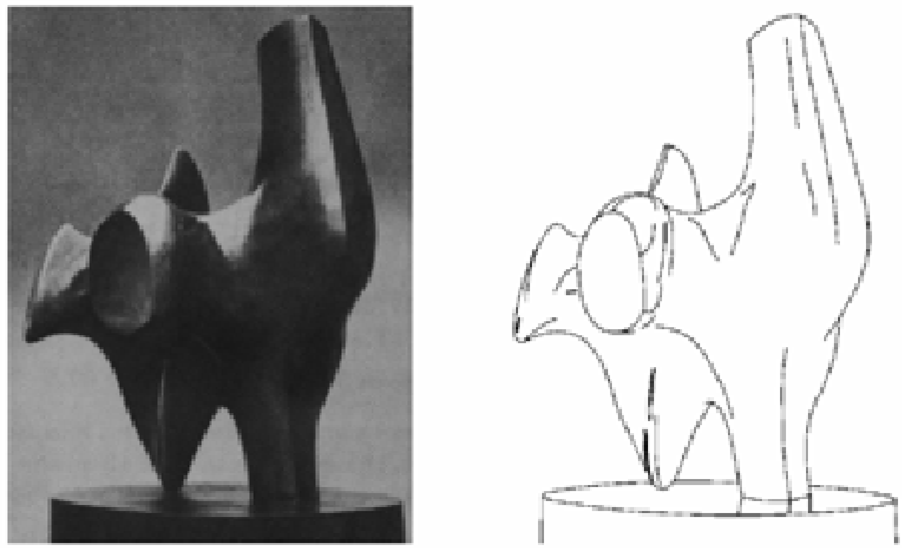
Edge Detection

CS 554 – Computer Vision
Pinar Duygulu
Bilkent University

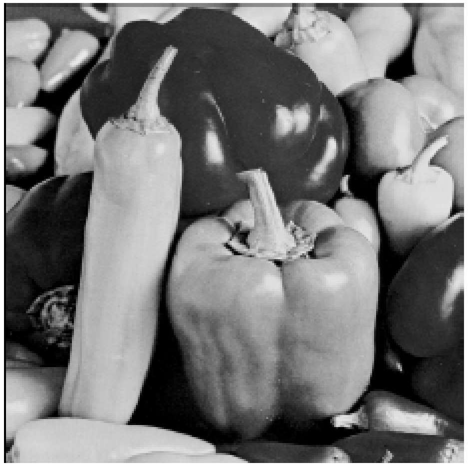
Edge Detection



Convert a 2D image into a set of curves

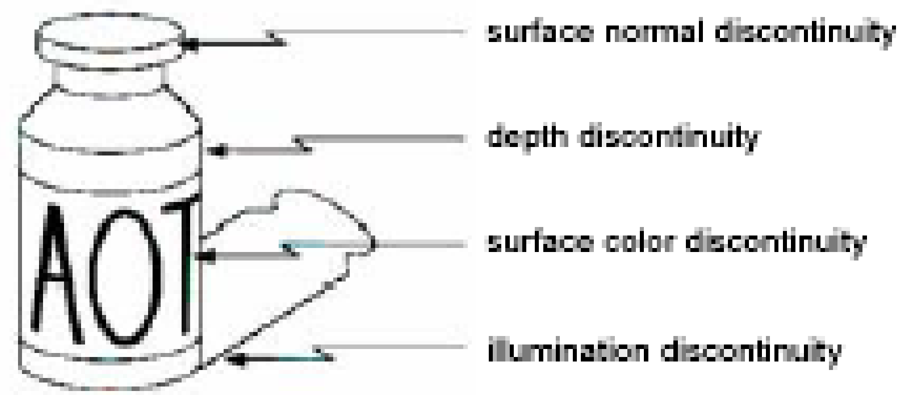
- Extracts salient features of the scene
- More compact than pixels

Edge Detection



Edge Detection

- Edge : points in the image where brightness change
- Sharp changes in the image brightness occur
  - Object boundaries
    - A light object may lie on a dark background or a dark object may lie on a light background
  - Reflectance changes
    - May have quite different characteristics – zebras have stripes, and leopards have spots
  - Cast shadows
  - Sharp changes in surface orientation



Where are the edges?

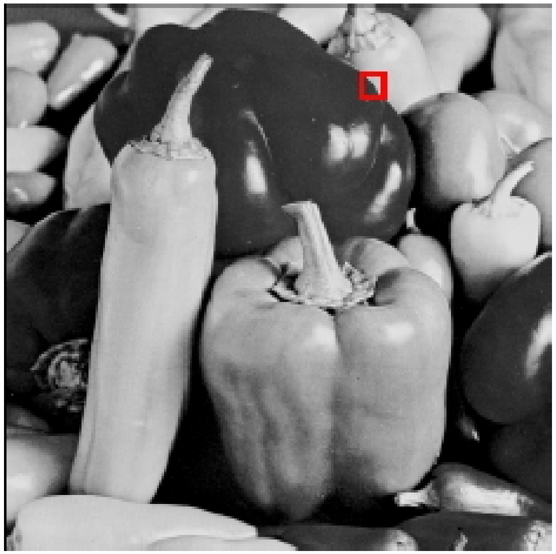
It is hard to tell where the edges are
It requires high level information

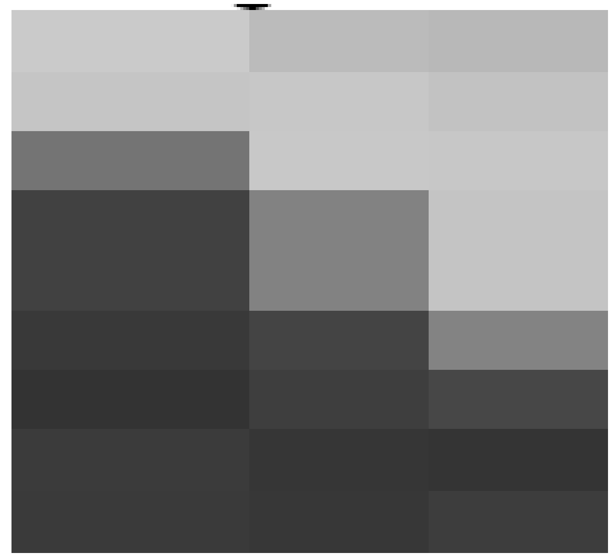
Why important?

- Information reduction
  - replace image by a cartoon in which objects and surface markings are outlined
  - these are the most informative parts of the image
- Biological plausibility
  - initial stages of mammalian vision systems involve detection of edges and local features

adapted from Larry Davis, University of Maryland

What is an edge?

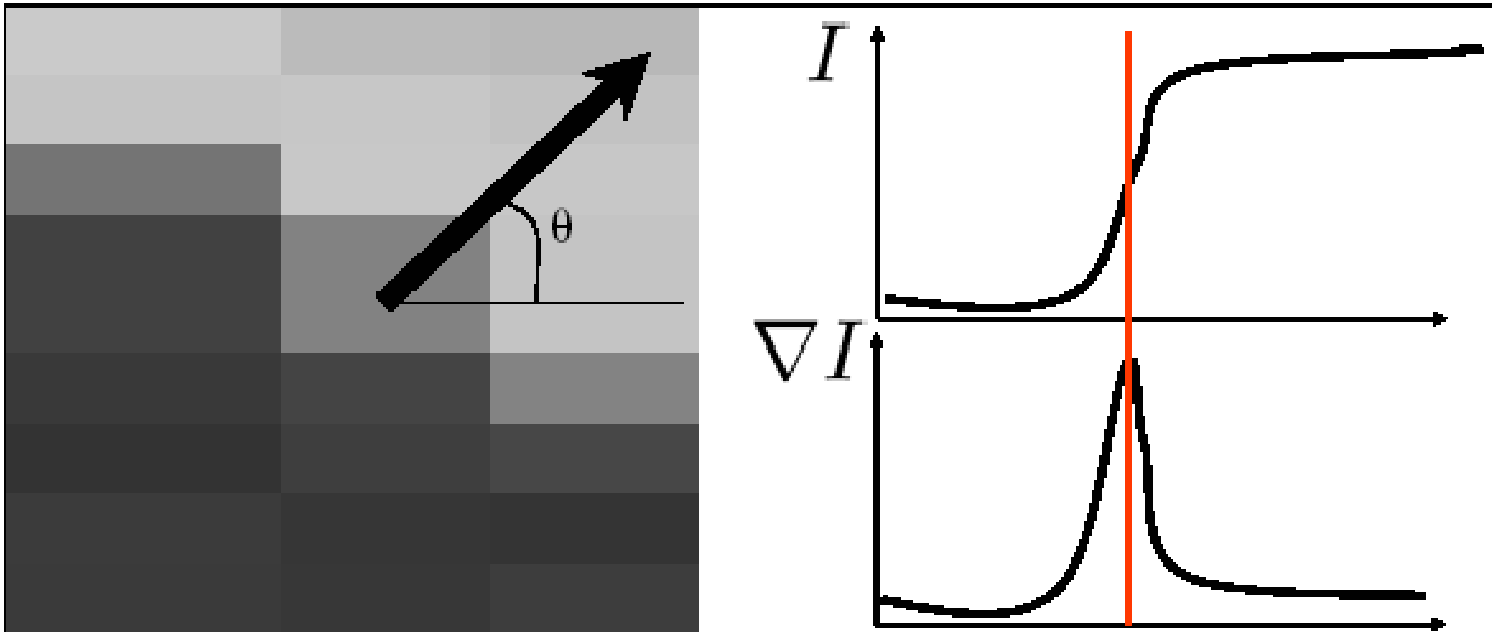




- Find the peak
  - Should be a local maximum
  - Should be sufficiently high

adapted from Martial Hebert, CMU

What is an edge?



Edge pixels are at local maxima of gradient magnitude

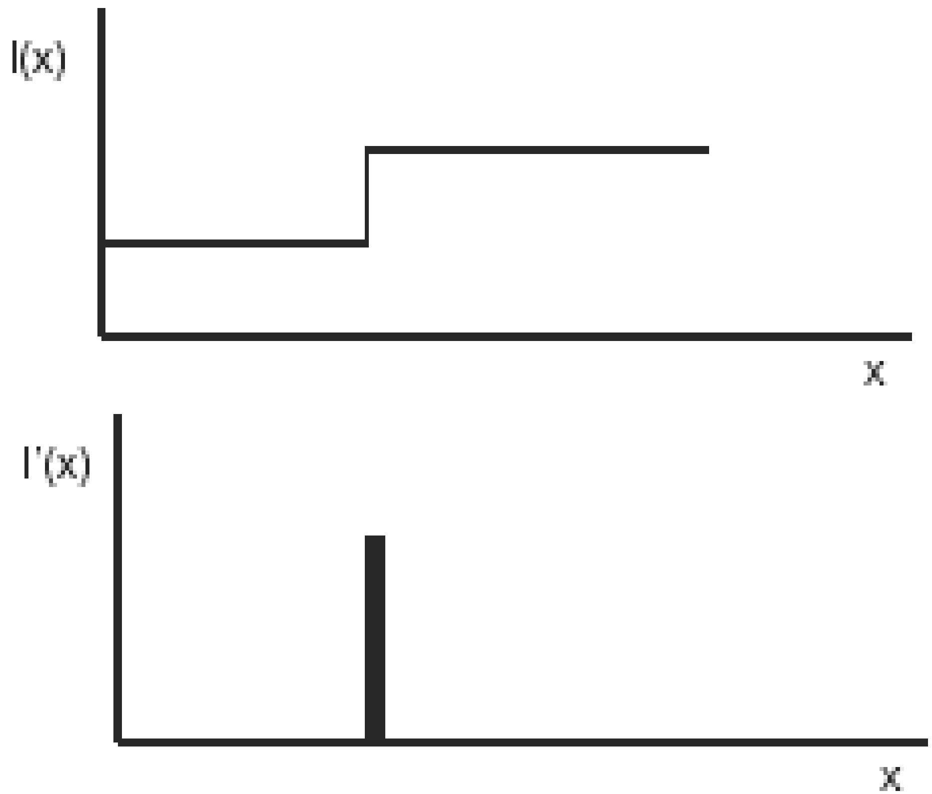
Gradient direction is always perpendicular to edge direction

$$|\nabla I| = \sqrt{\left(\frac{\partial I}{\partial x}\right)^2 + \left(\frac{\partial I}{\partial y}\right)^2} \quad \theta = atan2\left(\frac{\partial I}{\partial y}, \frac{\partial I}{\partial x}\right)$$

adapted from Martial Hebert, CMU

What is an edge?

