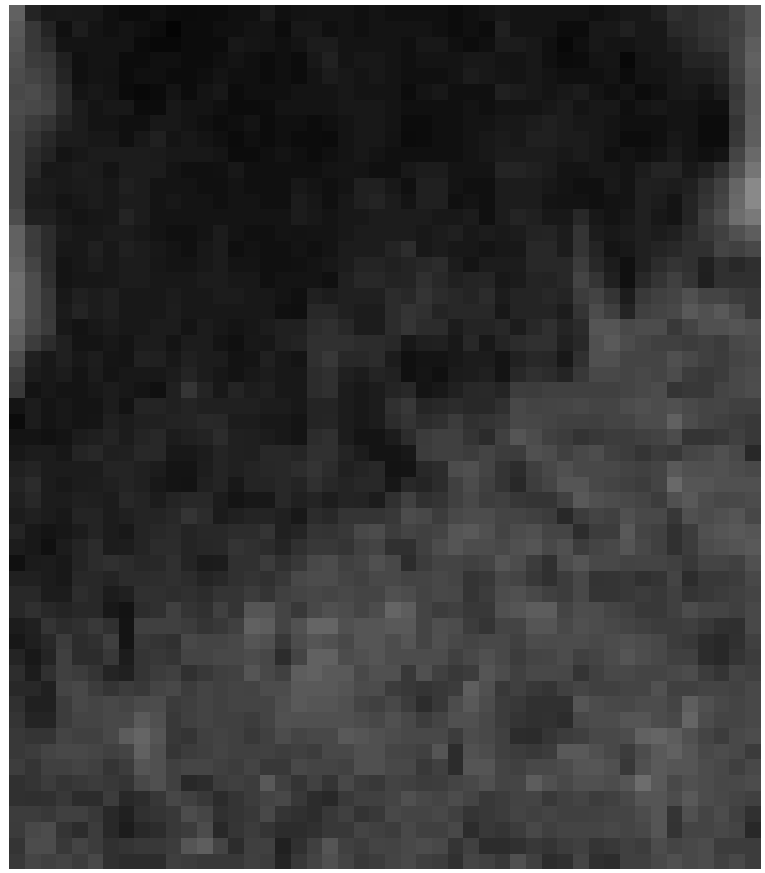
Surface normal discontinuities

Boundaries of materials properties

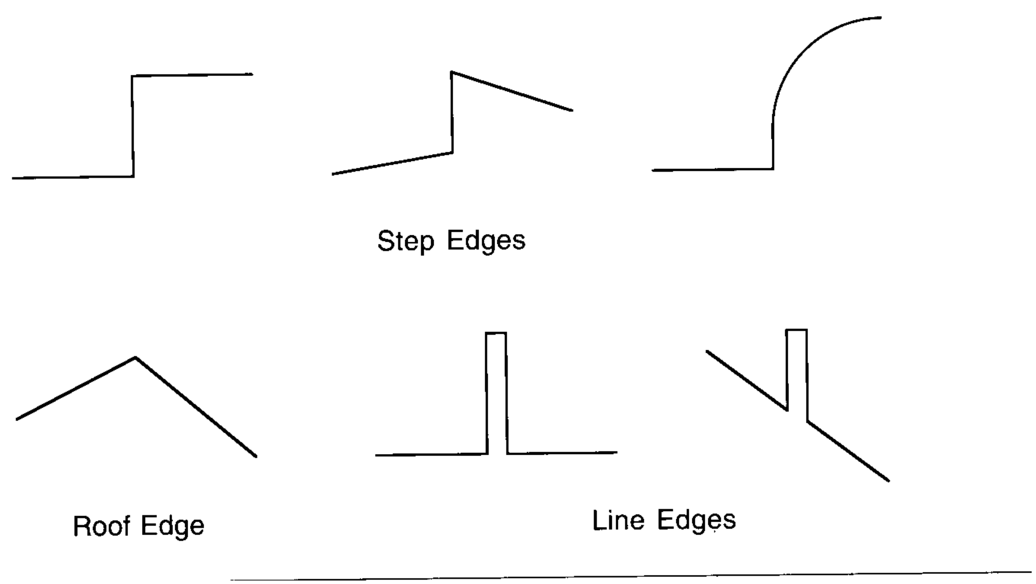




Boundaries of lighting

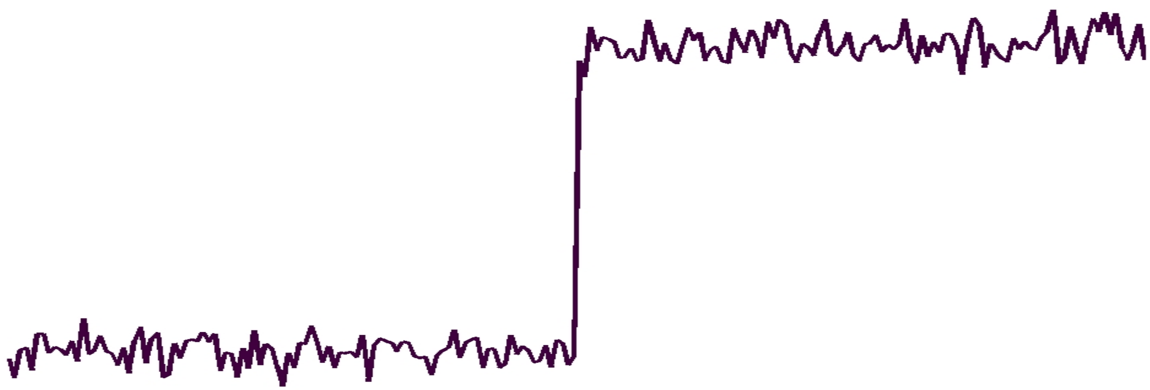


Profiles of image intensity edges



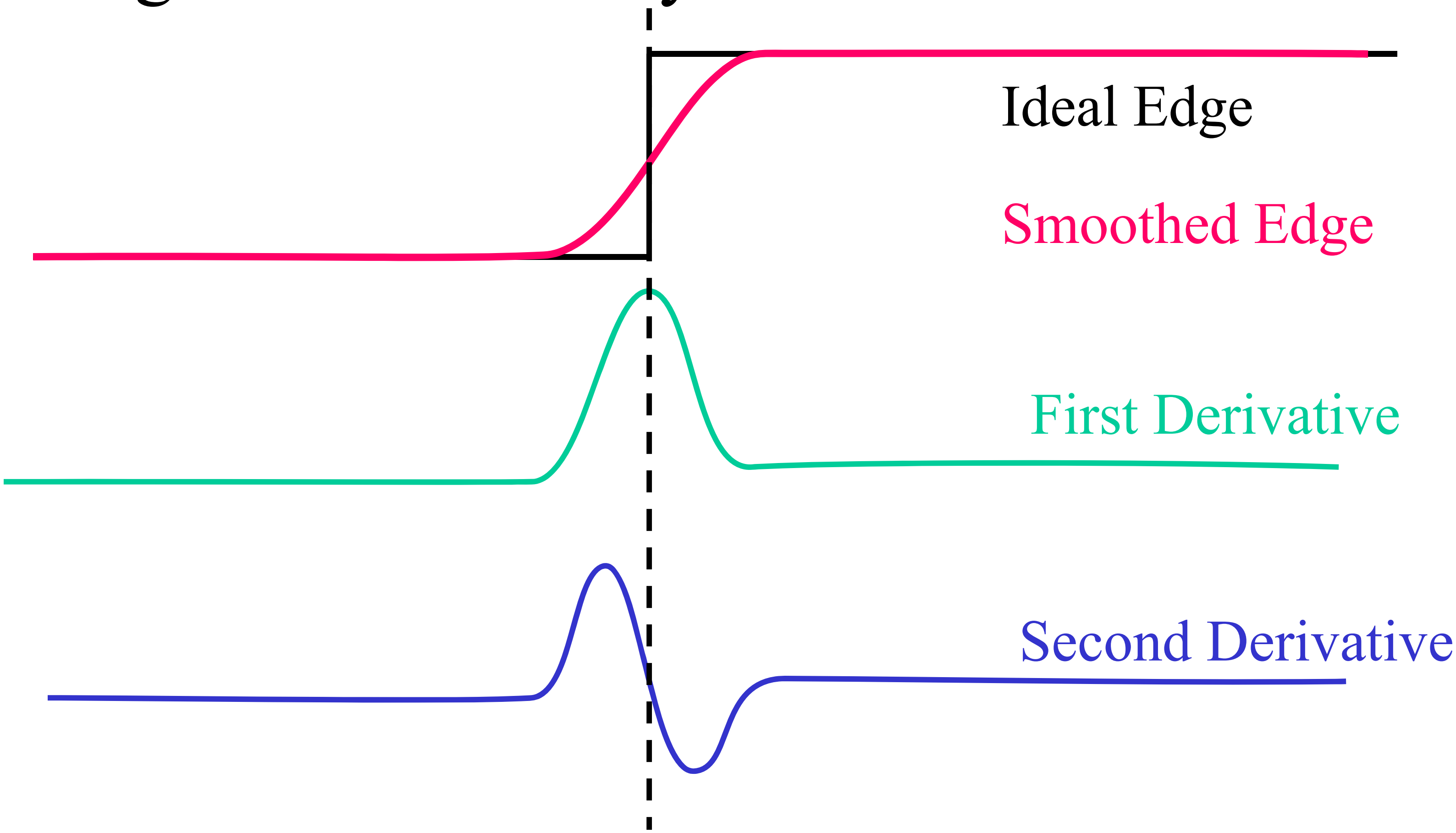
Noisy Step Edge

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



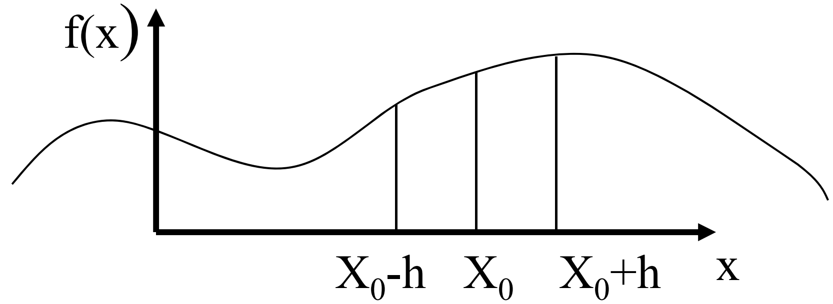
Edge is Where Change Occurs: 1-D

- 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

$$f(x) = f(x_0)+f'(x_0)(x-x_0) + \tfrac{1}{2} f''(x_0)(x-x_0)^2 + \cdots$$

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

$$f(x_0+h) = f(x_0)+f'(x_0)h+ \tfrac{1}{2} f''(x_0)h^2$$

$$f(x_0-h) = f(x_0)-f'(x_0)h+ \tfrac{1}{2} f''(x_0)h^2$$

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: [$1/h^2$ $-2/h^2$ $1/h^2$]

Numerical Derivatives

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

- 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:

$$\frac{d}{dx} = [-1/2 \quad 0 \quad 1/2]$$

$$\frac{d}{dy} = \begin{bmatrix} -1/2 \\ 0 \\ 1/2 \end{bmatrix}$$

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.

2D Edge Detection

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

$$J = I \otimes G$$

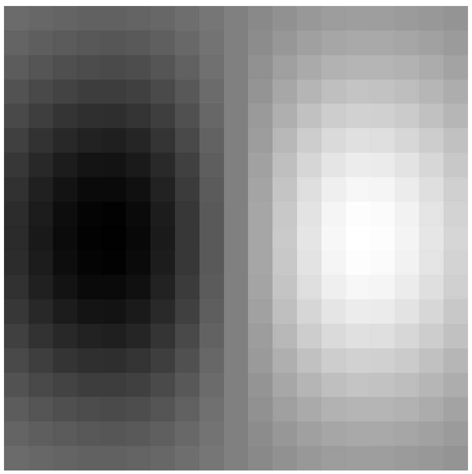
$$\nabla J = (J_x, J_y) = \left(\frac{\partial J}{\partial x}, \frac{\partial J}{\partial y} \right)$$ is the gradient

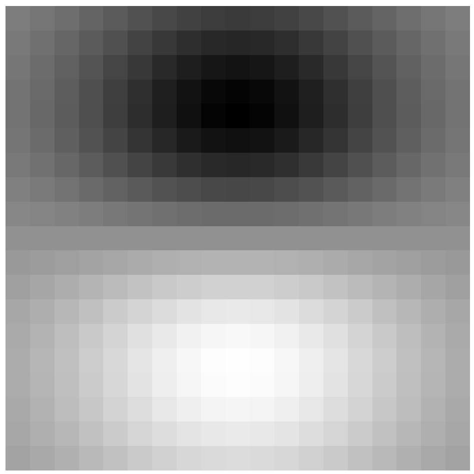
$$\left\| \nabla J \right\| = \sqrt{J_x^2 + J_y^2}$$ is the magnitude of the gradient

$$\tan^{-1}(J_y, J_y)$$ the direction of the gradient

Smoothing and Differentiation

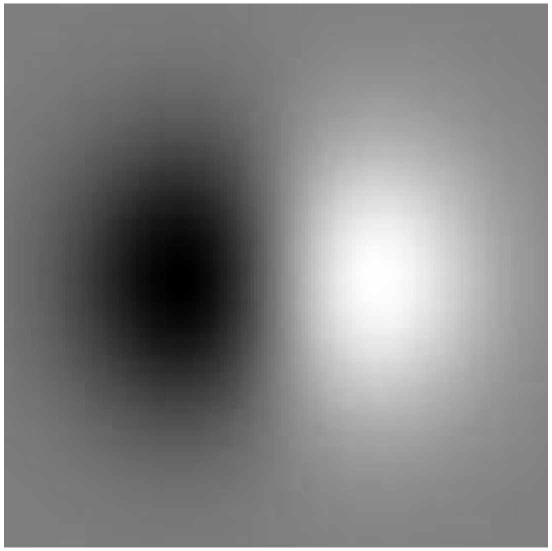
- Need two derivatives, in x and y direction.
- Filter with Gaussian and then compute Gradient, OR
- Use a derivative of Gaussian filter
    - because differentiation is convolution, and convolution is associative