- An ideal edge is a step function

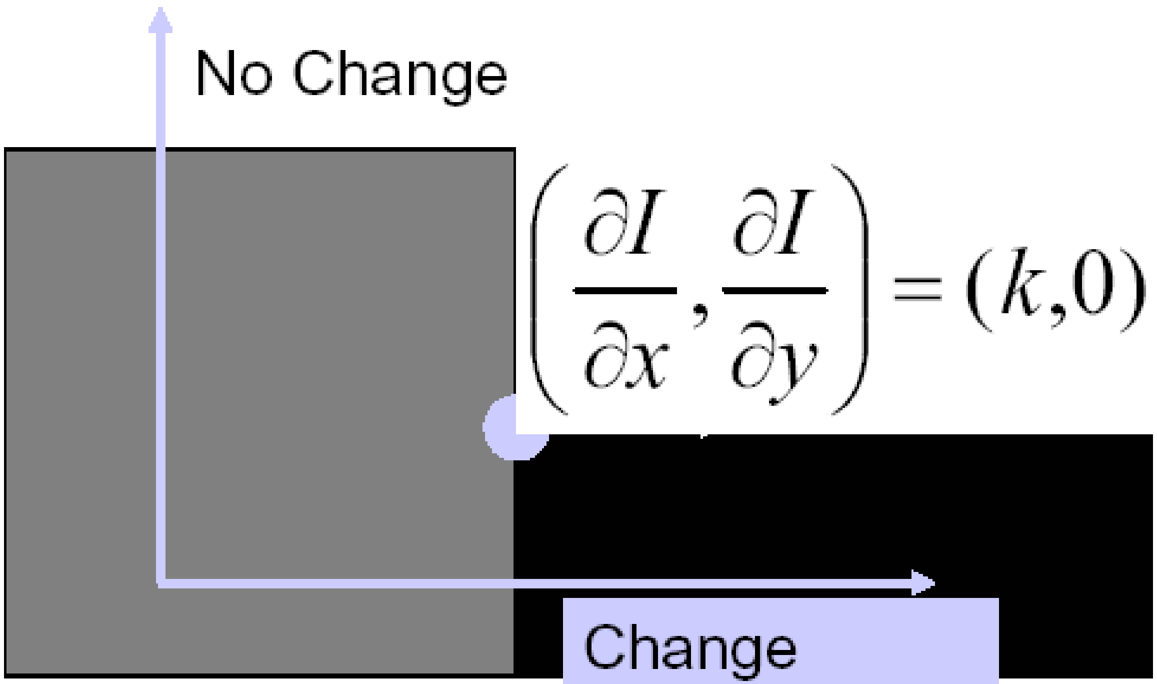


adapted from Larry Davis, University of Maryland

General Strategy

- Determine image gradient
- Mask points where gradient is particularly large with respect to neighbors (ideally curves of such points)

What is Gradient?

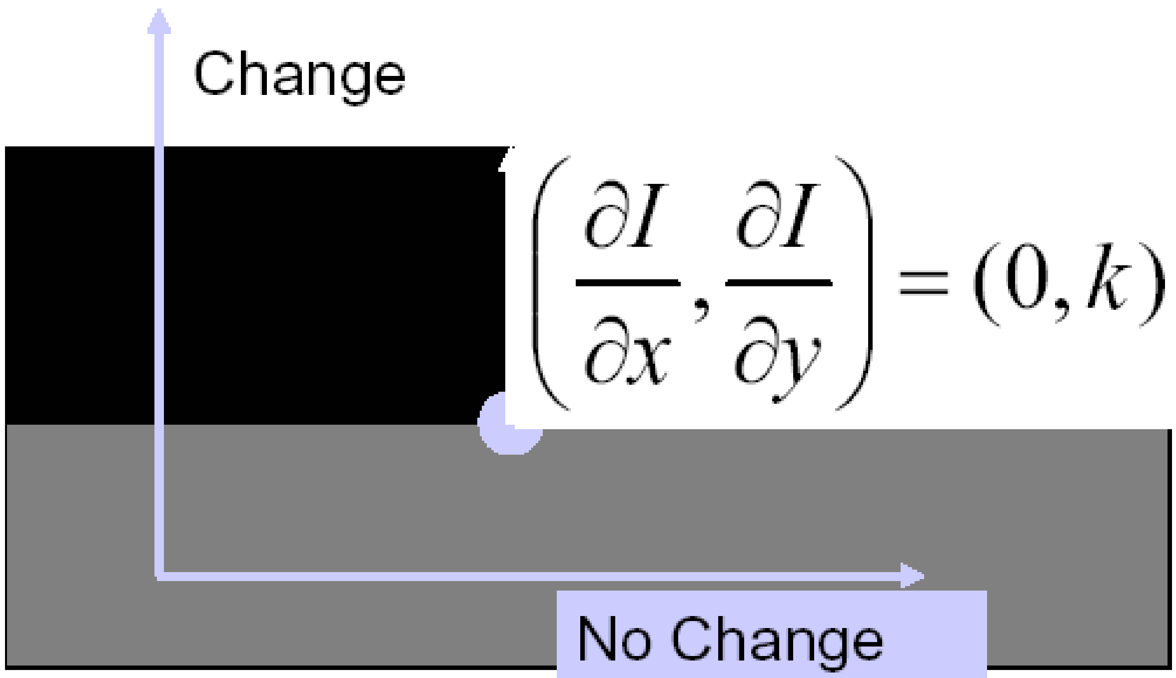


$$\left(\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}\right) = (k, 0)$$

Jacobs

adapted from Michael Black, Brown University

What is Gradient?



$$\left(\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}\right) = (0, k)$$

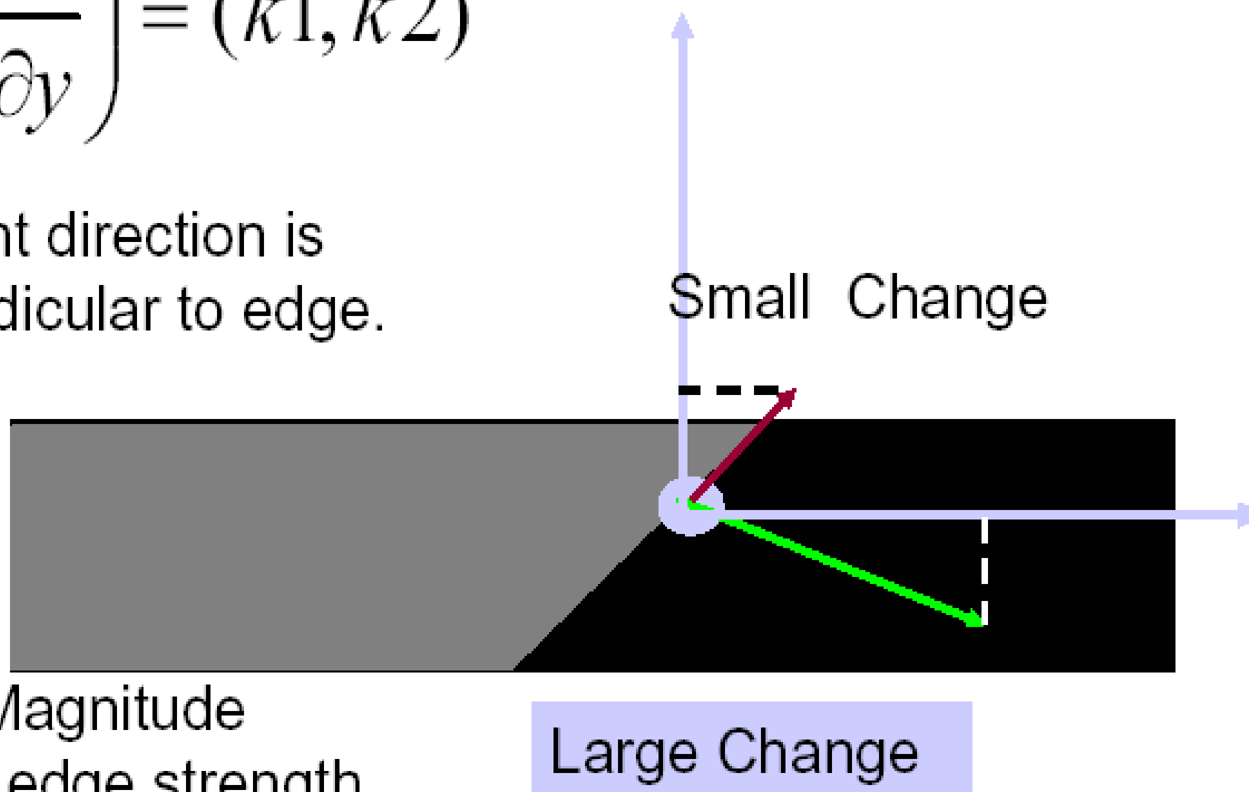
adapted from Michael Black, Brown University

What is Gradient?

$$\left(\frac{\partial I}{\partial x},\frac{\partial I}{\partial y}\right)=(k1,k2)$$

Gradient direction is perpendicular to edge.

Gradient Magnitude measures edge strength.



Jacobs

adapted from Michael Black, Brown University

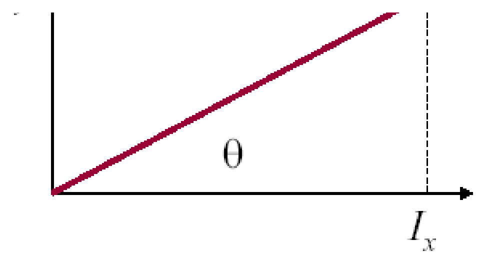
2D edge detection

Take a derivative

- Compute the magnitude of the gradient:

$$\nabla I = (I_x, I_y) = \left(\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y} \right) \text{ is the Gradient}$$

$$\|\nabla I\| = \sqrt{I_x^2 + I_y^2}$$



$$\theta(x, y) = \arctan(I_y(x, y), I_x(x, y))$$

adapted from Michael Black, Brown University

Finite Differences

Partial Derivatives

$$\frac{\partial}{\partial x} I(x,y) = I_x \approx I \otimes D_x, \quad \frac{\partial}{\partial y} I(x,y) = I_y \approx I \otimes D_y$$

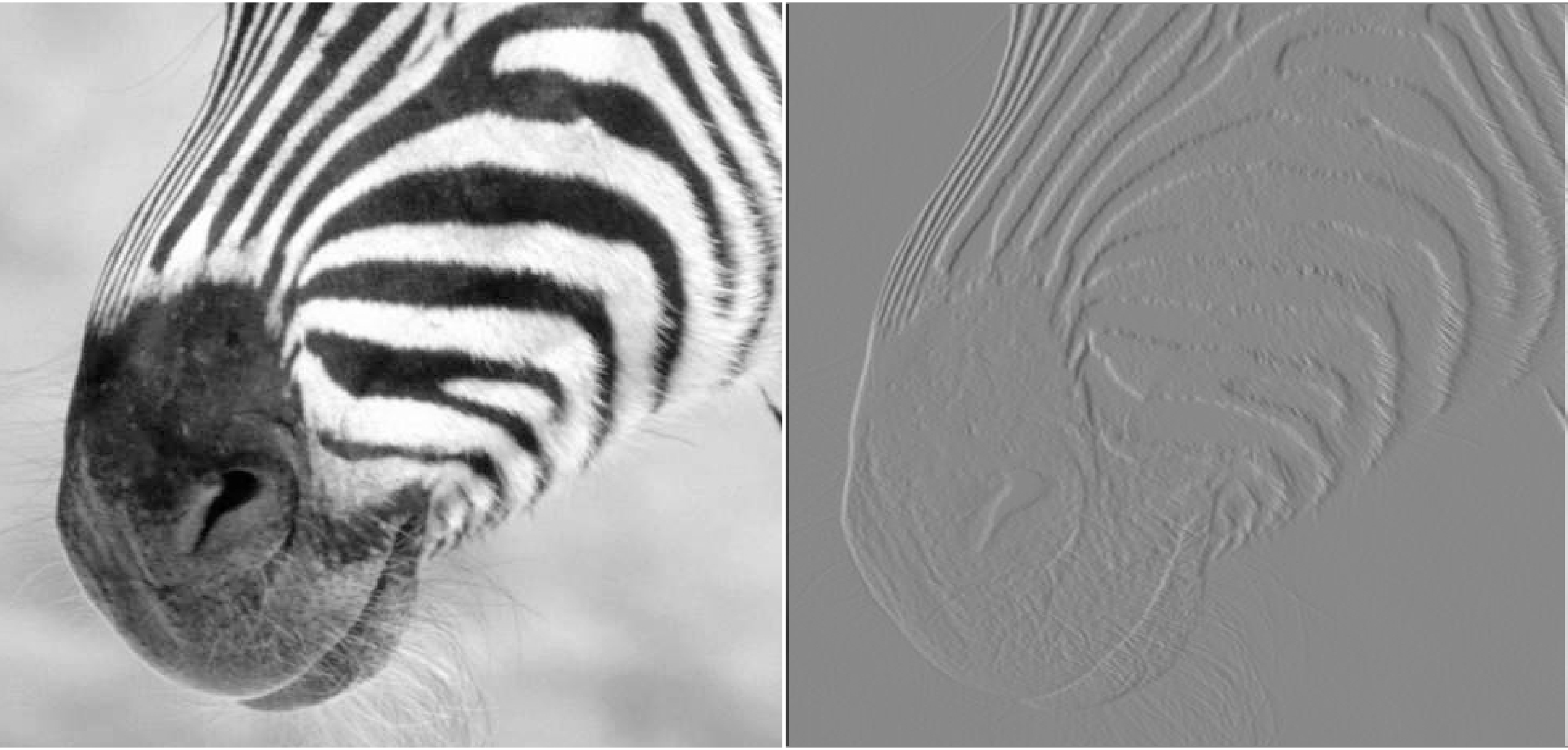
- Often approximated with simple filters:

$$D_x = \frac{1}{3}\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \quad D_y = \frac{1}{3}\begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

Finite differences

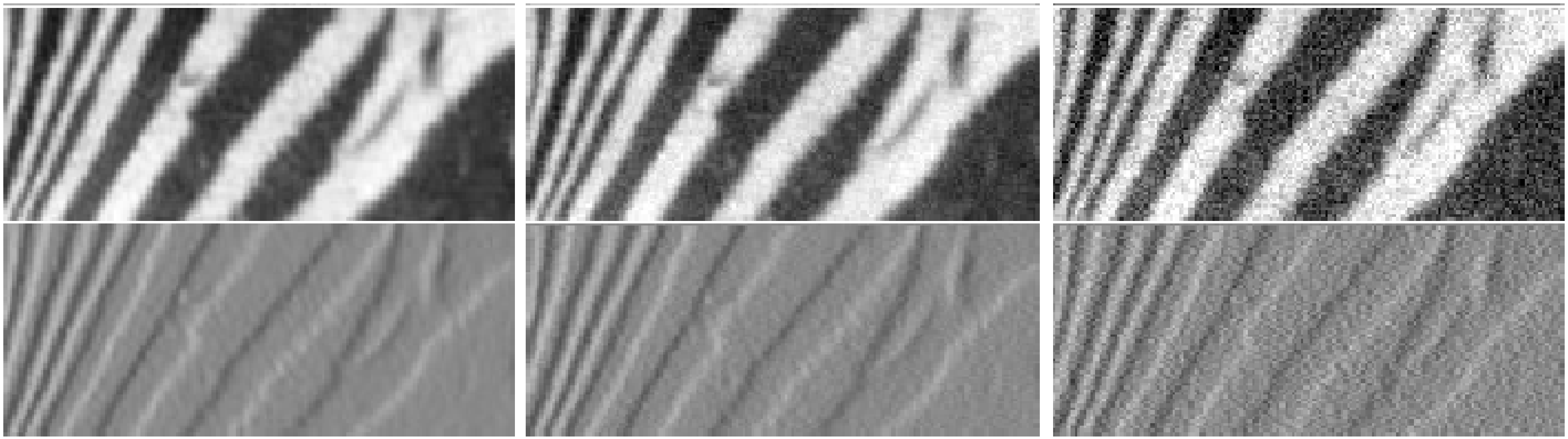
adapted from Michael Black, Brown University

Finite Differences



adapted from David Forsyth, UC Berkeley

Finite Differences responding to noise



- Issue: noise
  - smooth before differentiation

adapted from David Forsyth, UC Berkeley

Smoothing and differentiation

Smooth with a Gaussian

Why Gaussian?

-if we convolve a Gaussian with a Gaussian the result is a Gaussian

$G(\sigma 1)**G(\sigma 2) \rightarrow G(sqrt(\sigma 1*\sigma 1+ \sigma 2*\sigma 2))$

-efficient -> for $\sigma=1$ k should be 5, for $\sigma=10$ k should be 50

-convolving any function with itself repeatedly eventually yields a Gaussian

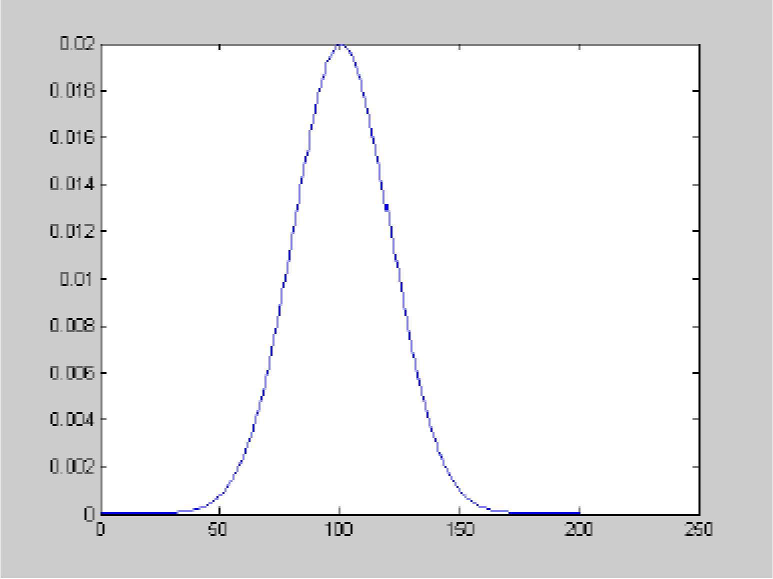
-Gaussian is separable

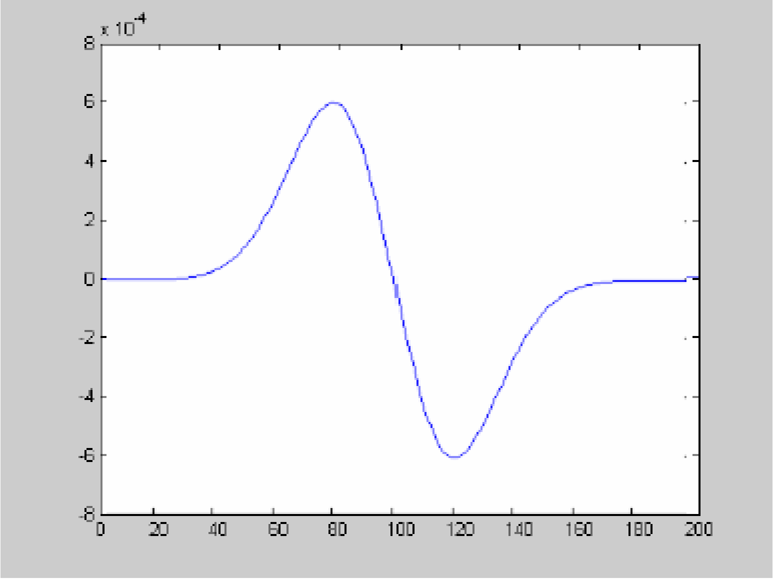
two convolutions to smooth, then differentiate?

– actually, no - we can use a derivative of Gaussian filter
  - because differentiation is convolution, and convolution is associative

Derivatives and Smoothing

$$D_x \otimes (G \otimes I) = (D_x \otimes G) \otimes I$$



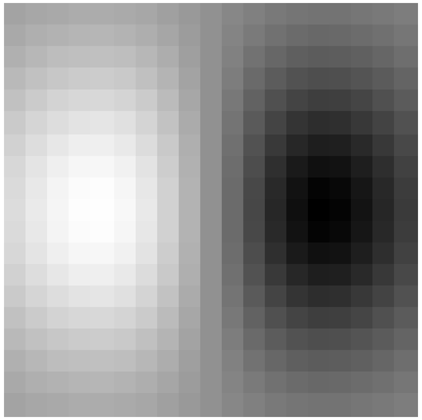


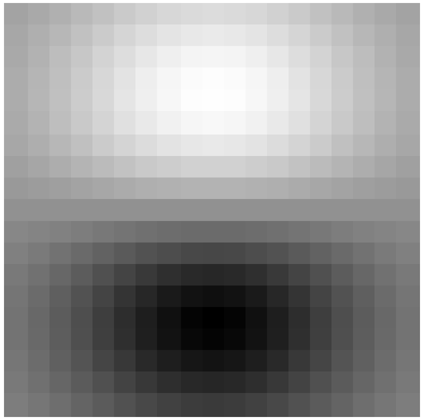
adapted from Michael Black, Brown University

Derivatives and Smoothing

In 2D

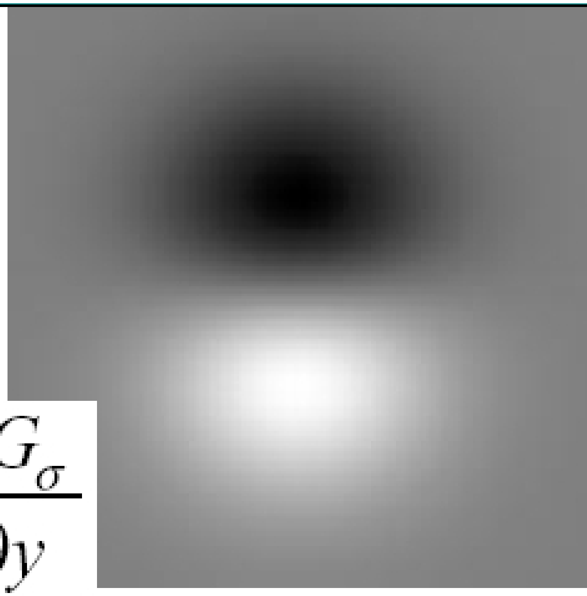
$$D_x \otimes (G \otimes I) = (D_x \otimes G) \otimes I$$