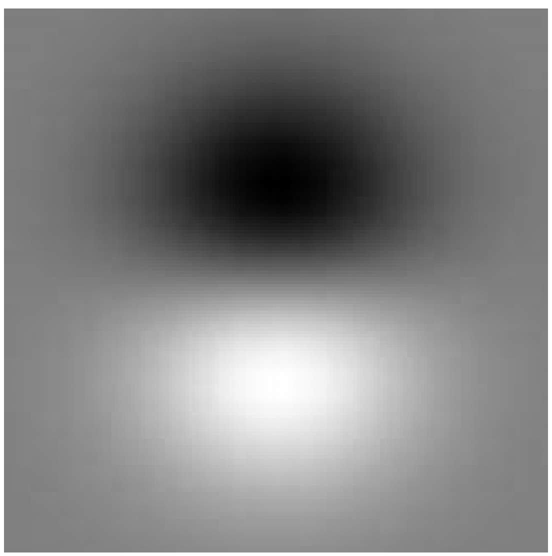


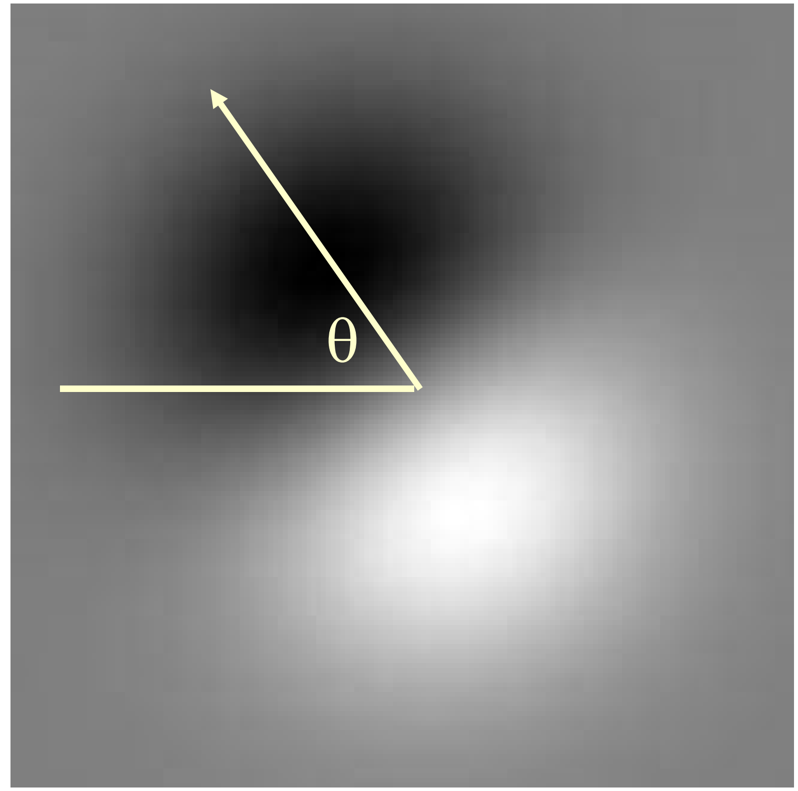
Directional Derivatives

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



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





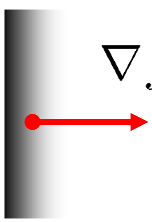
$$\cos\theta \frac{\partial G_\sigma}{\partial x} + \sin\theta \frac{\partial G_\sigma}{\partial y}$$

Gradient

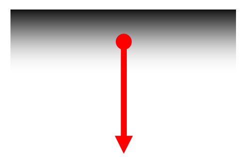
- Given a function f(x,y) -- e.g., intensity is f

- Gradient equation: $$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$

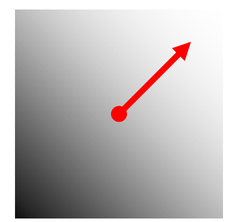
- Represents direction of most rapid change in intensity



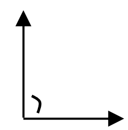
$$\nabla f = \left[\frac{\partial f}{\partial x}, 0\right]$$



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



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



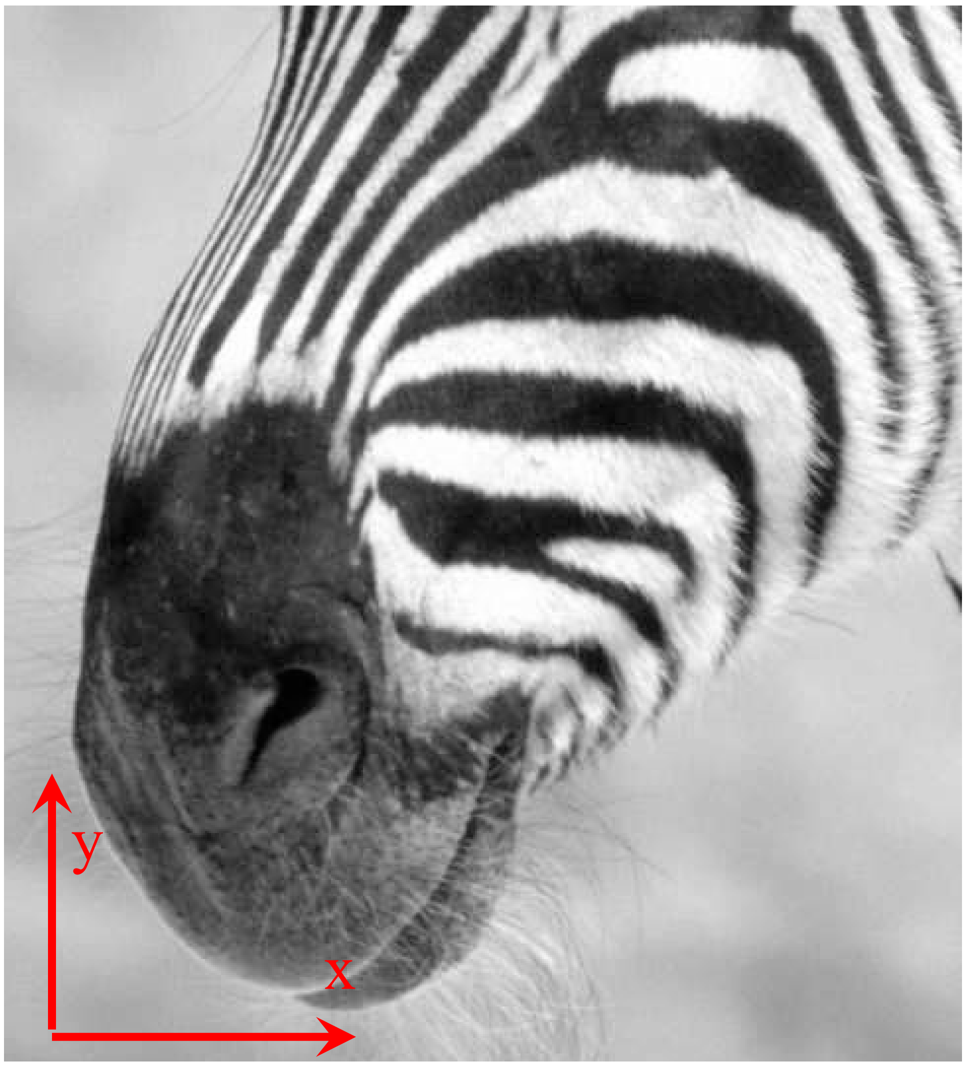
- Gradient direction: $$\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$$

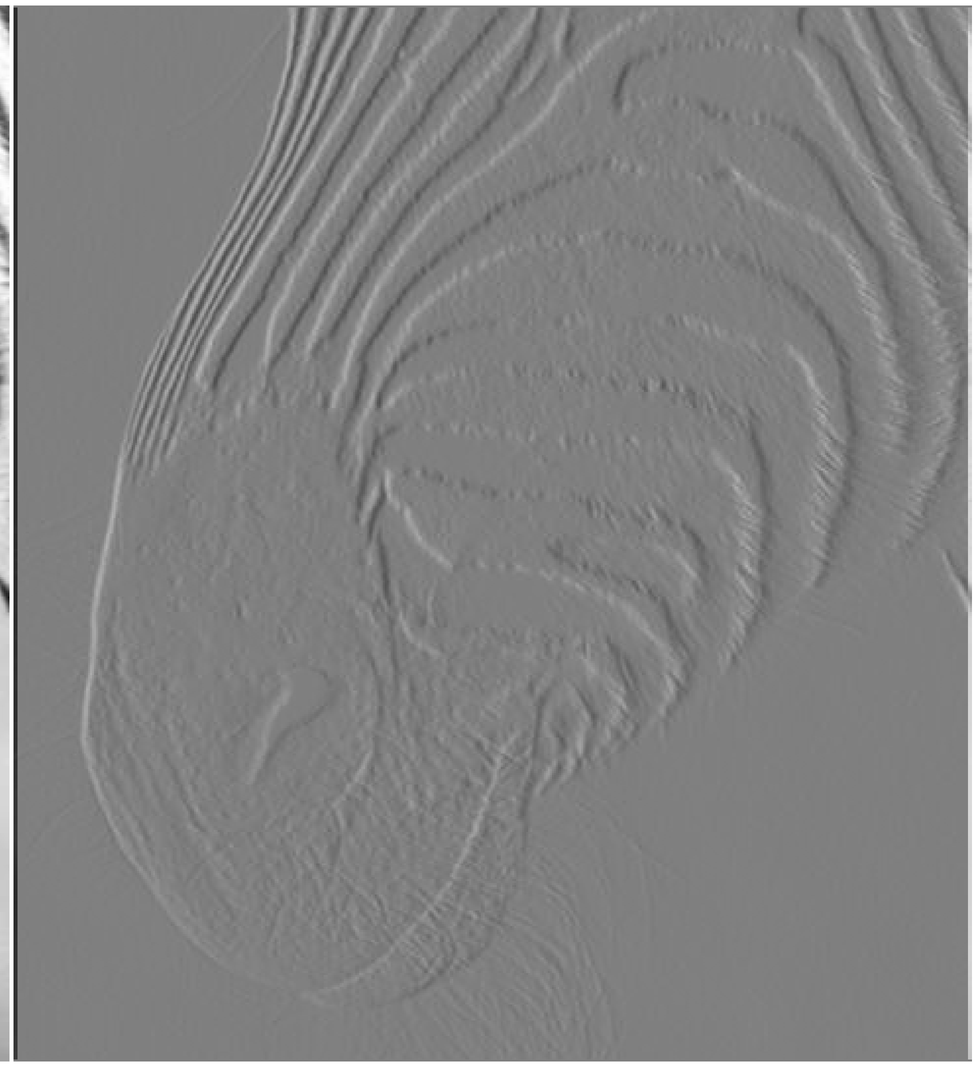
- The *edge strength* is given by the gradient magnitude

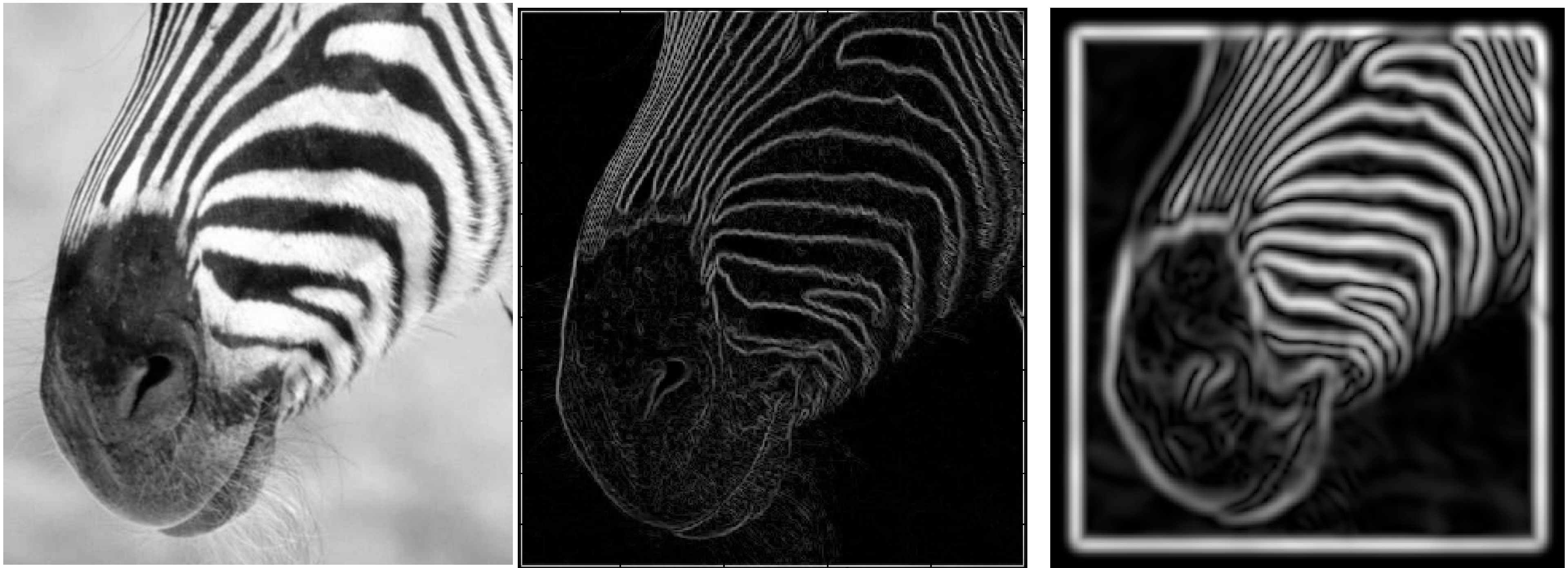
$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

Finding derivatives

Is this dI/dx or dI/dy?