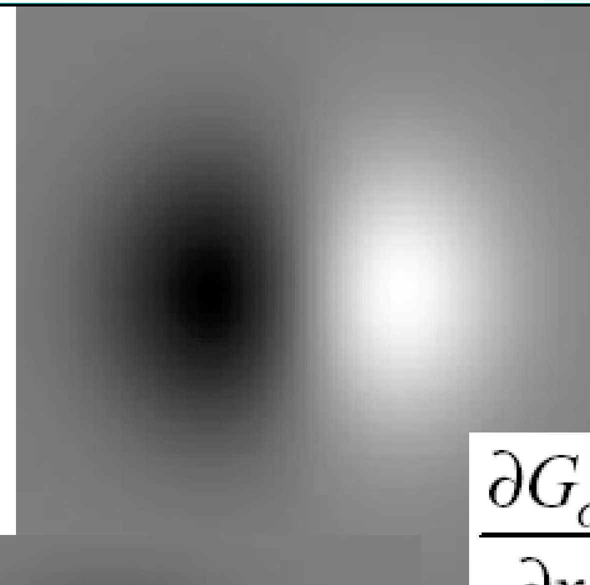


adapted from Michael Black, Brown University

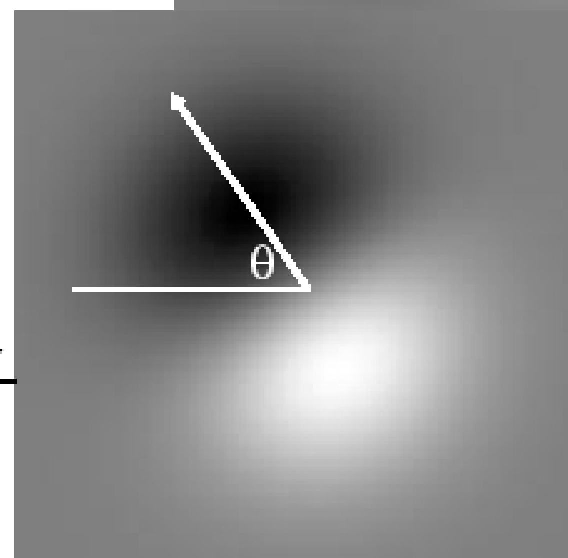
First Derivative of Gaussian



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



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

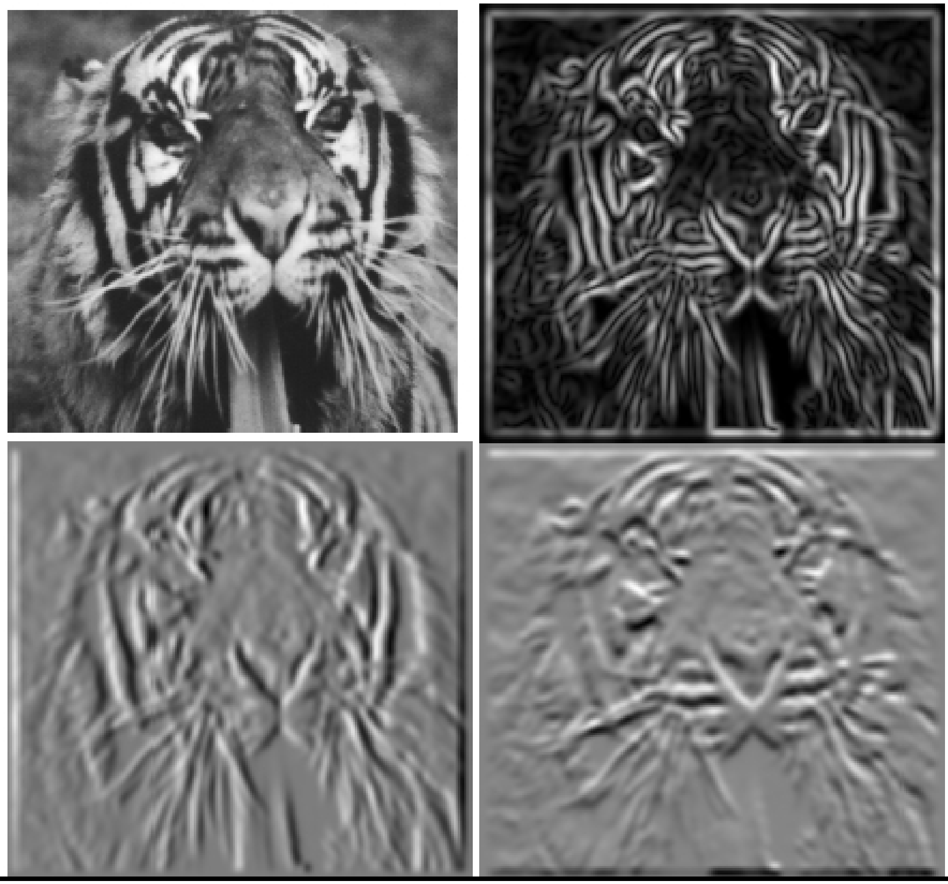


$$\cos\theta \frac{\partial G_\sigma}{\partial x} + \sin\theta \frac{\partial G_\sigma}{\partial y}$$

adapted from Martial Hebert, CMU

First Derivative of Gaussian

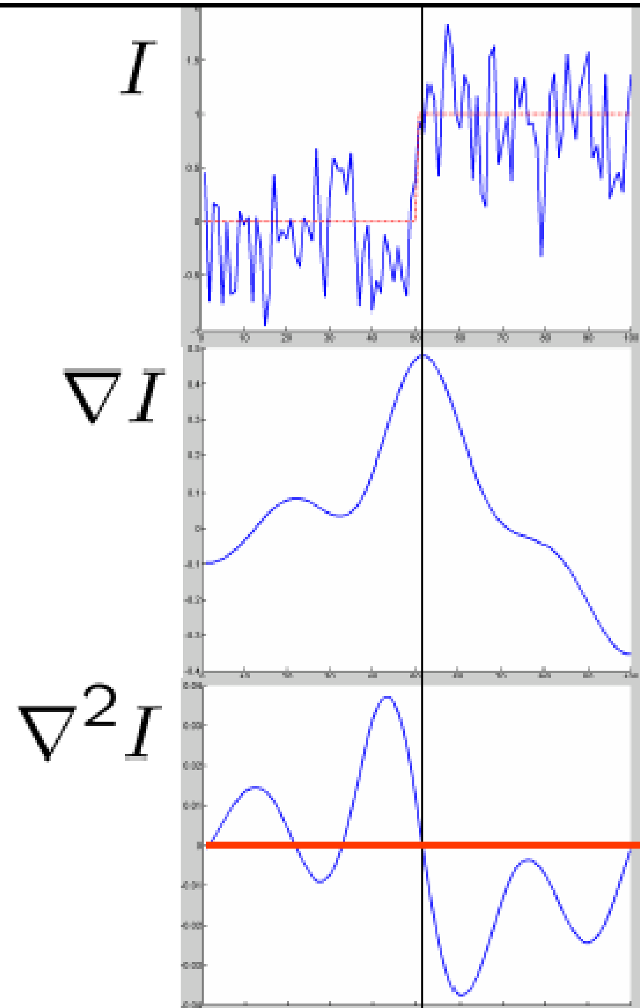
Applying the first derivative of Gaussian



adapted from Martial Hebert, CMU

Second derivative

- Another way to detect an extremal first derivative is to look for a zero second derivative

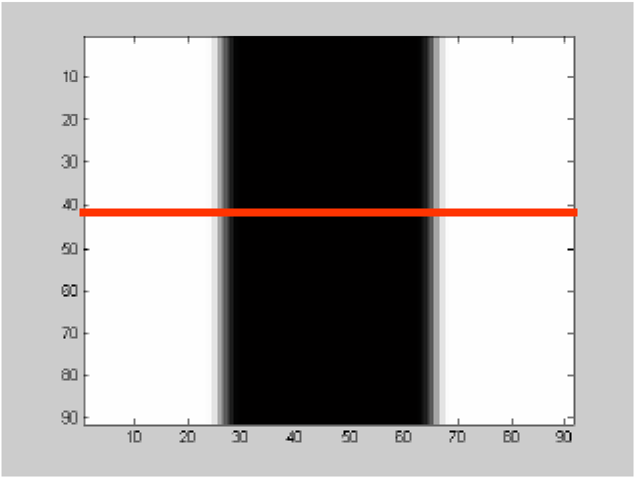


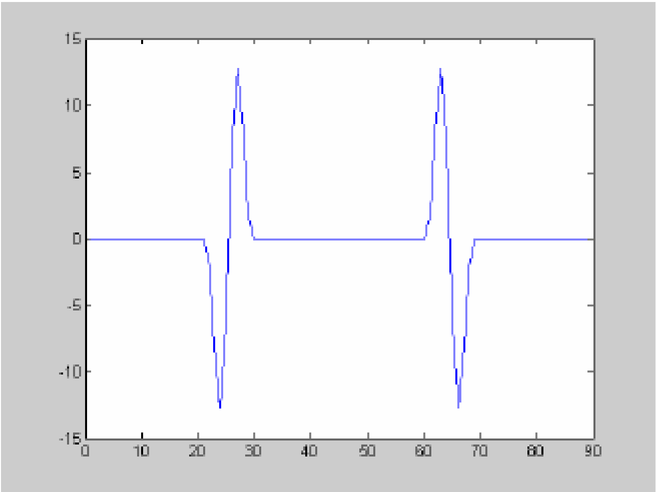
Discontinuities generate maxima of first derivative (gradient magnitude)

Discontinuities generate Zero-crossings of 2[nd] derivative (Laplacian)

adapted from Martial Hebert, CMU

Step edge





Recall: the zero-crossings of the second derivative tell us the location of edges.

adapted from Michael Black, Brown University

Second derivative

$$f''(x) = \lim_{h \to 0} \frac{f'(x+h) - f'(x)}{h} \approx f'(x+1) - f'(x)$$

$$= f(x+2) - 2f(x+1) + f(x)$$

Mask?

[1 -2 1]

adapted from Michael Black, Brown University

Sobel Operator

$$\frac{1}{8}\begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline \end{array} \qquad \frac{1}{8}\begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -2 & -1 \\ \hline \end{array}$$

$$s_x \qquad\qquad s_y$$

Laplacian of Gaussian (LOG)

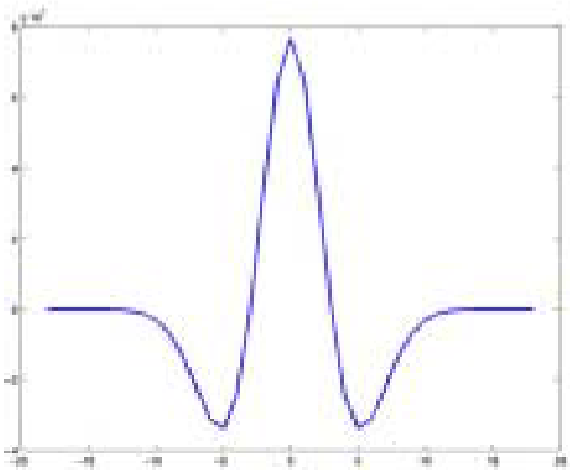
- Bad idea to apply a Laplacian without smoothing
  - smooth with Gaussian, apply Laplacian
  - this is the same as filtering with a Laplacian of Gaussian filter

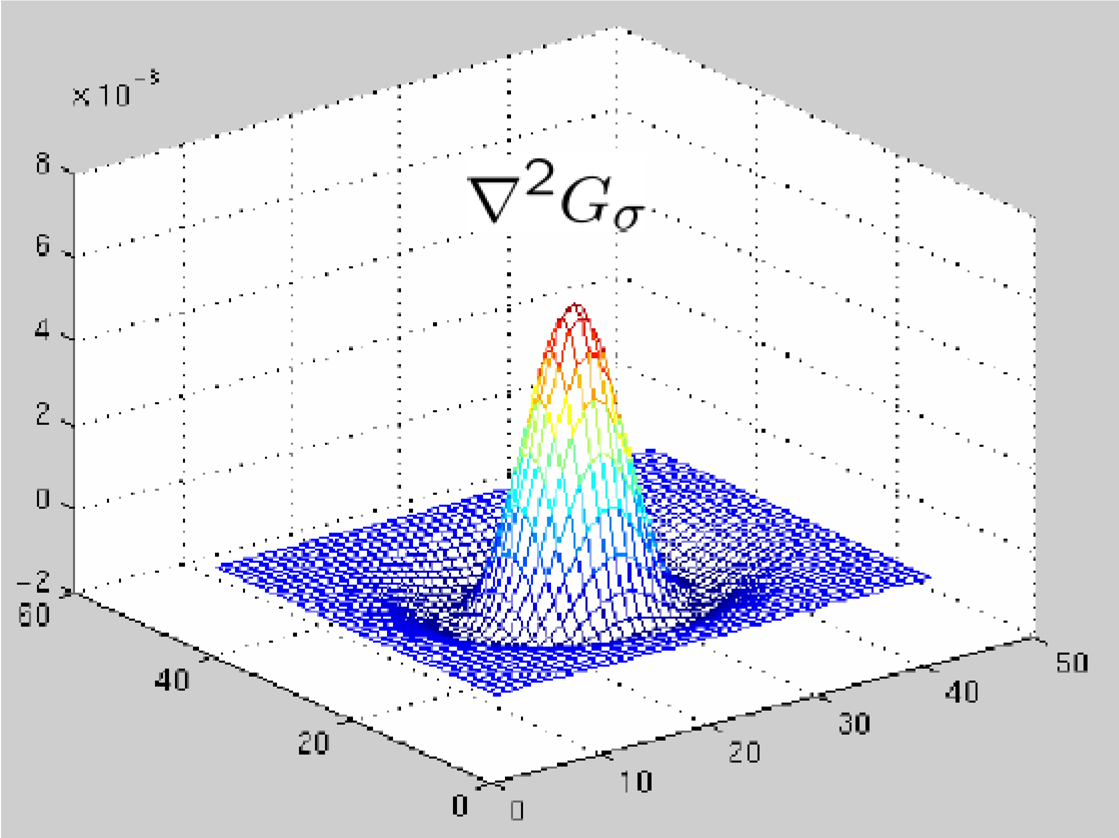
$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

Just another linear filter.

$$\nabla^2 (f(x,y) \otimes G(x,y)) = \nabla^2 G(x,y) \otimes f(x,y)$$

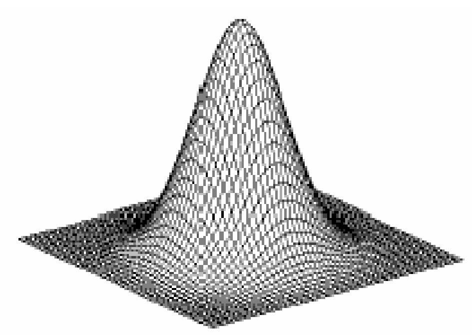
Laplacian of Gaussian (LOG)