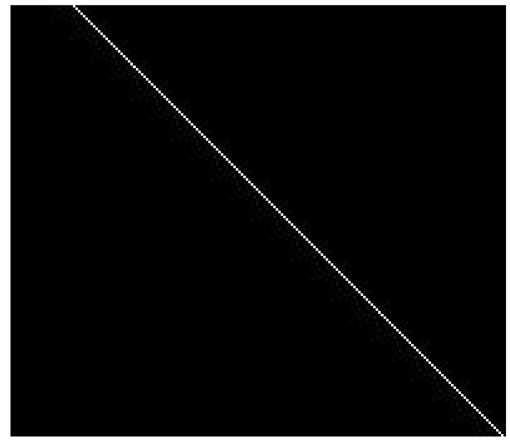
$\sigma = 1$ $\sigma = 2$

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

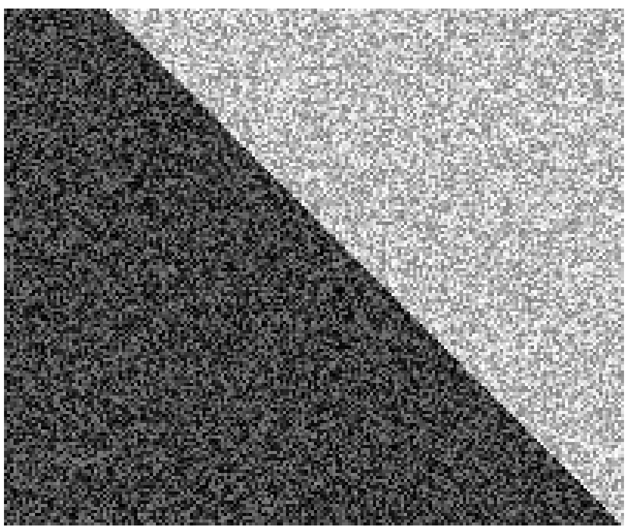
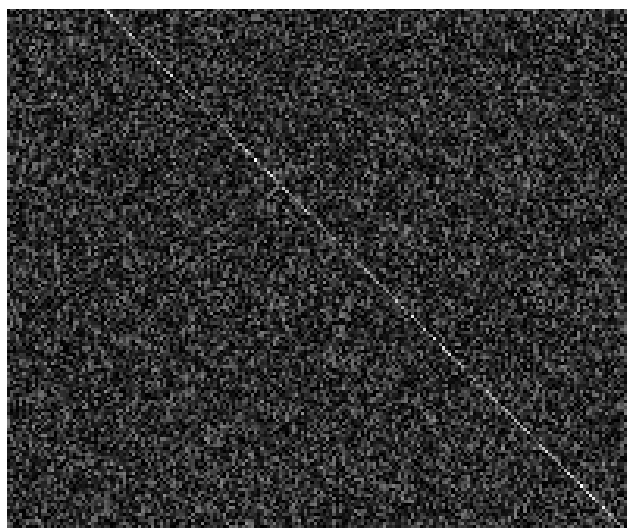
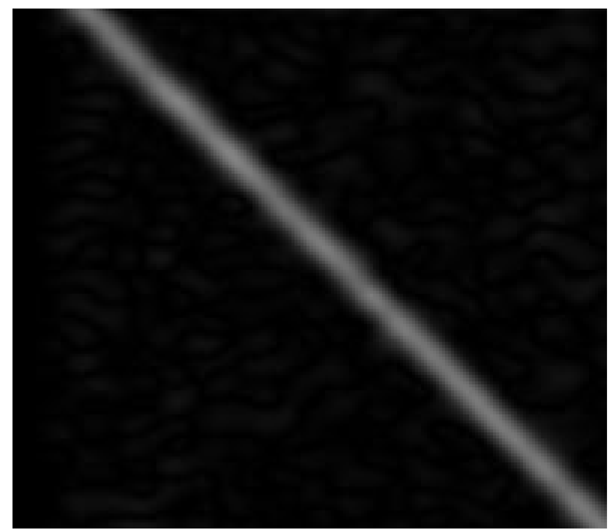


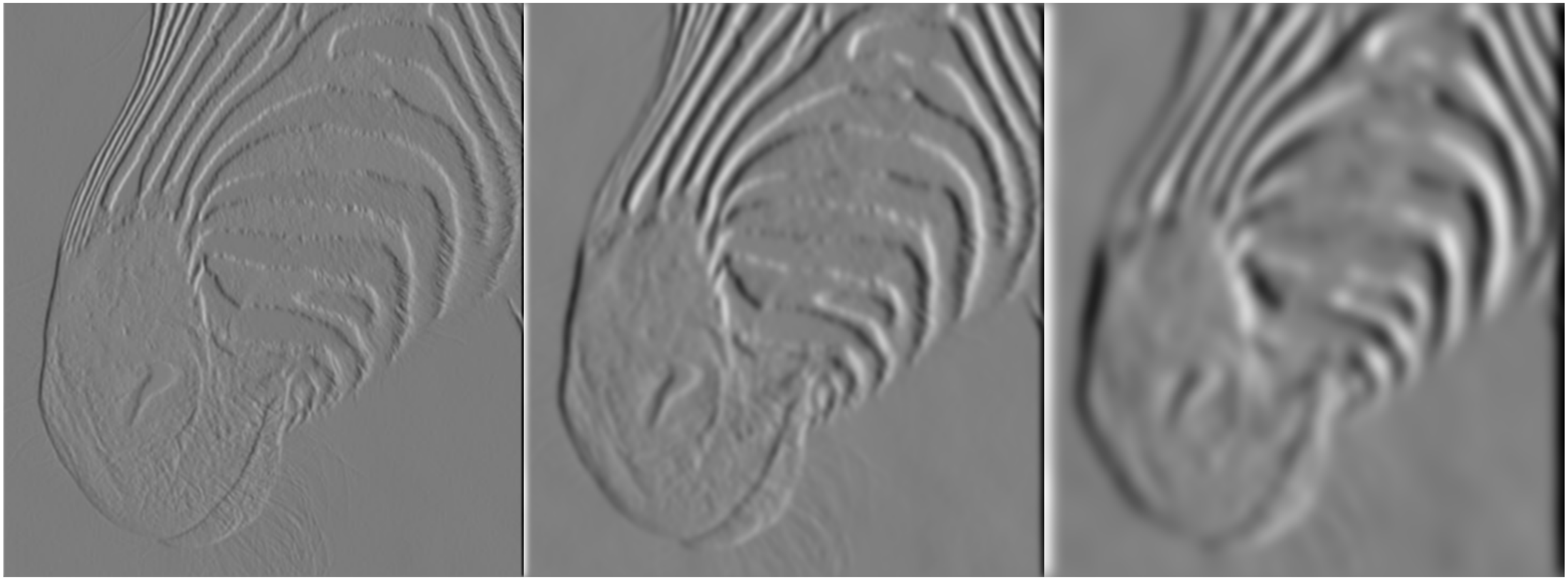
Image + Noise



Derivatives detect
edge *and* noise

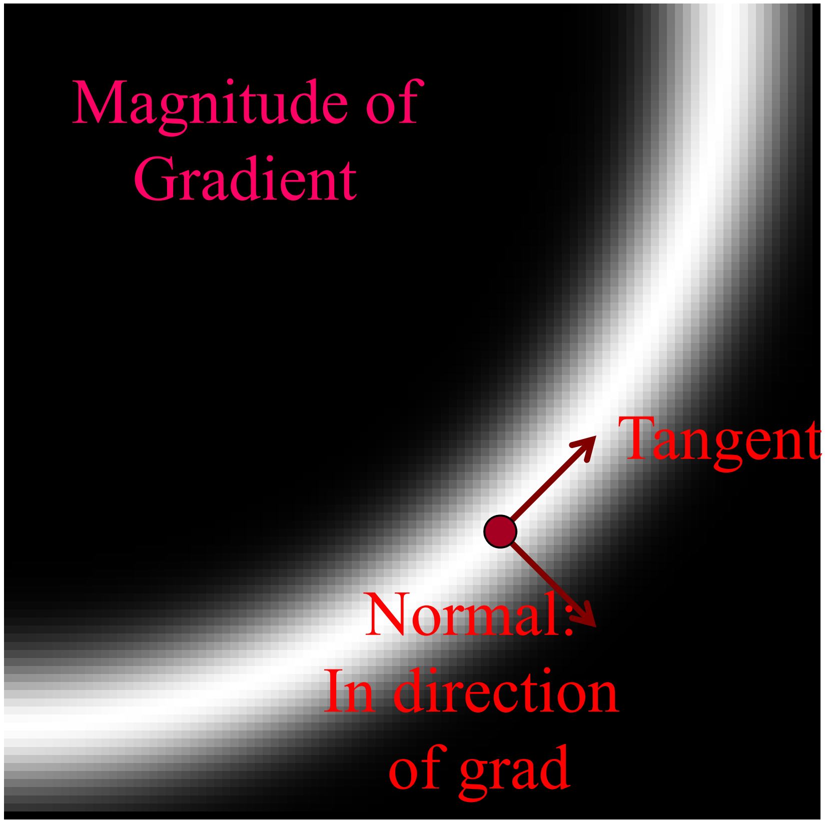


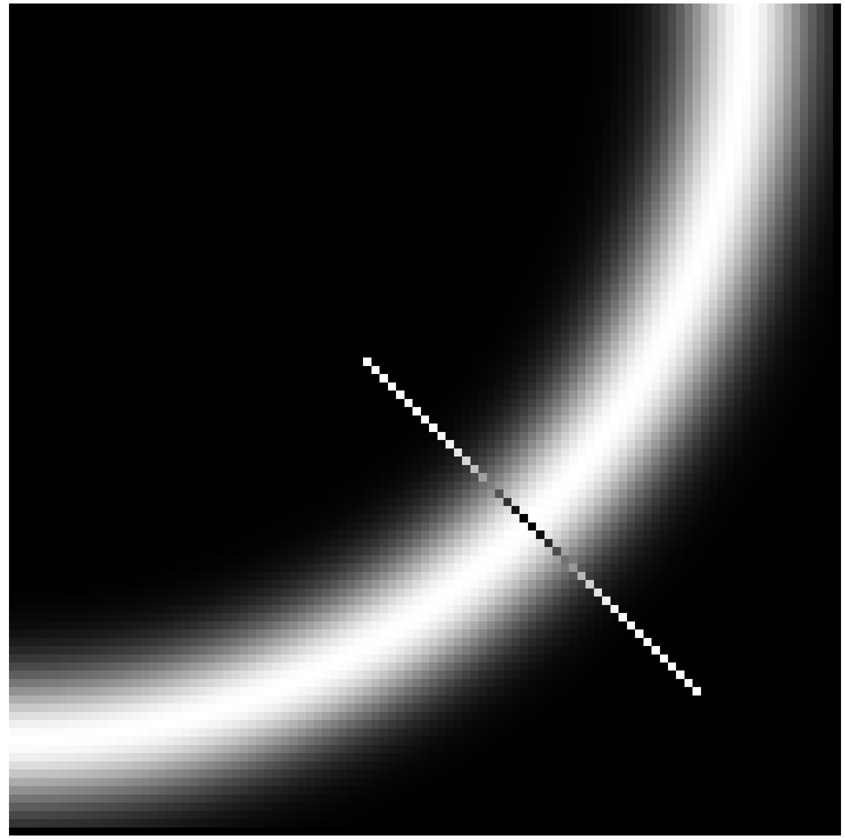
Smoothed derivative removes
noise, but blurs edge