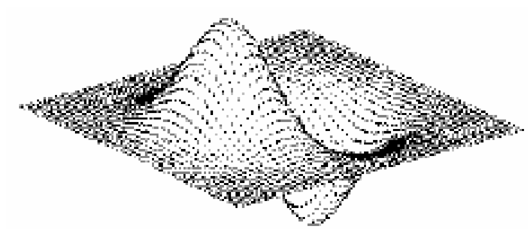
adapted from Martial Hebert, CMU

Filters



Gaussian

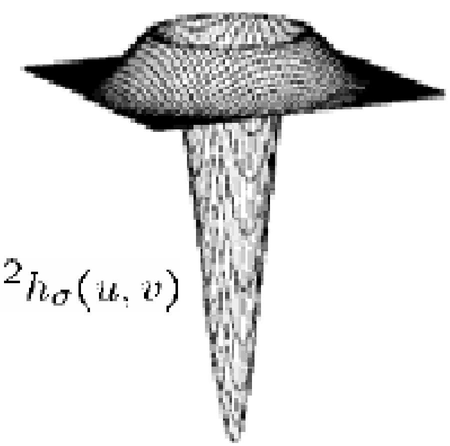
$$h_\sigma(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$



derivative of Gaussian

$$\frac{\partial}{\partial x} h_\sigma(u,v)$$

Laplacian of Gaussian

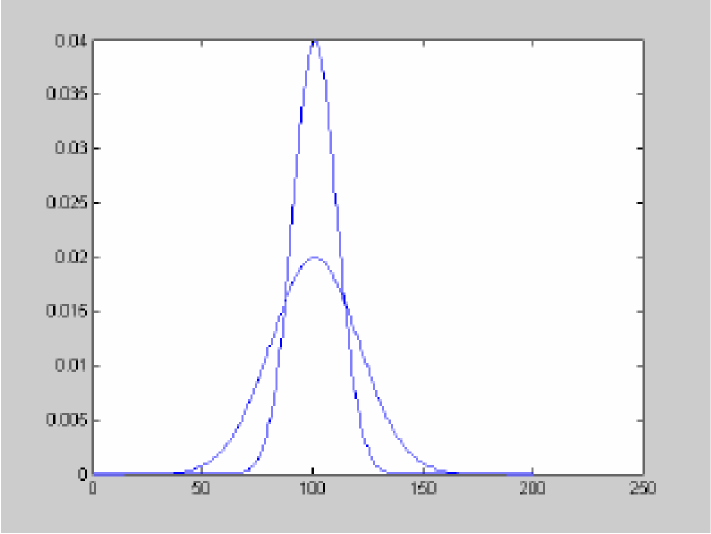


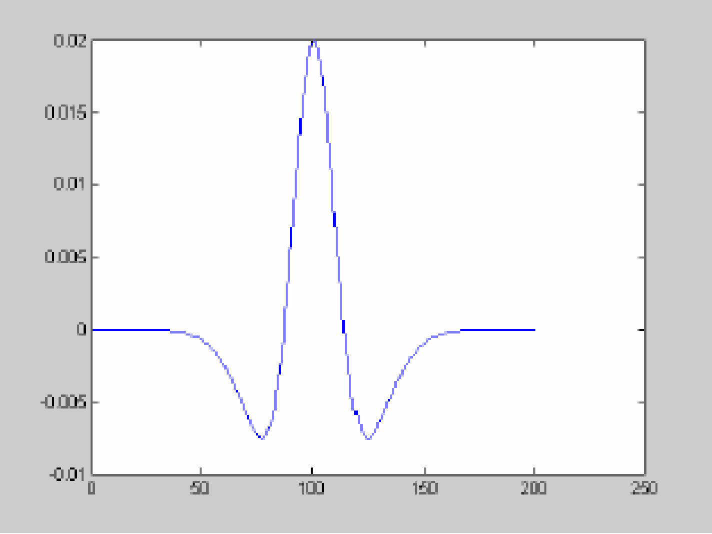
$$\nabla^2 h_\sigma(u,v)$$

Approximating the Laplacian

We can approximate Laplacian of Gaussian by

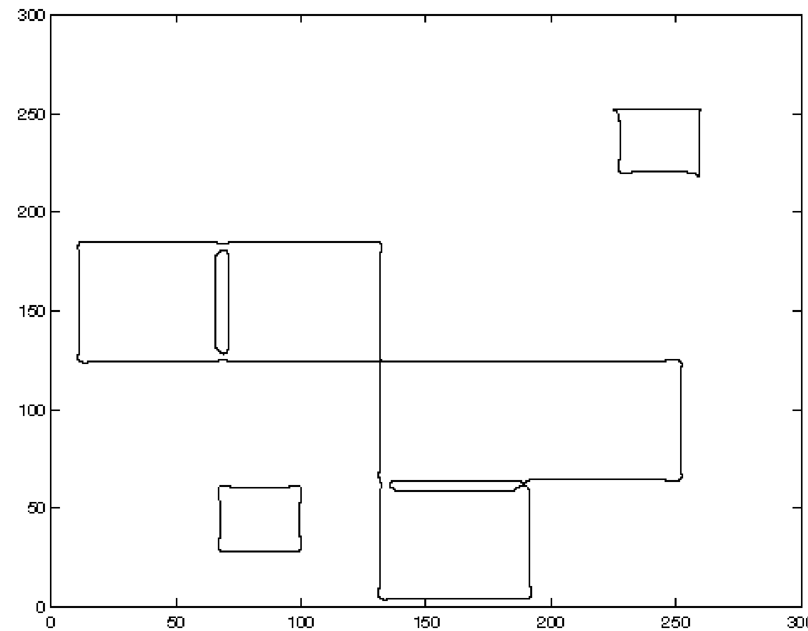
- Difference of Gaussians at different scales.





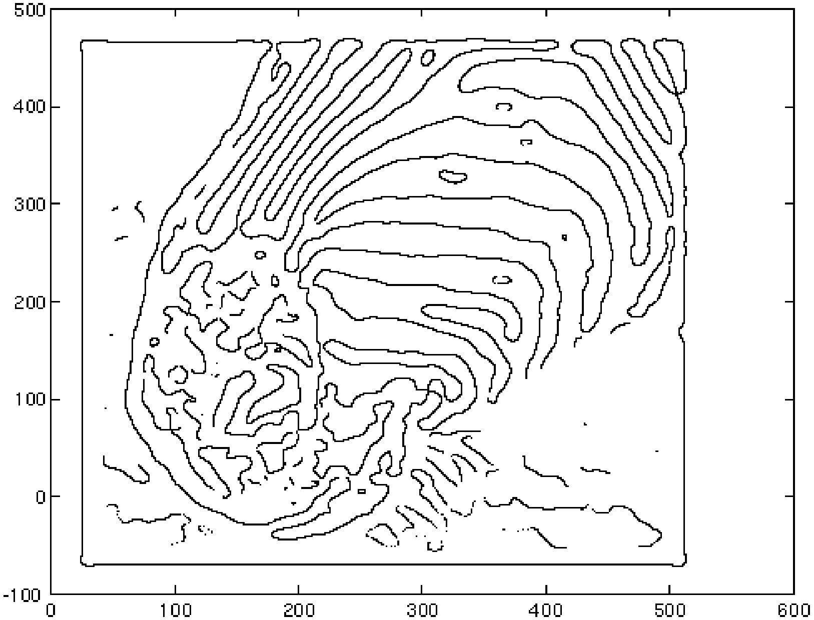
adapted from Michael Black, Brown University

Laplacian of Gaussian (LOG)