1 pixel 3 pixels 7 pixels

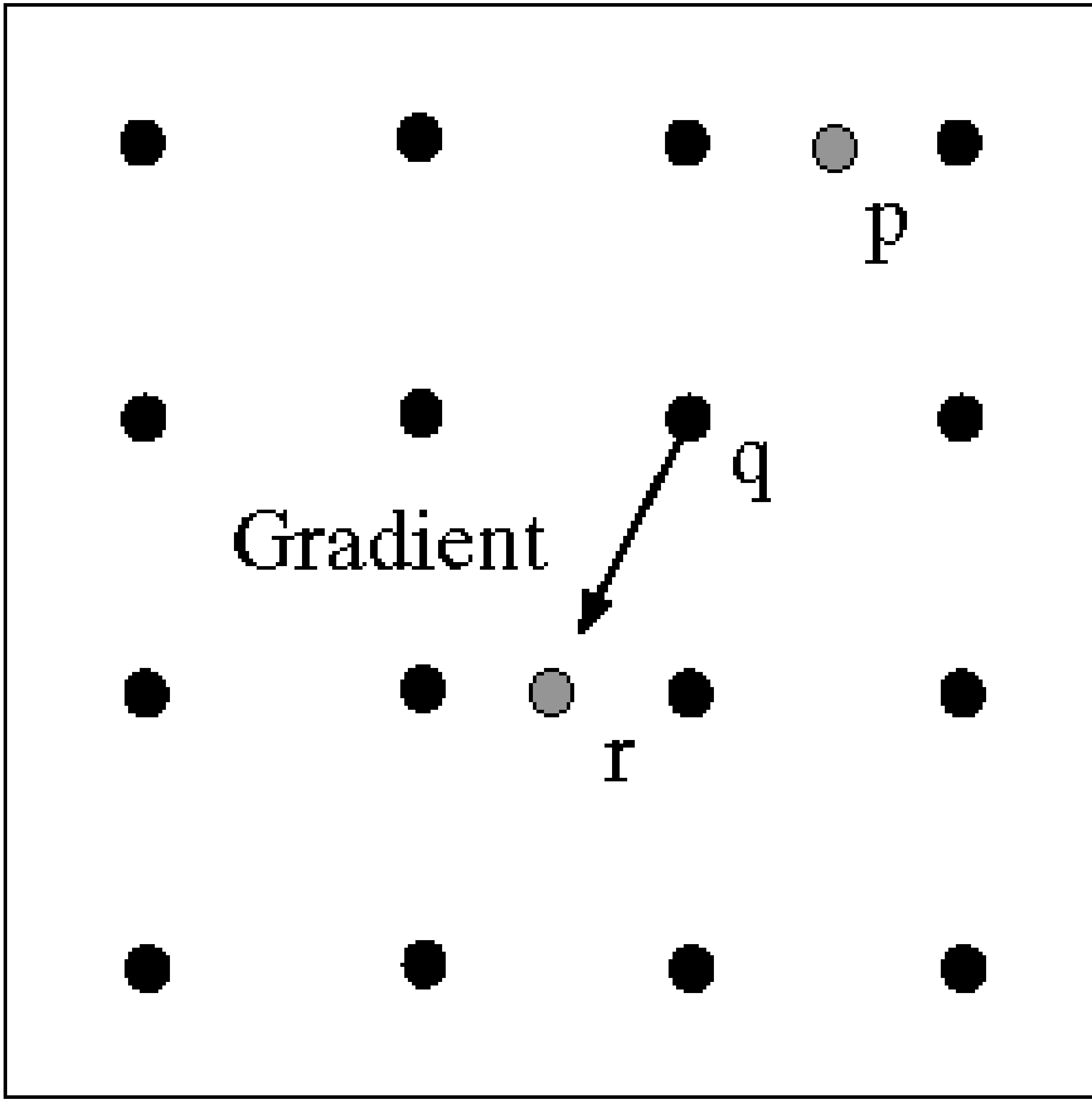
The scale of the smoothing filter affects derivative estimates





We wish to mark points along the curve where the magnitude is biggest. We can do this by looking for a maximum along a slice normal to the curve (non-maximum suppression). These points should form a curve. There are then two algorithmic issues: which point is the maximum, and where is the next point on the curve?

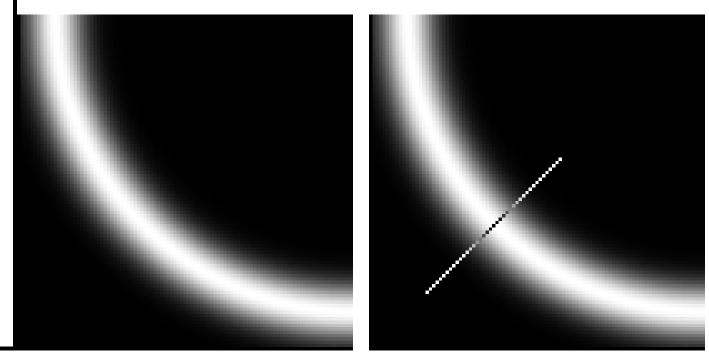
Non-maximum suppression



Using normal at q, find two points p and r on adjacent rows (or columns)

q is a maximum if $|\nabla J(q)|$ is larger than $|\nabla J(p)|$ and $|\nabla J(r)|$

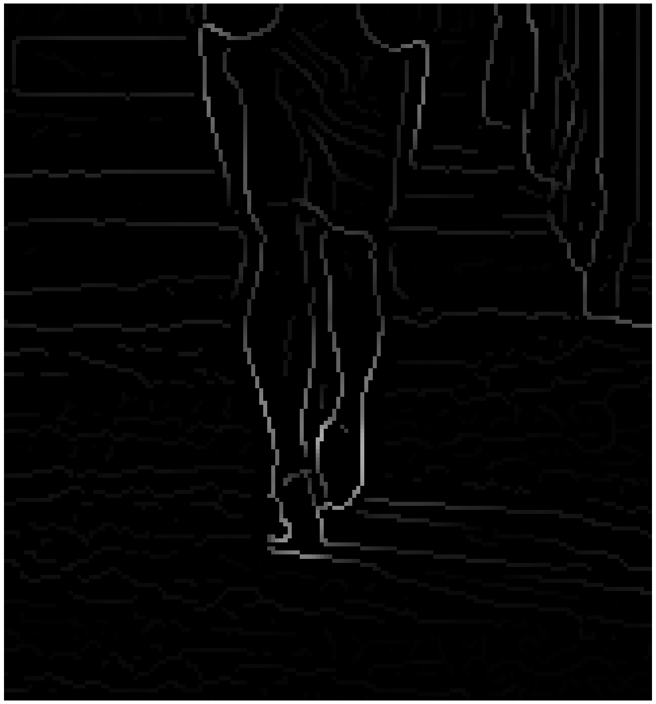
Interpolate to get values



Before Non-max Suppression

Gradient magnitude (x4 for visualization)

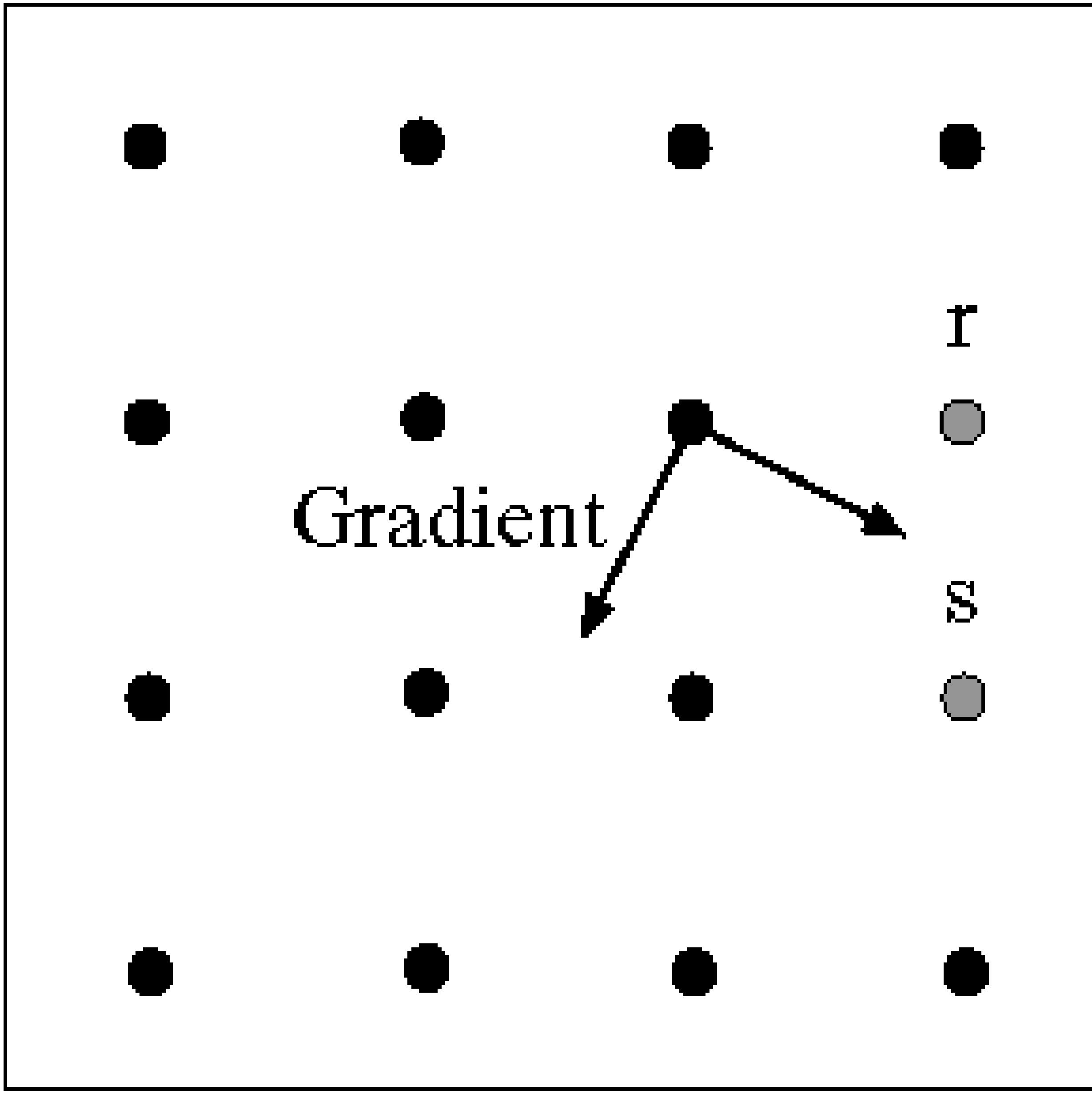
After non-max suppression



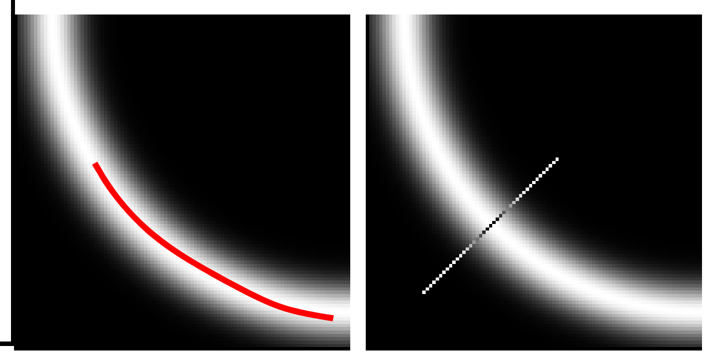
Gradient magnitude (x4 for visualization)

Non-maximum suppression

Predicting the next edge point



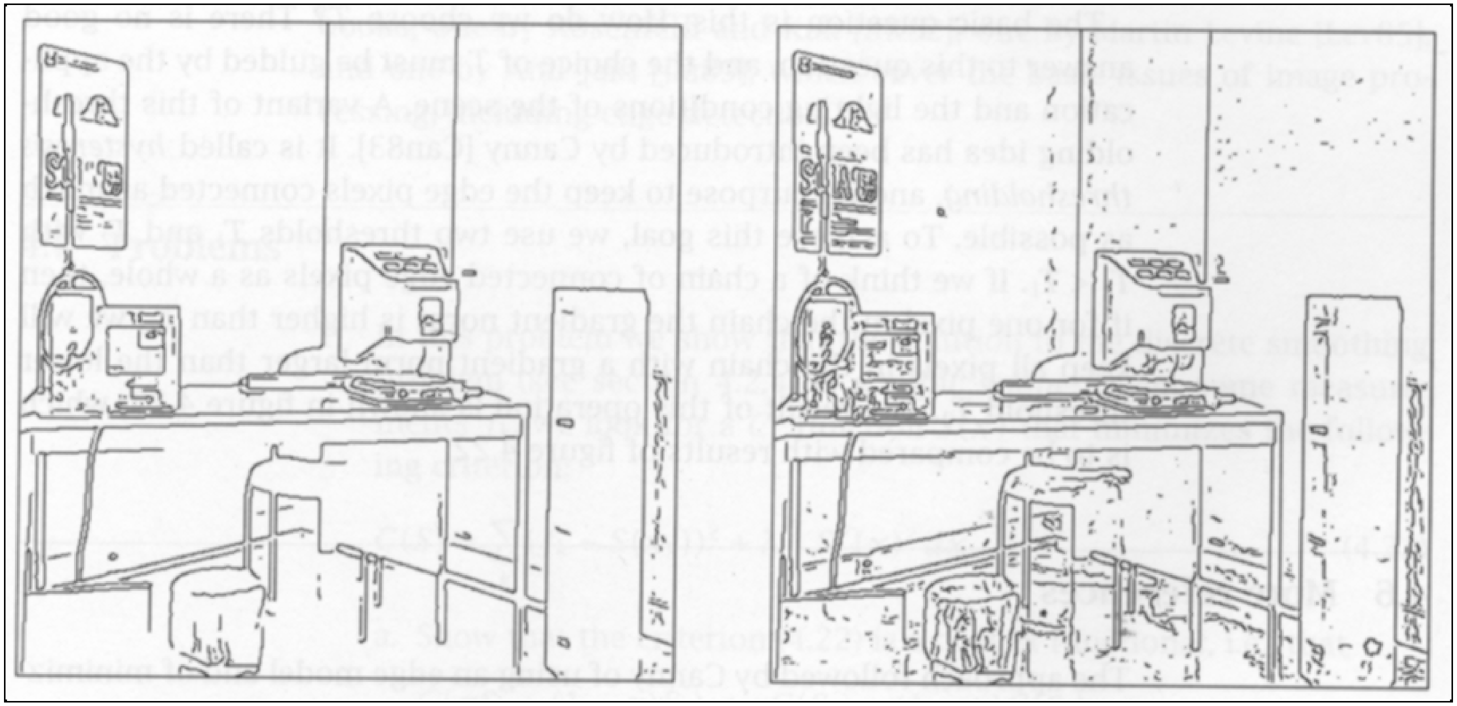
- The marked point is an edge point.
- From edge tangent (normal to gradient), predict next point along edge curve (here either r or s)
- Link together to create edge curve



Input image

Single Threshold

T=15

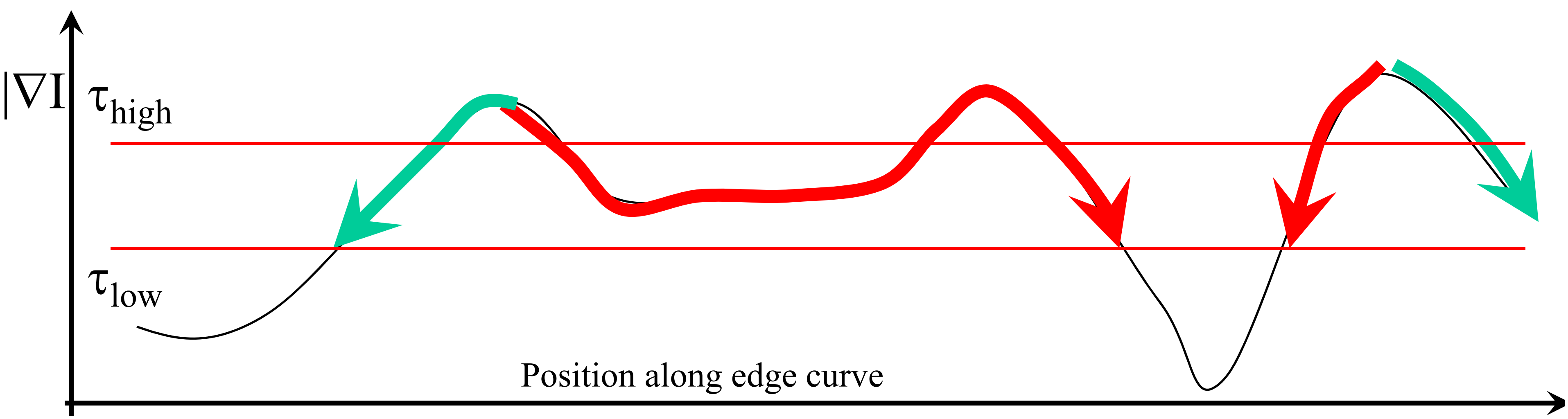


T=5

- When threshold is to high, important edges may be missed or be broken
- When threshold is too low, many extraneous edges, but non missed
- Hysteresis thresholding: Best of both

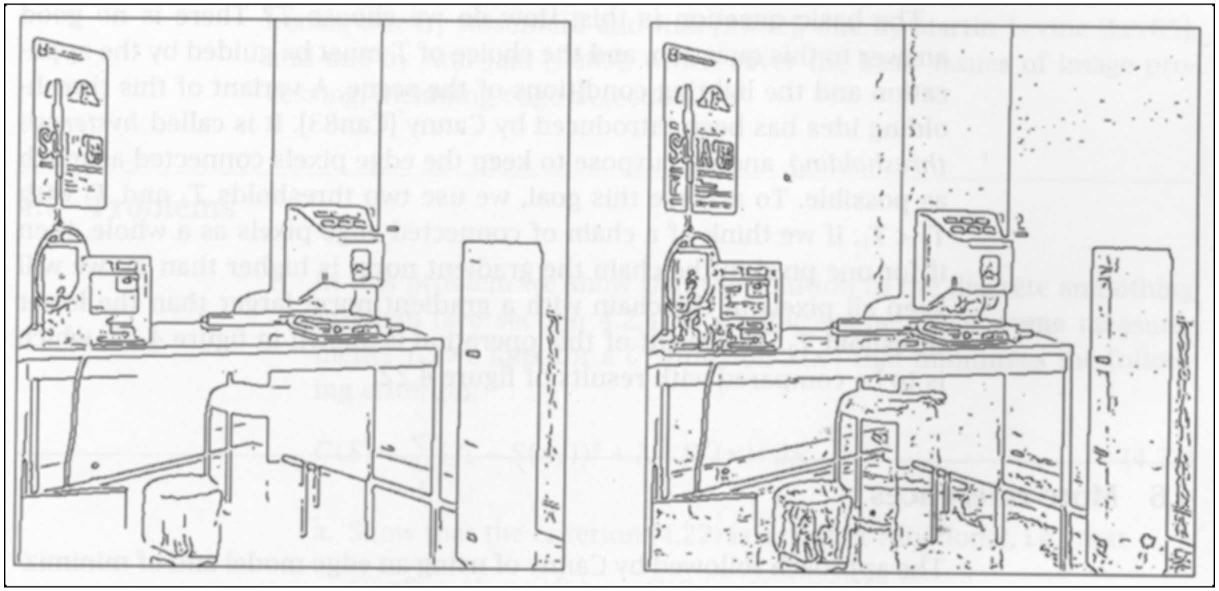
Hysteresis Thresholding

- Start tracking an edge chain at pixel location that is local maximum of gradient magnitude where gradient magnitude > $\tau_{high.}$
- Follow edge in direction orthogonal to gradient.
- Stop when gradient magnitude < τ_{low}.
  - i.e., use a high threshold to start edge curves and a low threshold to continue them.

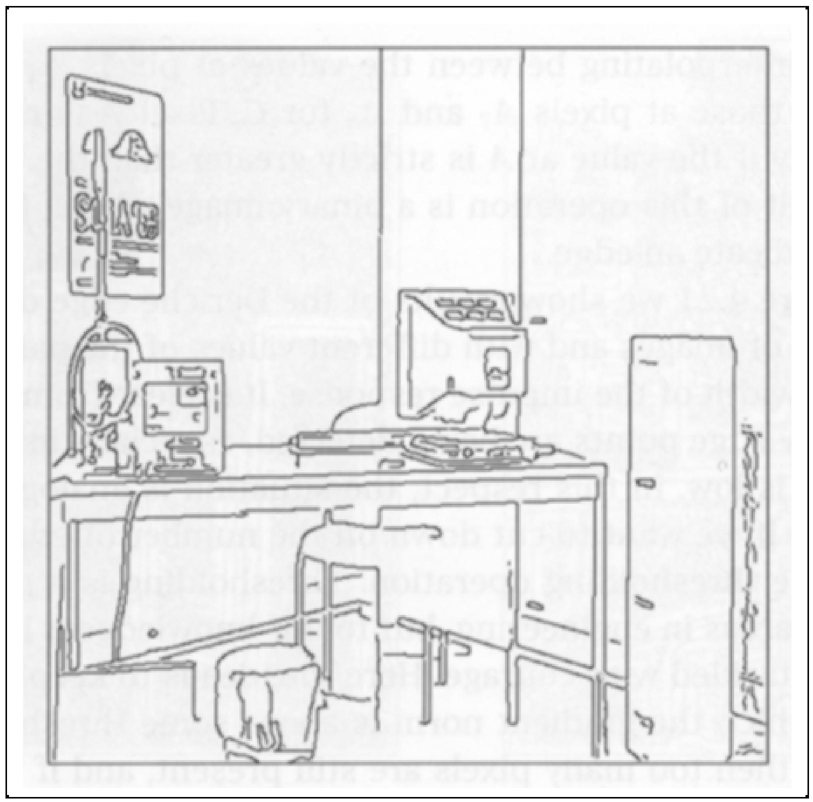


Single Threshold

T=15



T=5

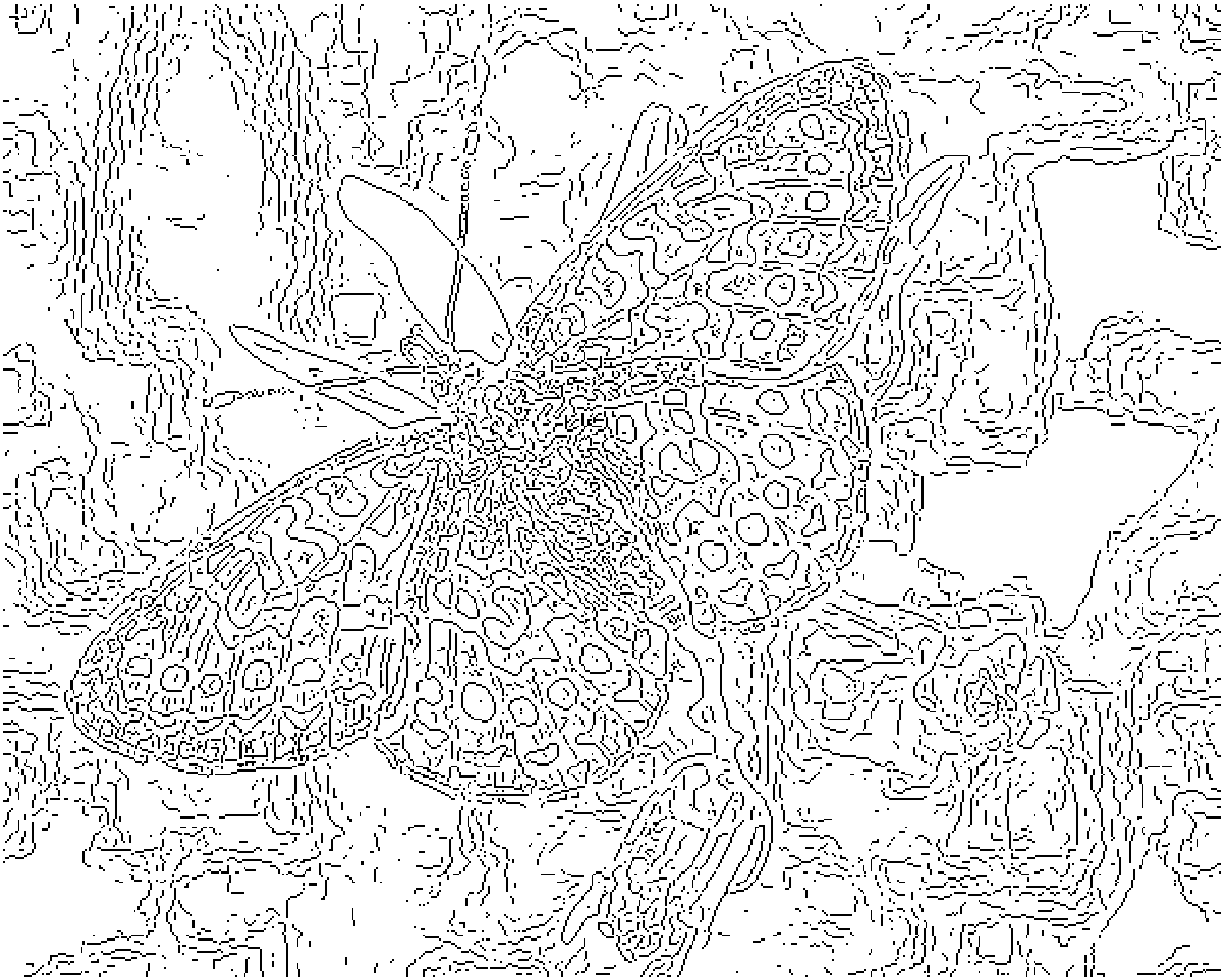


Hysteresis $T_h=15$ $T_l = 5$

Hysteresis thresholding

Canny Edge Detection Algorithm

1. Three parameters σ, τ_{high}, τ_{low}
2. Filter with symmetric Gaussian of width σ
3. Computer gradient, magnitude, direction
4. Non-maximal supression
5. Hysteresis thresholding using τ_{high}, τ_{low}