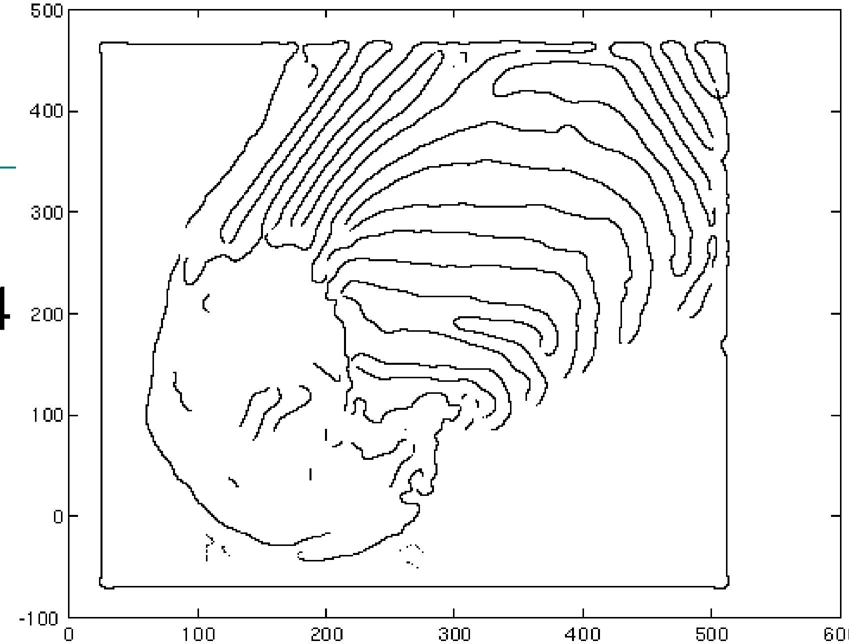


At the corners where three or more edges meet, contours behave strangely

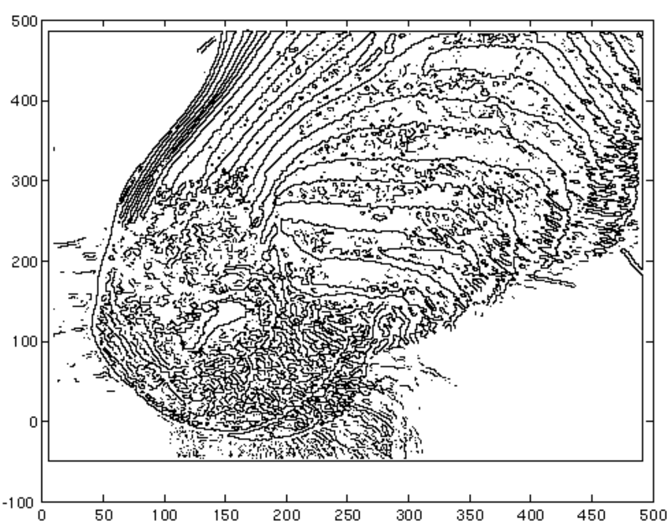
adapted from David Forsyth, UC Berkeley



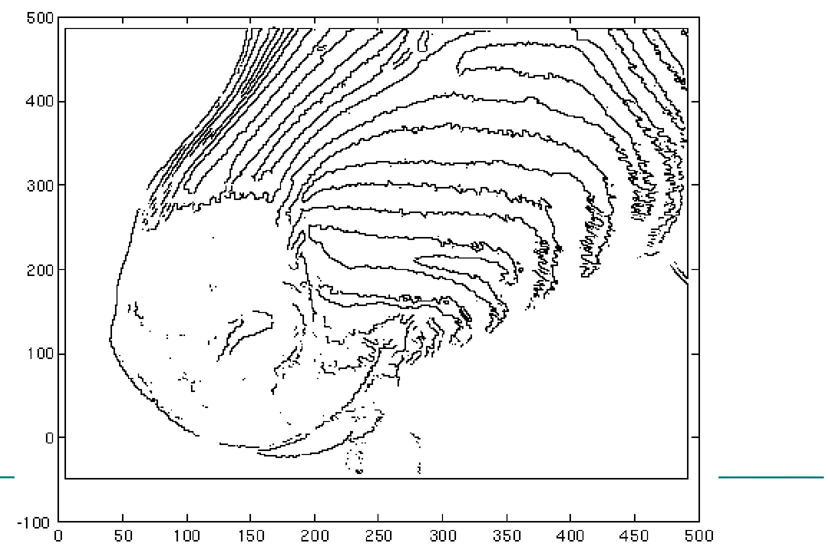
sigma=4



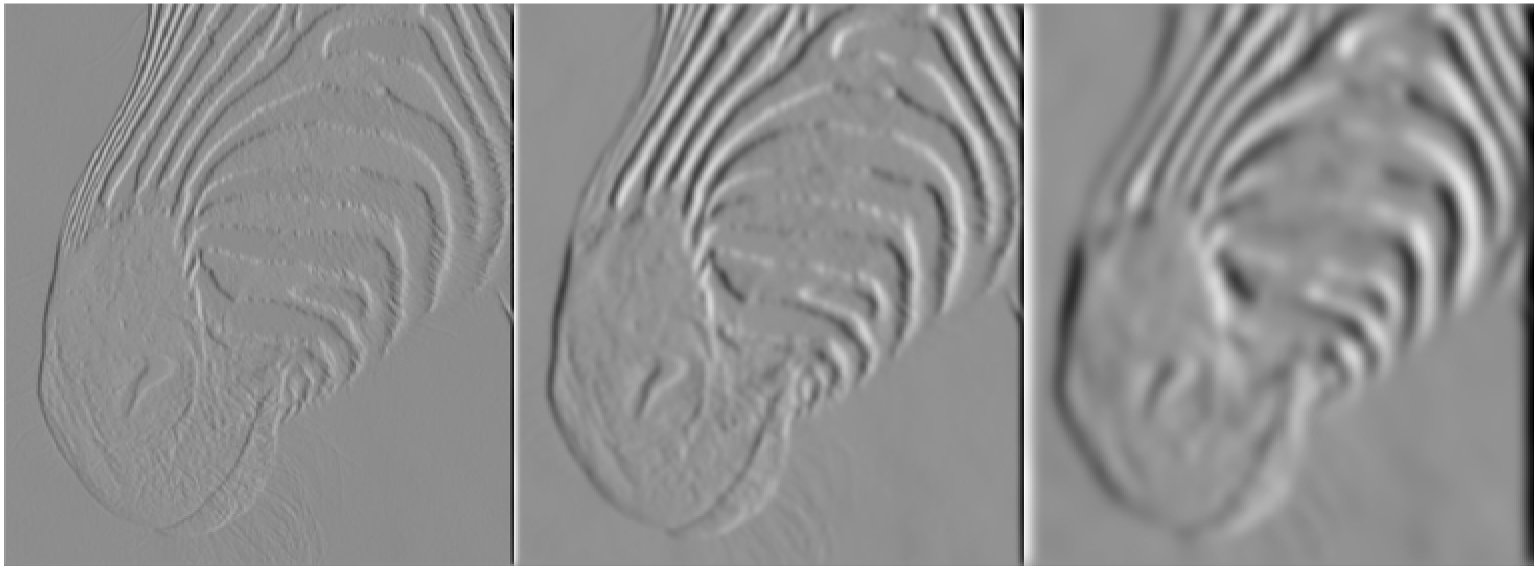
Gradient threshold=1 LOG zero crossings Gradient threshold=4



sigma=2



Gradient based algorithms



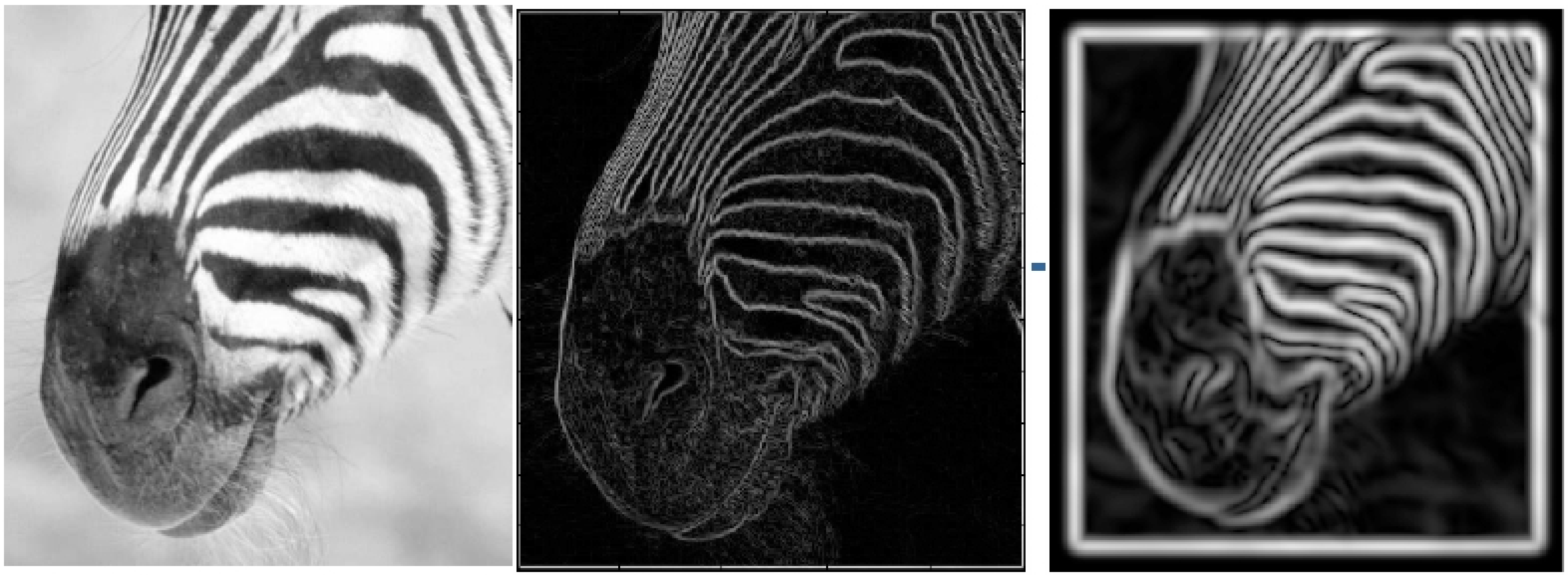
1 pixel 3 pixels 7 pixels

The scale of the smoothing filter affects derivative estimates, and also the semantics of the edges recovered.

Note: strong edges persist across scales.

adapted from David Forsyth, UC Berkeley

Gradient based algorithms



There are three major issues:

1) The gradient magnitude at different scales is different; which 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?

adapted from David Forsyth, UC Berkeley

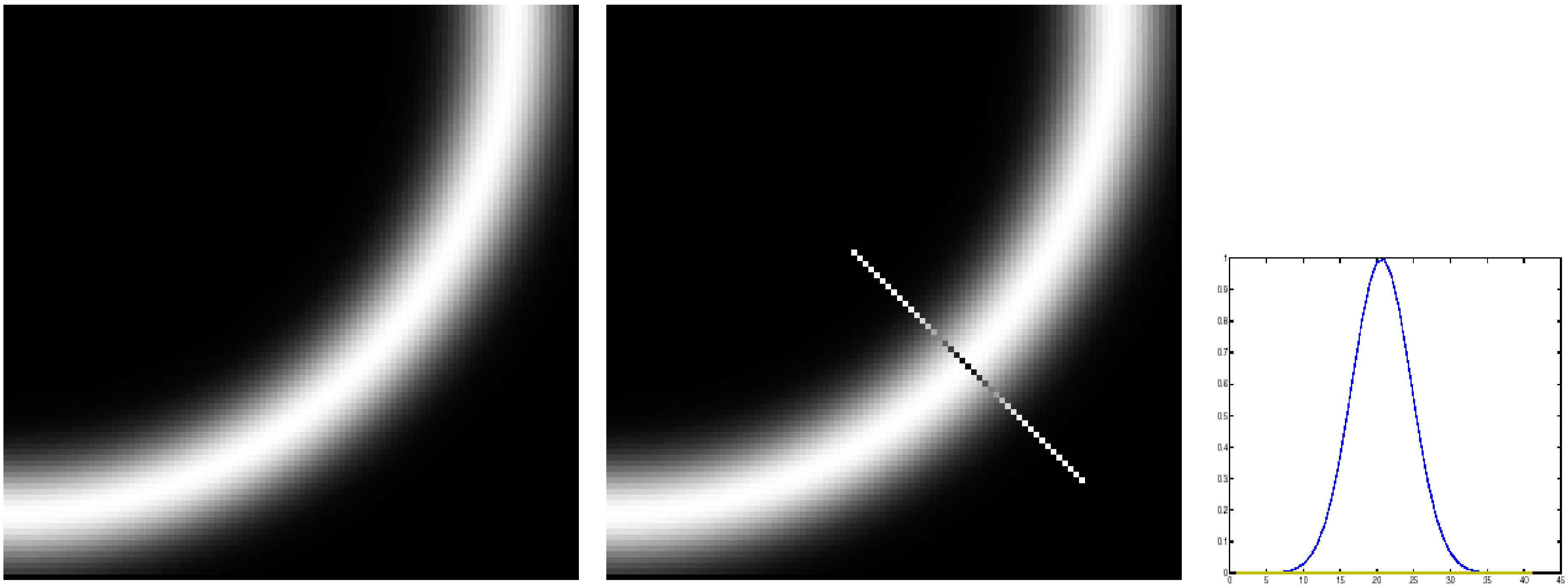
Gradient based edge detection

Gradient magnitude can be large along a thick trail in an image of edge points
We need to obtain most distinctive points on this trail

Look for the points where gradient magnitude is a maximum along the direction perpendicular to the edge

Algorithm :
Form an estimate of the image gradient
Obtain the gradient magnitude from this estimate
Identify image points where
 the value of the gradient magnitude is maximal
 in the direction perpendicular to the edge
 and also large
as edge points

Gradient based edge detection



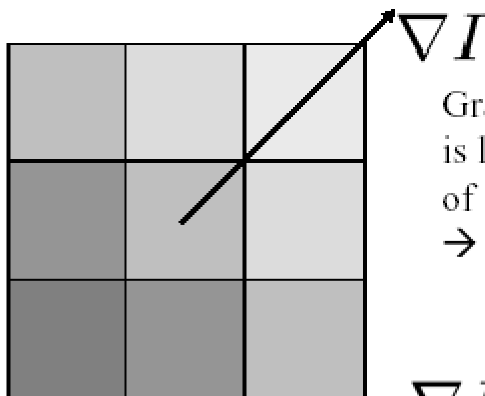
We wish to mark points along the curve where the magnitude is biggest. We can do this by looking for a maximum along a slice normal to the curve (non-maximum suppression). These points should form a curve. There are then two algorithmic issues: at which point is the maximum, and where is the next one?

adapted from David Forsyth, UC Berkeley

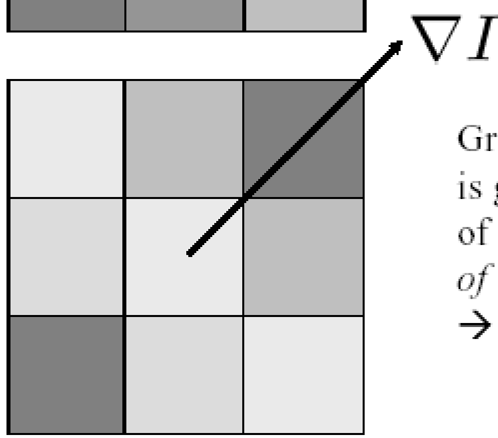
Non-Maximum Suppression Algorithm

```
While there are points with high gradient
that have not been visited

  Find a start point that is a local maximum in the
    direction perpendicular to the gradient
    erasing points that have been checked

  while possible, expand a chain through
    the current point by:
      1) predicting a set of next points, using
        the direction perpendicular to the gradient

      2) finding which (if any) is a local maximum
        in the gradient direction

      3) testing if the gradient magnitude at the
        maximum is sufficiently large

      4) leaving a record that the point and
        neighbours have been visited

      record the next point, which becomes the current point

  end

end
```