fine scale,
high
threshold

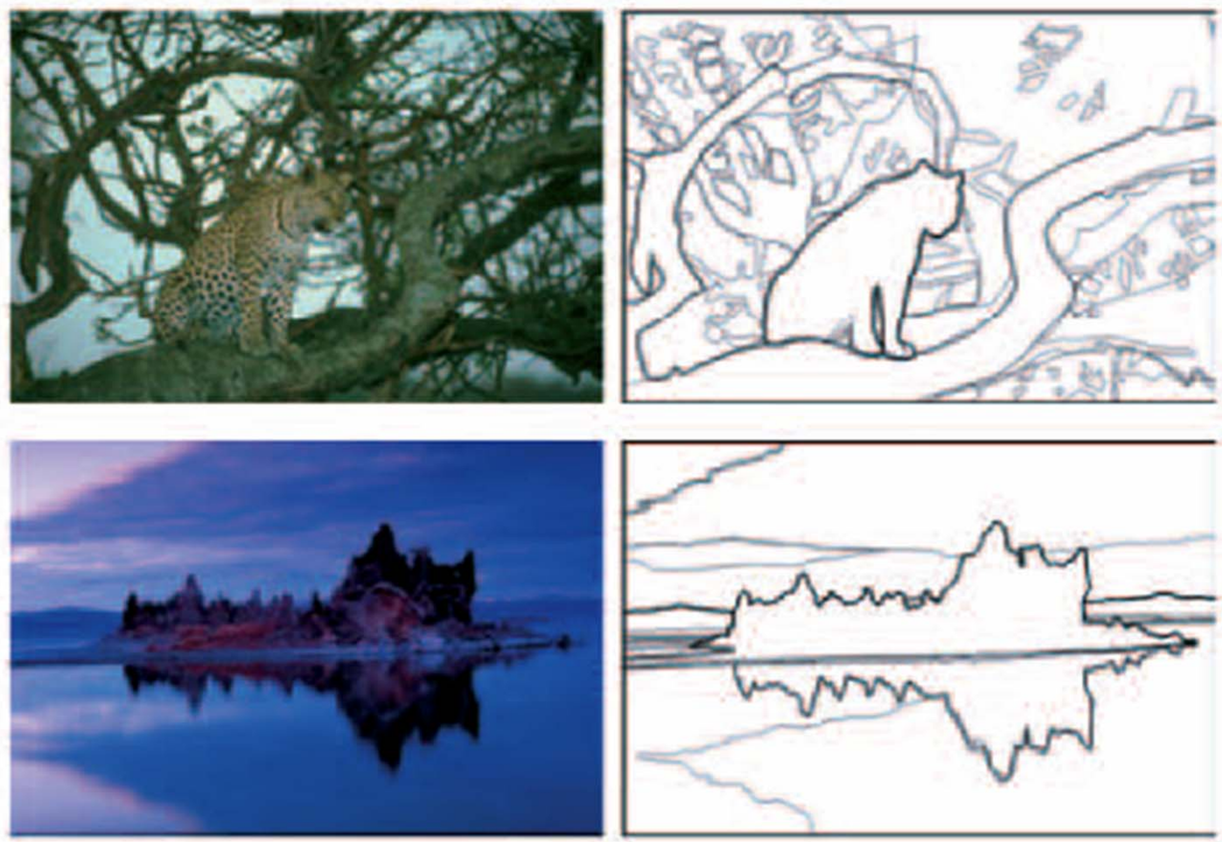
coarse
scale,
high high
threshold

coarse
scale,
low high
threshold

Why is Canny so Dominant

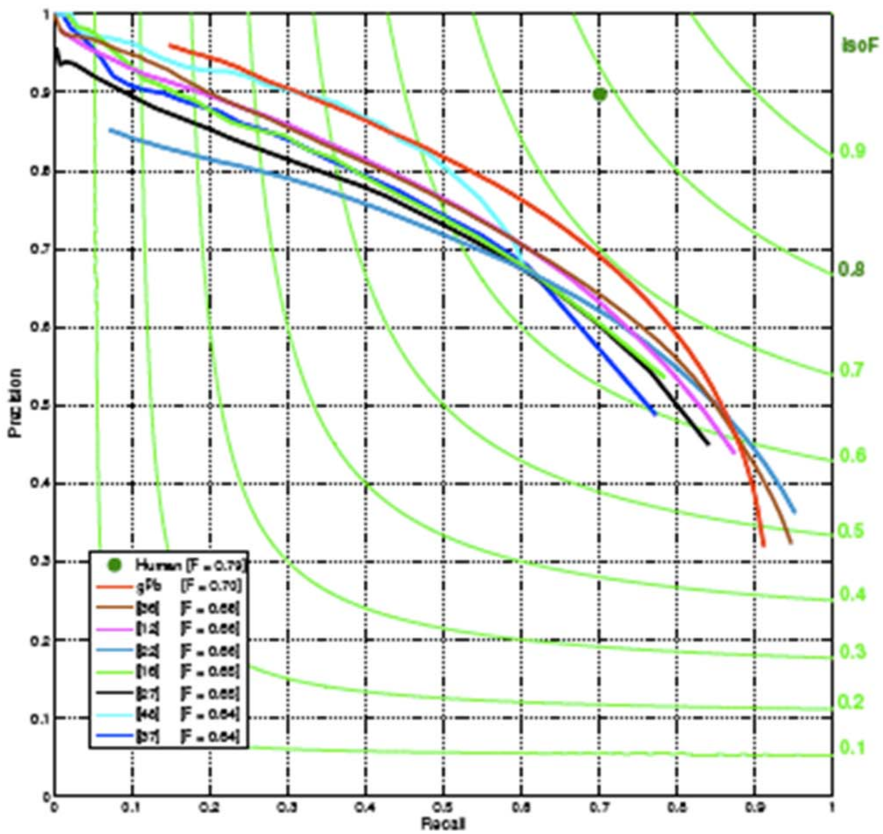
- Widely used for 30 years.
- Theory is nice
- Details are good
  - Magnitude of gradient,
  - Non-max supression
  - Hysteresis thresholding
- Most subsequent detectors weren’t much better until learning-based detectors came along
- Code was distributed

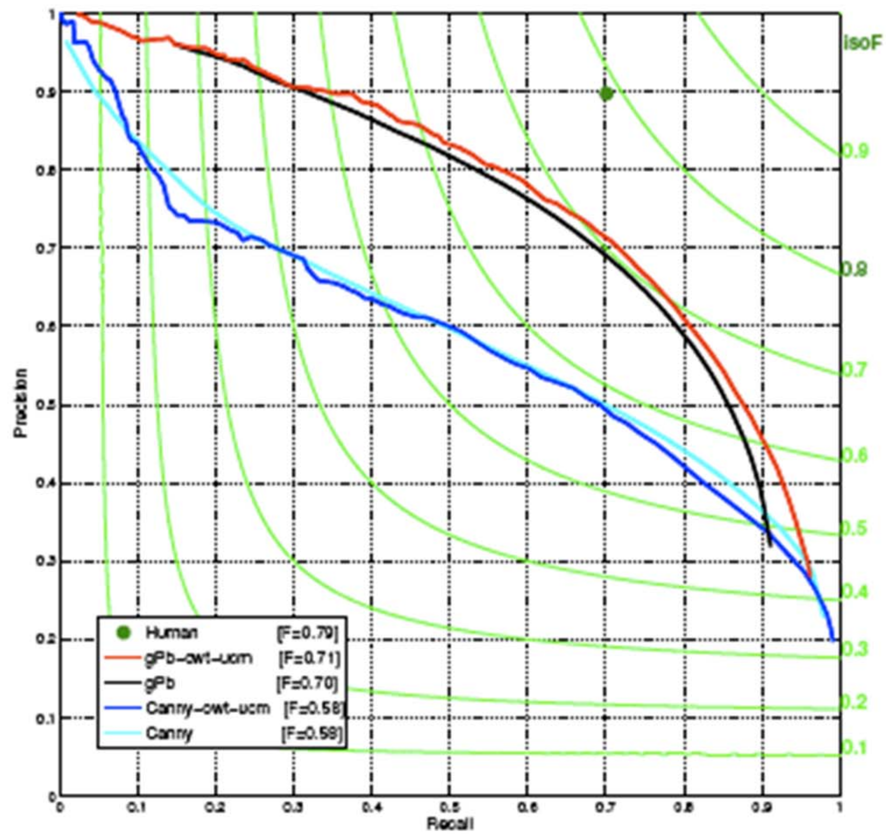
Learning-based detectors: Not edges, but boundaries



- **Brightness**
- **Color**
- **Texture**
- **Subjective contours**
- **Grouping**
- **Multiscale**

Boundary detection





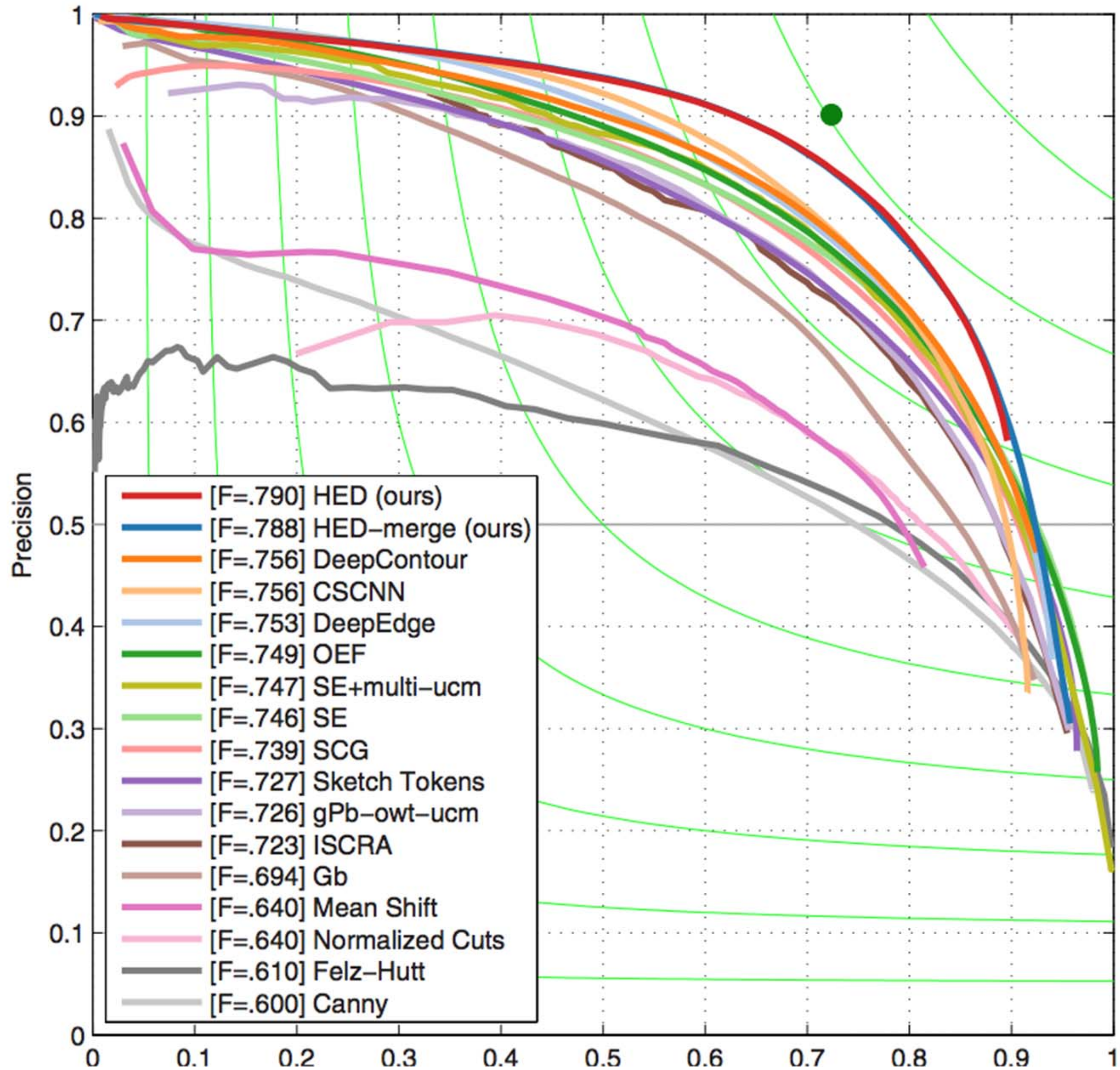
- Precision is the fraction of detections that are true positives rather than false positives, while recall is the fraction of true positives that are detected rather than missed.

- From Contours to Regions: An Empirical Evaluation, Arbelaez, M. Maire, C. Fowlkes, and J. Malik, CVPR 2008

Learned Edge Detectors

- Dollar, Piotr, Zhuowen Tu, and Serge Belongie. "Supervised learning of edges and object boundaries." Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on. Vol. 2. IEEE, 2006
- Dollár, Piotr, and C. Lawrence Zitnick. "Structured forests for fast edge detection." Proceedings of the IEEE International Conference on Computer Vision. 2013.
- Xie, Saining, and Zhuowen Tu. ." Proceedings of the IEEE international conference on computer vision. 2015.
- Long, Jonathan, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation." Proceedings of the IEEE conference on computer vision and pattern recognition. 2015

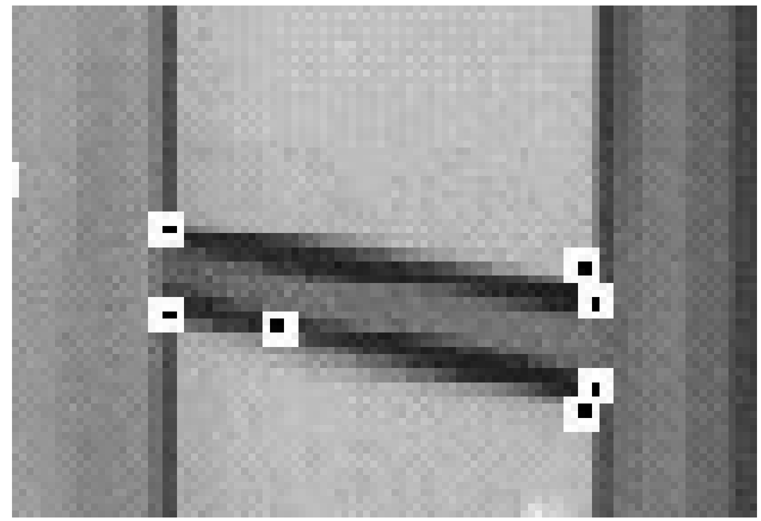
HED Performance



Xie and Tu. "Holistically-nested edge detection." ICCV 2015

Corner Detection

Feature extraction: Corners

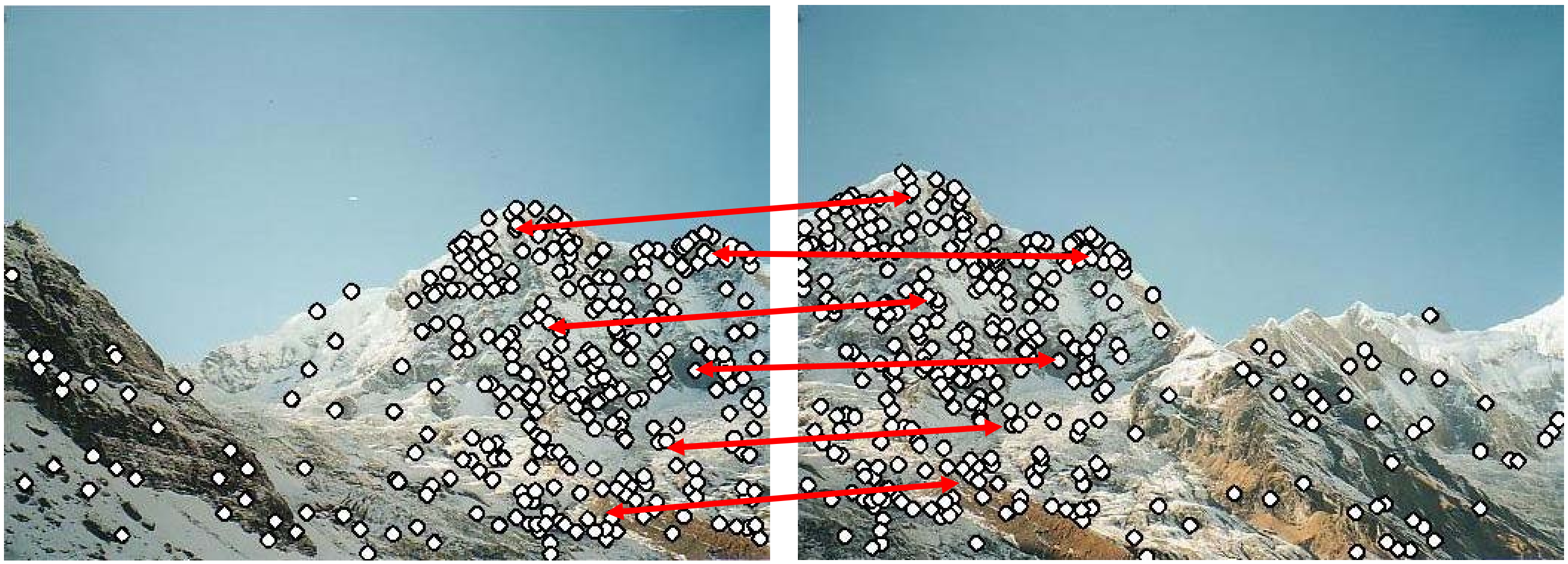


Why extract features?

- 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?

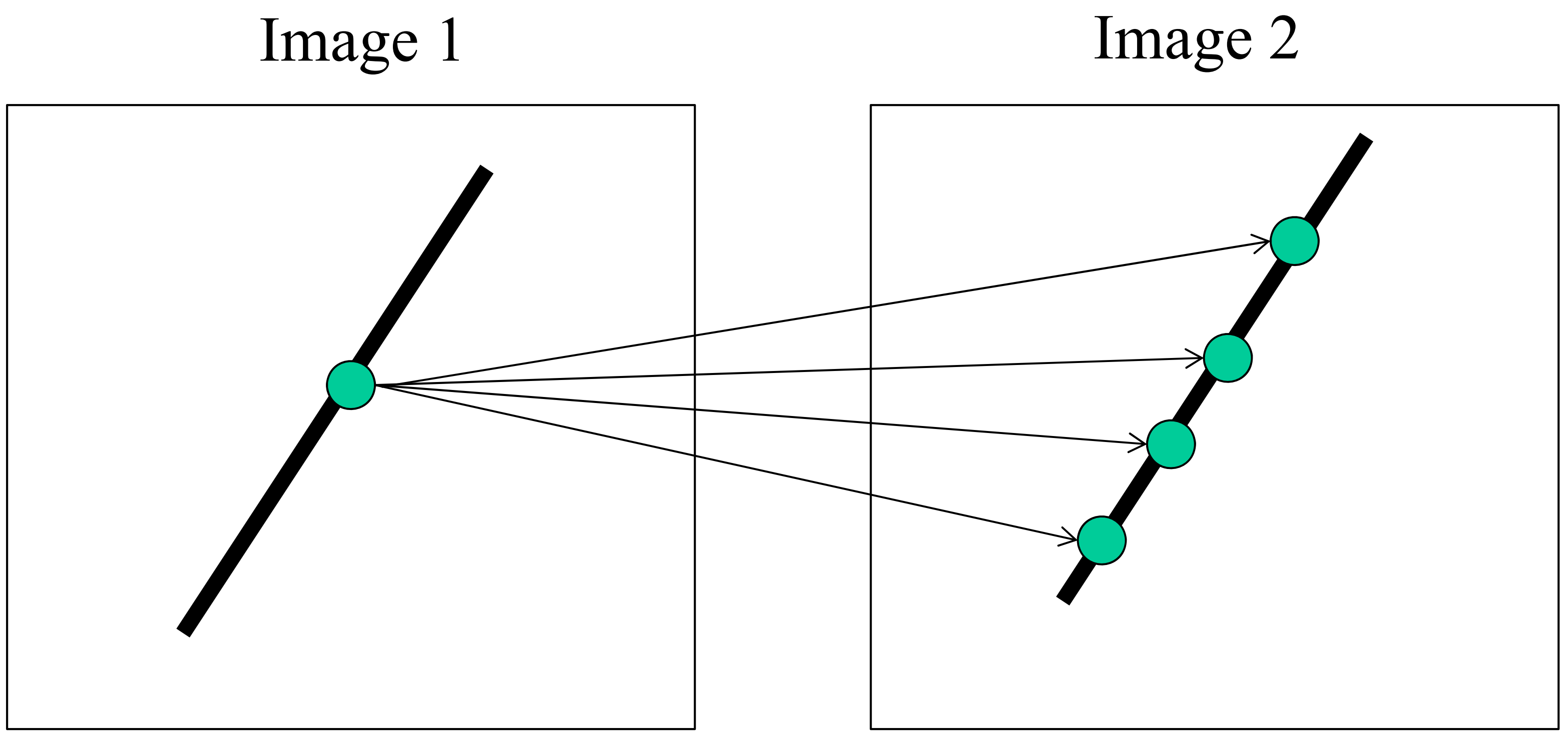
Step 1: extract features

Step 2: match features

Step 3: align images

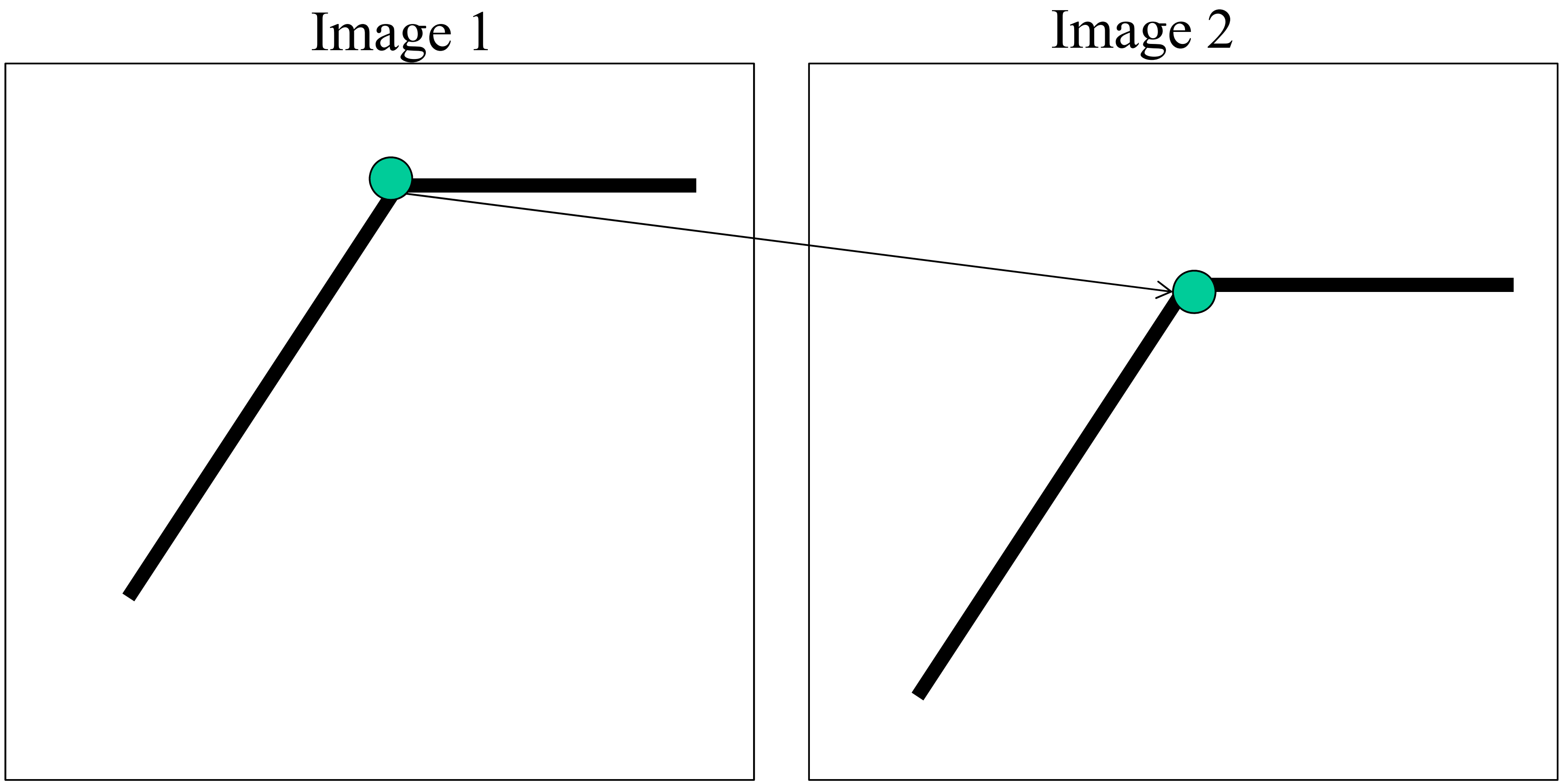
Corners contain more info than lines.

- A point on a line is hard to match.



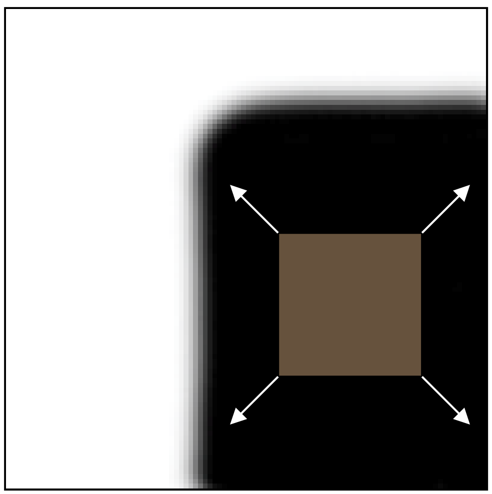
Corners contain more info than lines.