Non-Maximum Suppression Algorithm



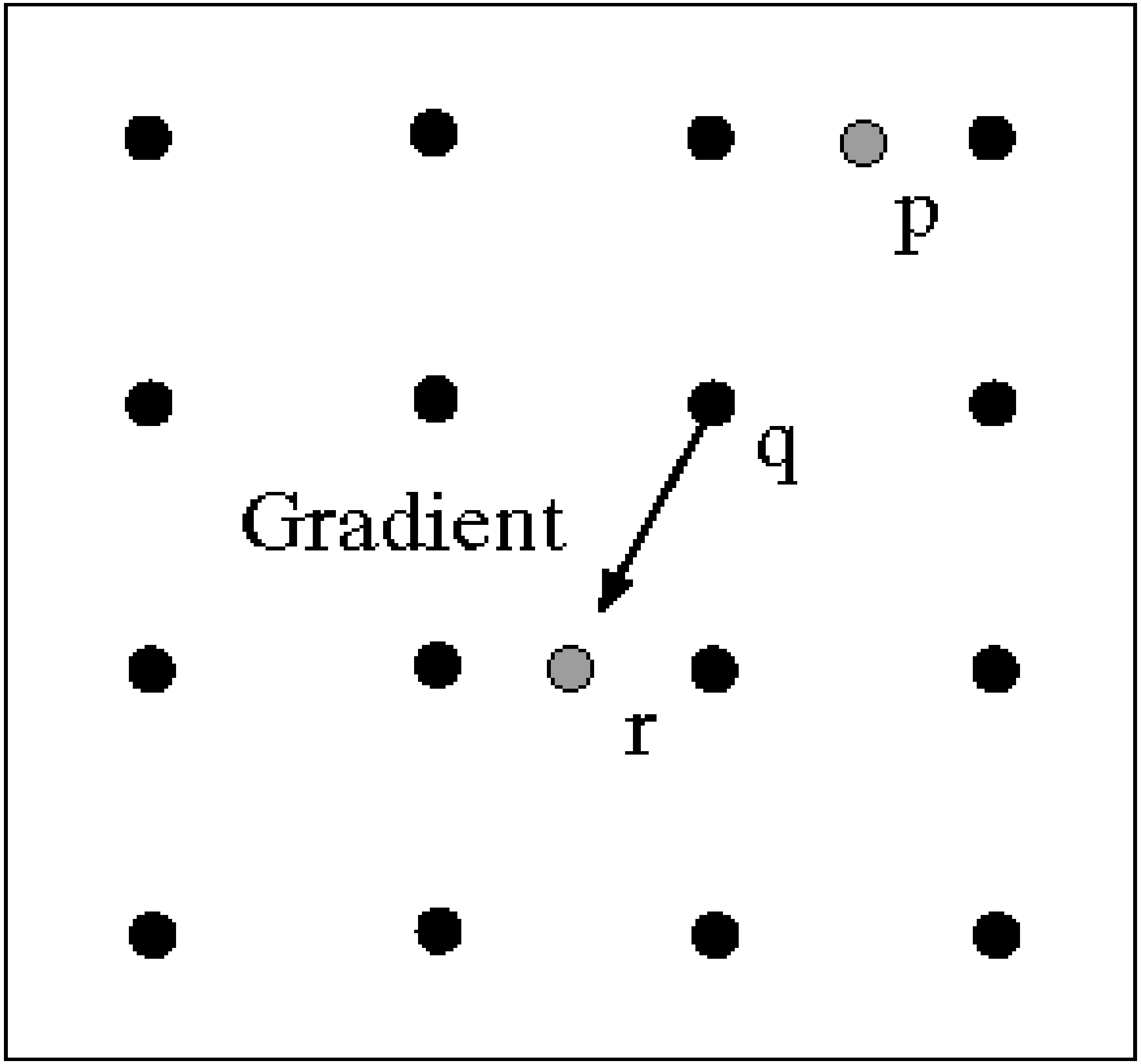
Gradient magnitude at center pixel
is lower than the gradient magnitude
of a neighbor *in the direction of the gradient*
→ Discard center pixel (set magnitude to 0)



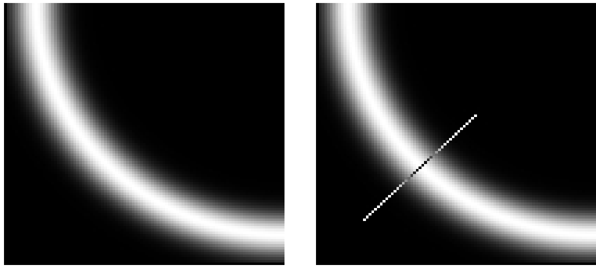
Gradient magnitude at center pixel
is greater than gradient magnitude
of all the neighbors *in the direction of the gradient*
→ Keep center pixel unchanged

adapted from Martial Hebert, CMU

Non-Maximum Suppression

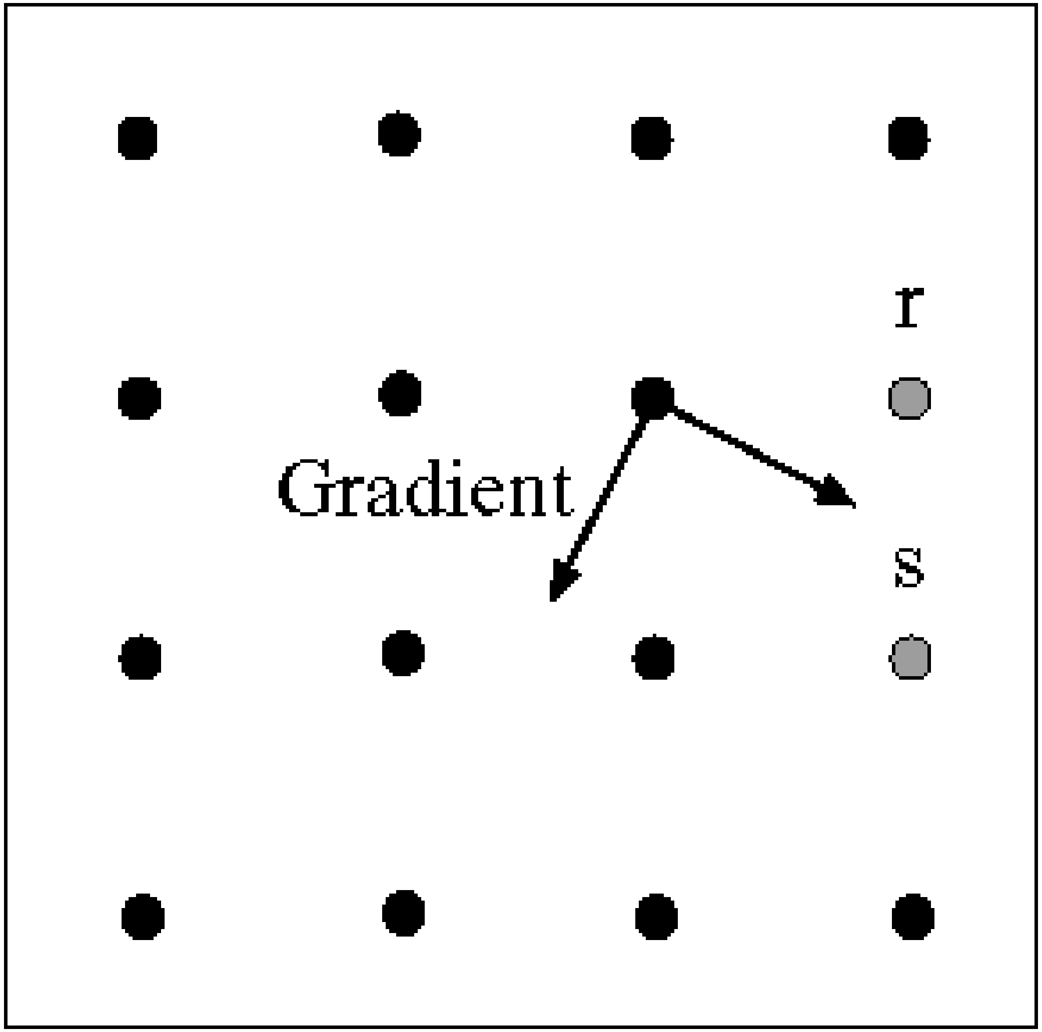


At q, we have a maximum if the value is larger than those at both p and at r. Interpolate to get these values.



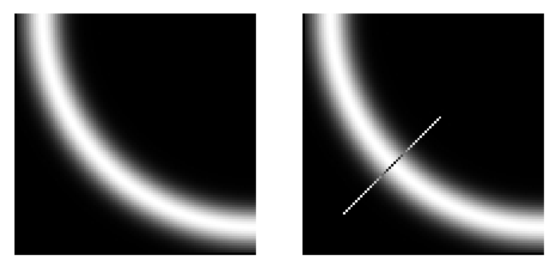
adapted from David Forsyth, UC Berkeley

Non-Maxima Suppression



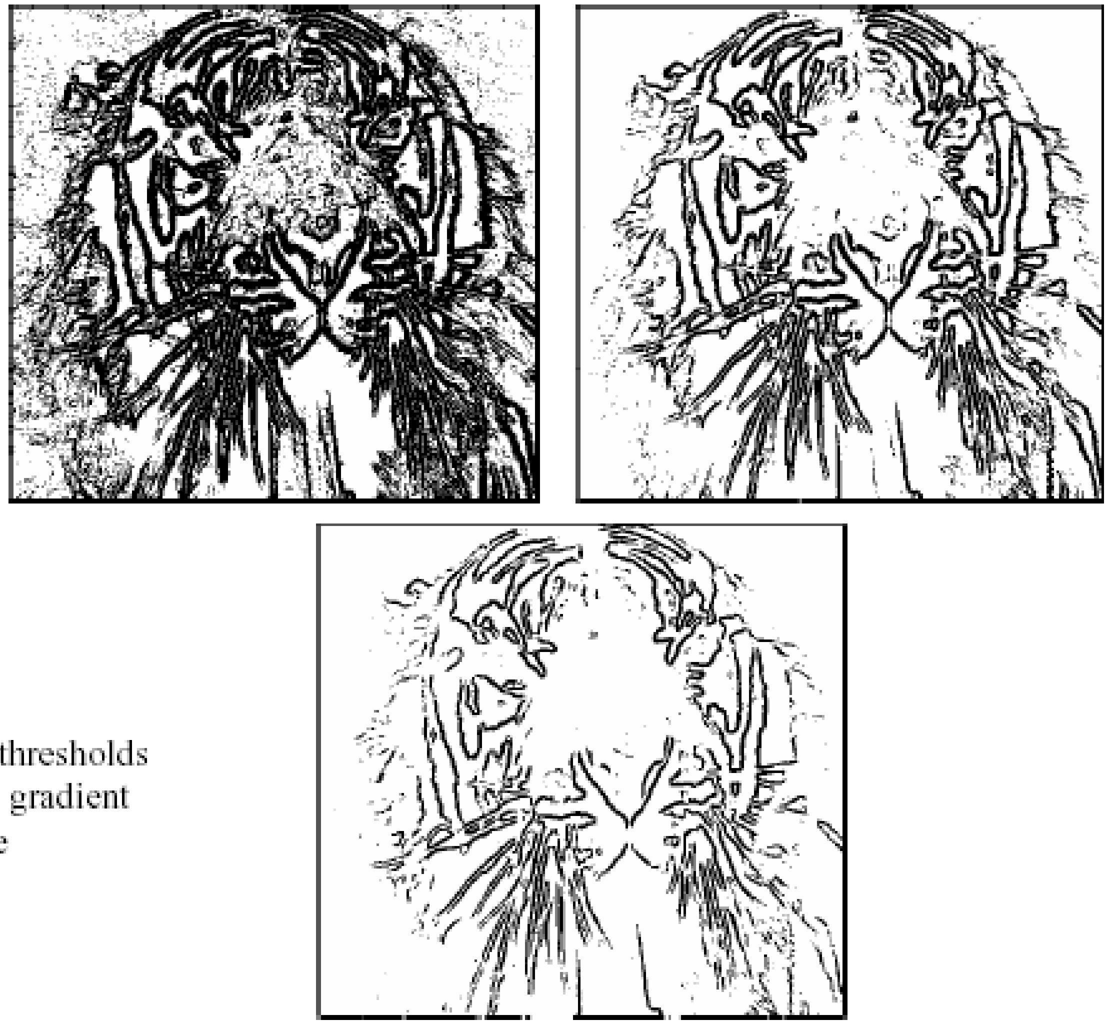
Predicting the next edge point

Assume the marked point 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 (here either r or s).