- A corner is easier to match

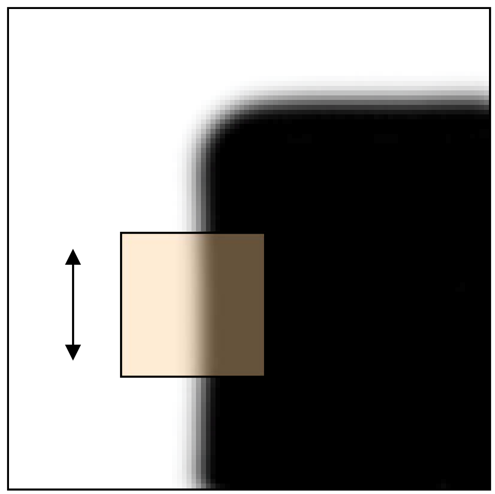


The Basic Idea

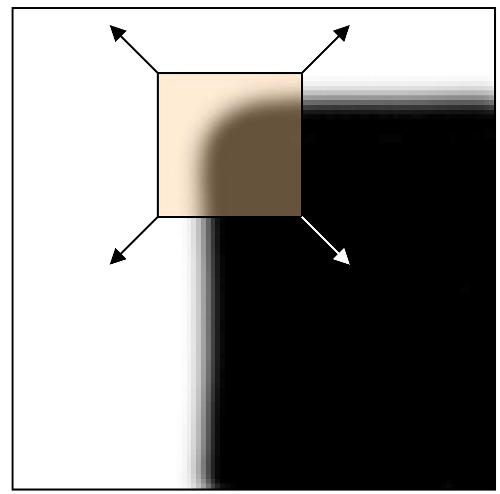
- We should easily recognize the point by looking through a small window
- Shifting a window in *any direction* should give *a large change* in intensity



"flat" region: no change in all directions



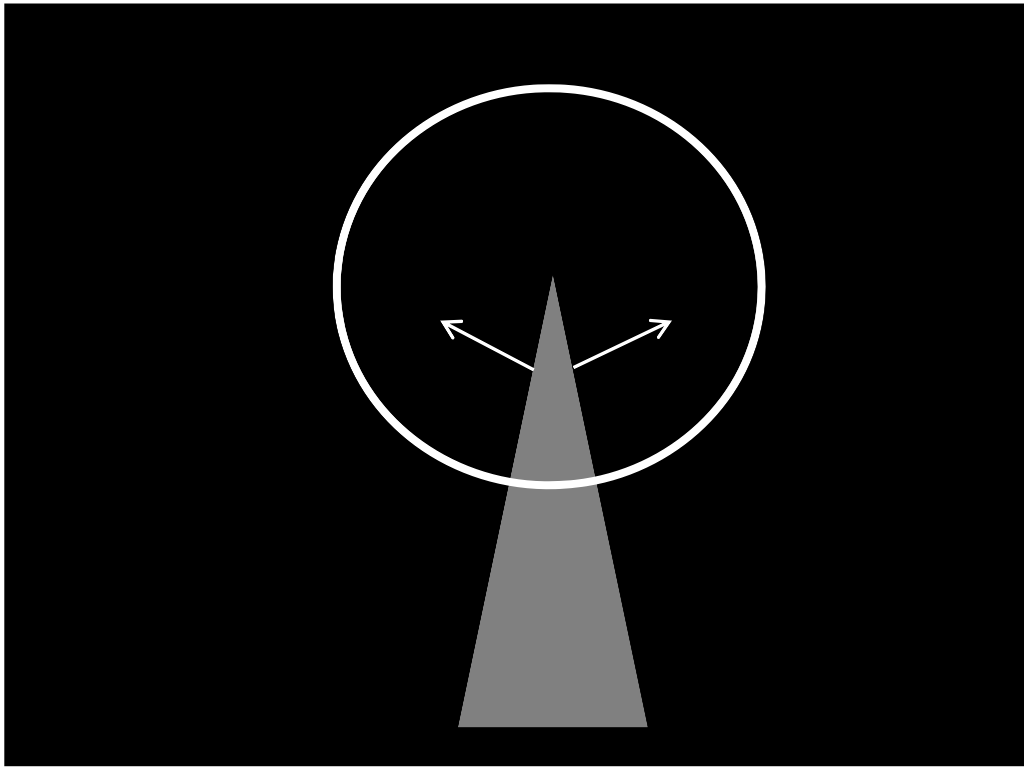
"edge": no change along the edge direction



"corner": significant change in all directions

Source: A. Efros

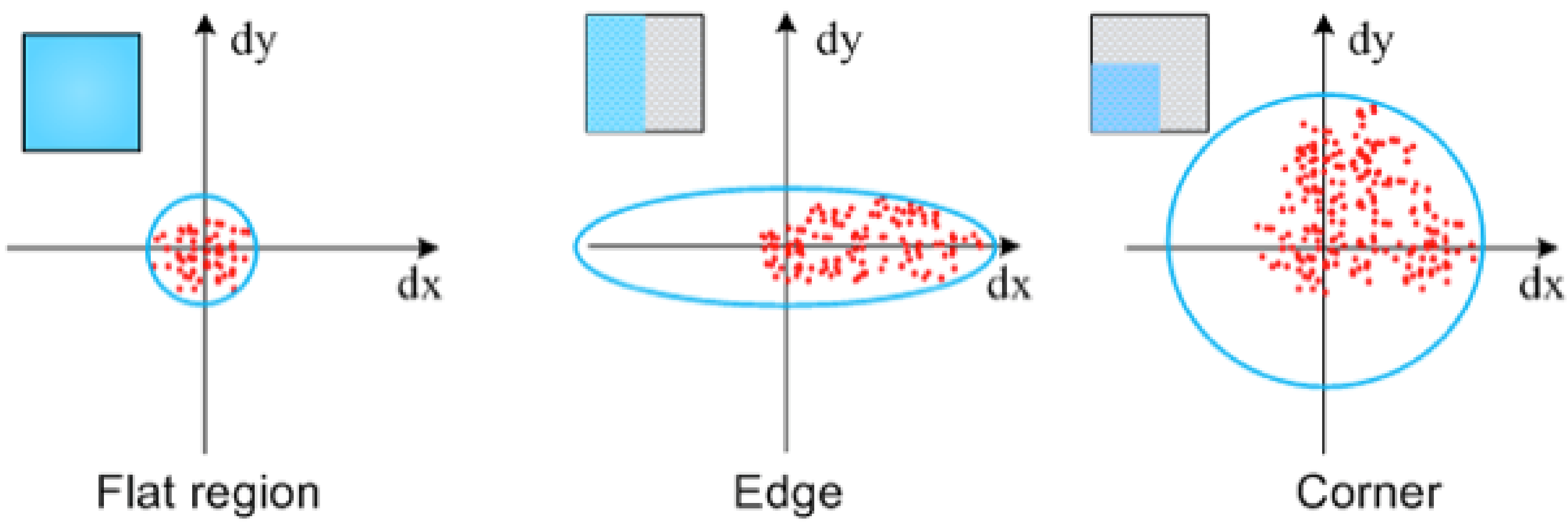
Finding Corners



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

Formula for Finding Corners

Shi-Tomasi Detector

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:

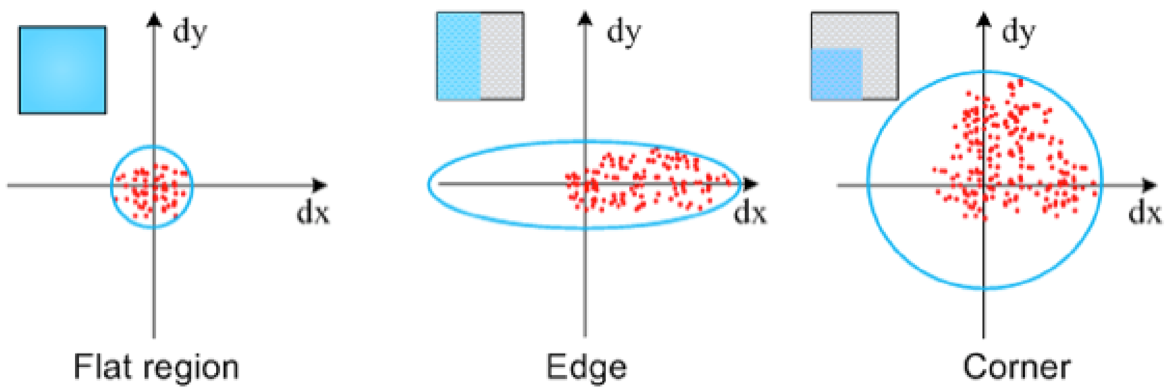
$$C = R^{-1} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} R = R^T \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} R$$

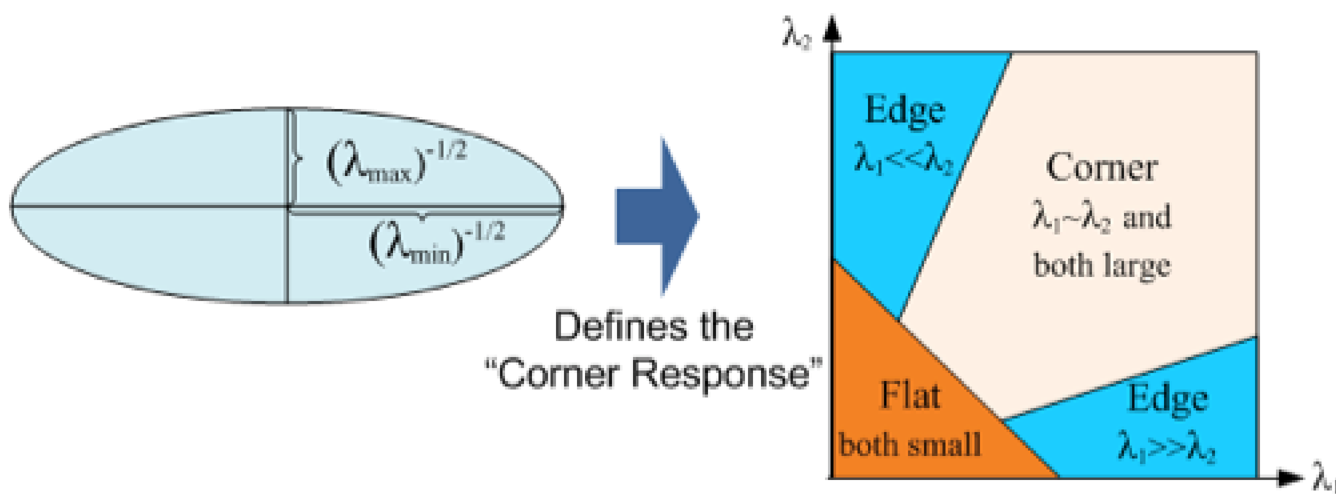
where R is a 2x2 rotation matrix and λ_1 and λ_2 are non-negative.

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 $\det(C-\lambda I) = 0$ for λ.

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 $\lambda_2 >> 0?$





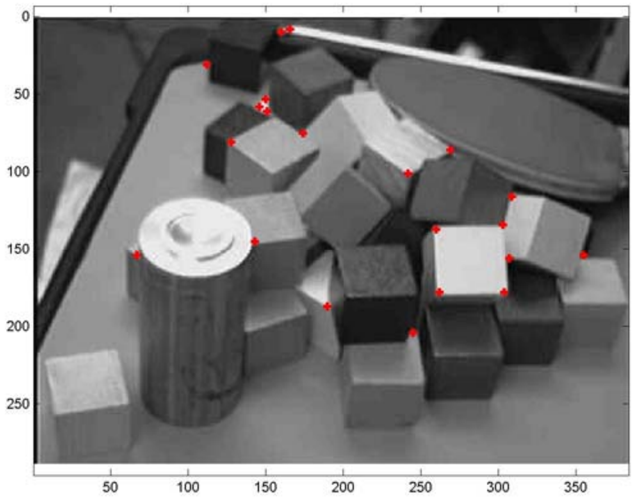
Shi-Tomasi Corner Detector

- 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 large, 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 $e(x,y) > \tau$

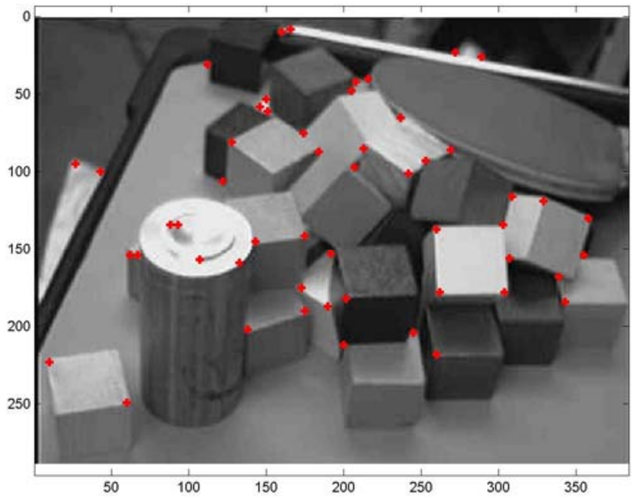
Parameters: Gaussian std. dev, window size, threshold

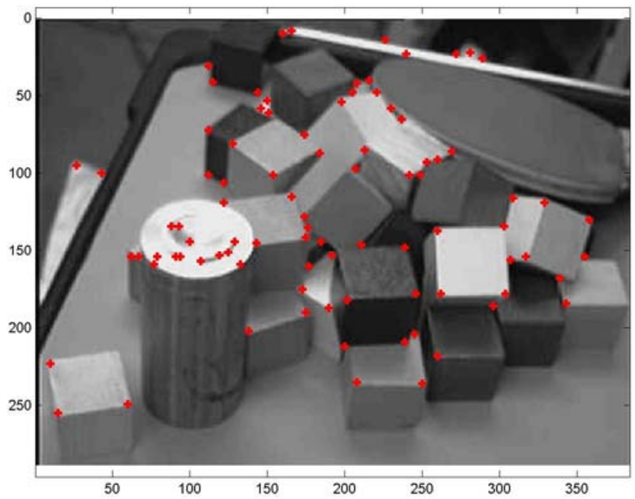
Corner Detection Sample Results

Threshold=25,000



Threshold=10,000





Threshold=5,000

Next Lecture

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