adapted from David Forsyth, UC Berkeley

Thresholding



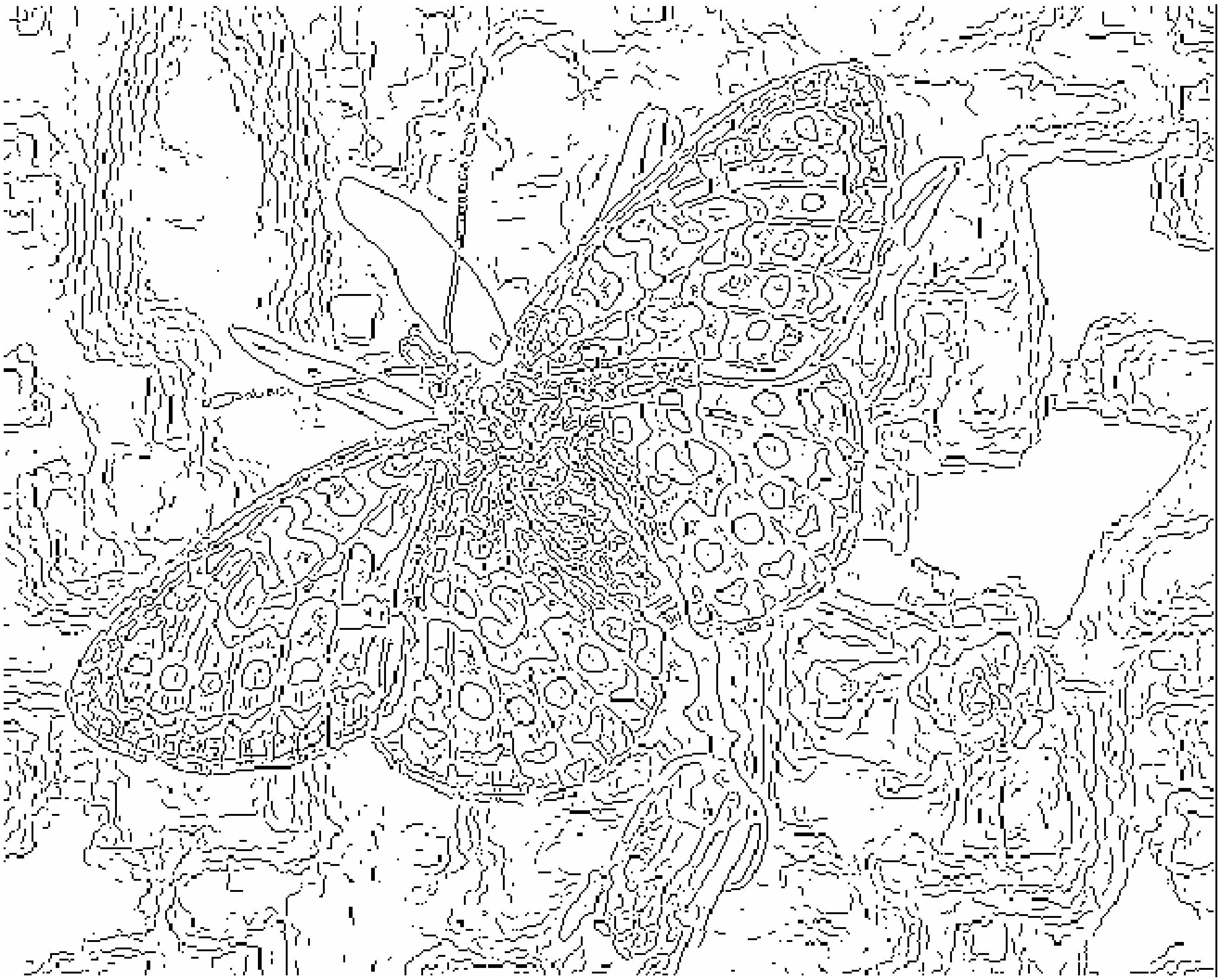
Different thresholds applied to gradient magnitude

adapted from Martial Hebert, CMU

Example

adapted from David Forsyth, UC Berkeley

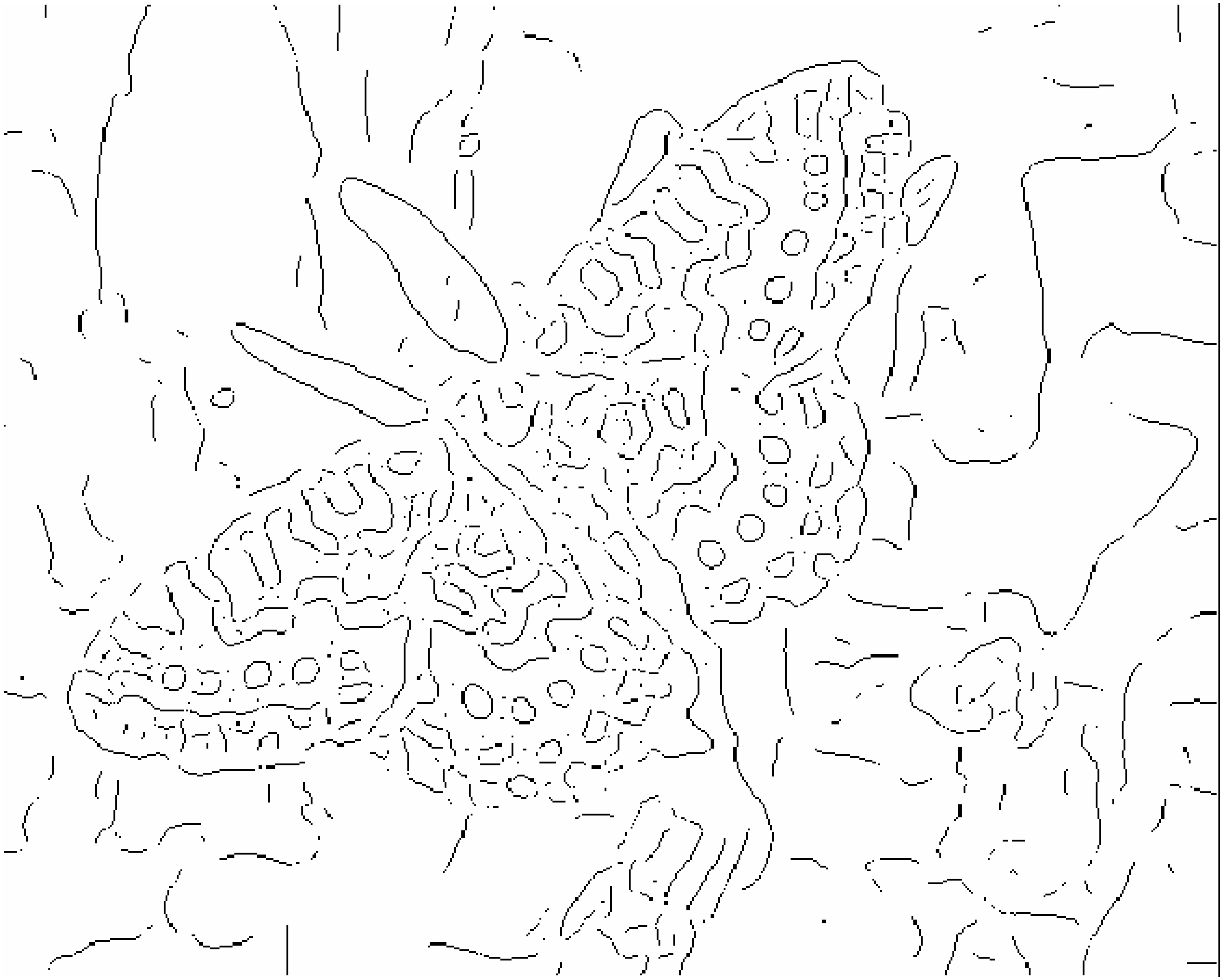
Example



fine scale
high
threshold

adapted from David Forsyth, UC Berkeley

Example



coarse scale, high threshold

adapted from David Forsyth, UC Berkeley

Example

coarse
scale
low
threshold

adapted from David Forsyth, UC Berkeley

Hysteresis Thresholding

.

•Idea : use a high threshold to start edge curves and a low threshold to continue them

•Start with a high threshold (strong edge)

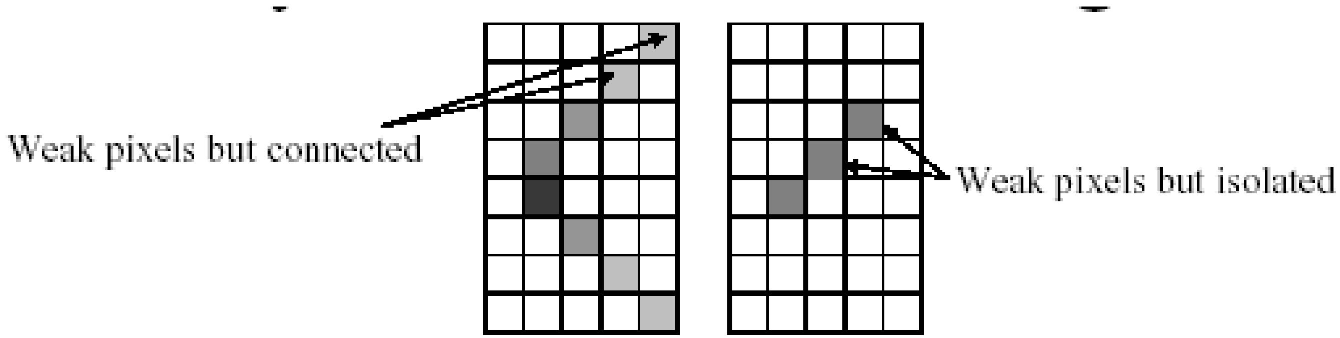
•Apply another threshold which is lower than the higher one

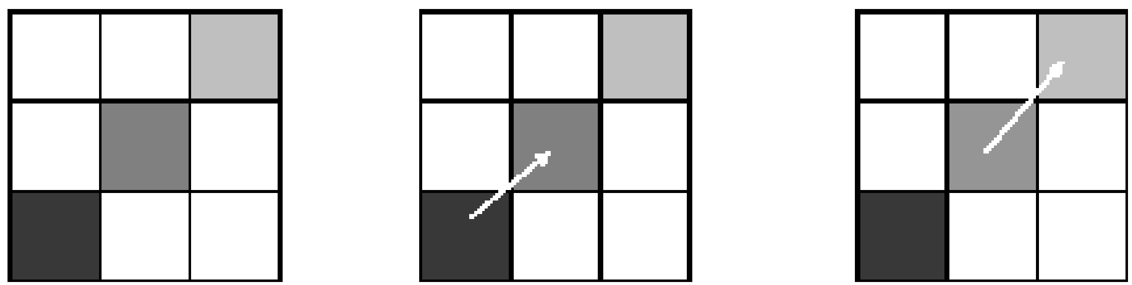
•Look for the other edges which are higher than the lower threshold and connected to the strong edges

•Add them to the edge list

adapted from Martial Hebert, CMU

Hysteresis Thresholding





Very strong edge response. Let's start here

Weaker response but it is connected to a confirmed edge point. Let's keep it.

Continue….

Note: Darker squares illustrate stronger edge response (larger M)

adapted from Martial Hebert, CMU

Complete Algorithm

1. Compute x and y derivatives of image

$$I_x = G_\sigma^x * I \quad I_y = G_\sigma^y * I$$

2. Compute magnitude of gradient at every pixel

$$M(x,y) = |\nabla I| = \sqrt{I_x^2 + I_y^2}$$

3. Eliminate those pixels that are not local maxima of the magnitude in the direction of the gradient

4. Hysteresis Thresholding

   - Select the pixels such that $M > T_h$ (high threshold)

   - Collect the pixels such that $M > T_l$ (low threshold) that are neighbors of already collected edge points

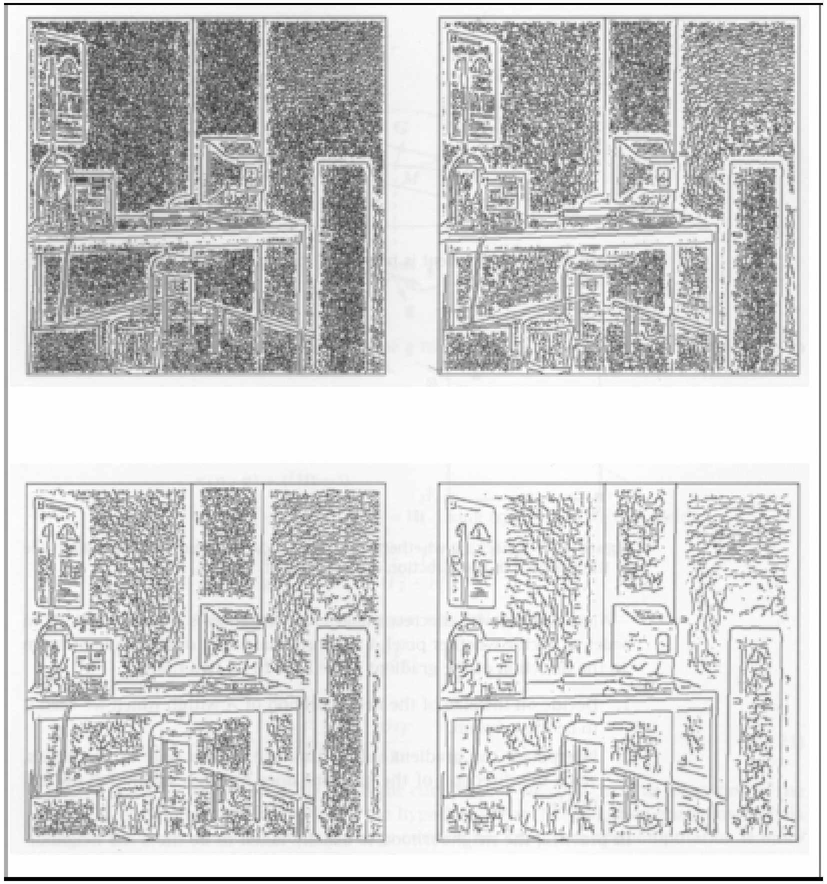
adapted from Martial Hebert, CMU

Example

Input image

adapted from Martial Hebert, CMU

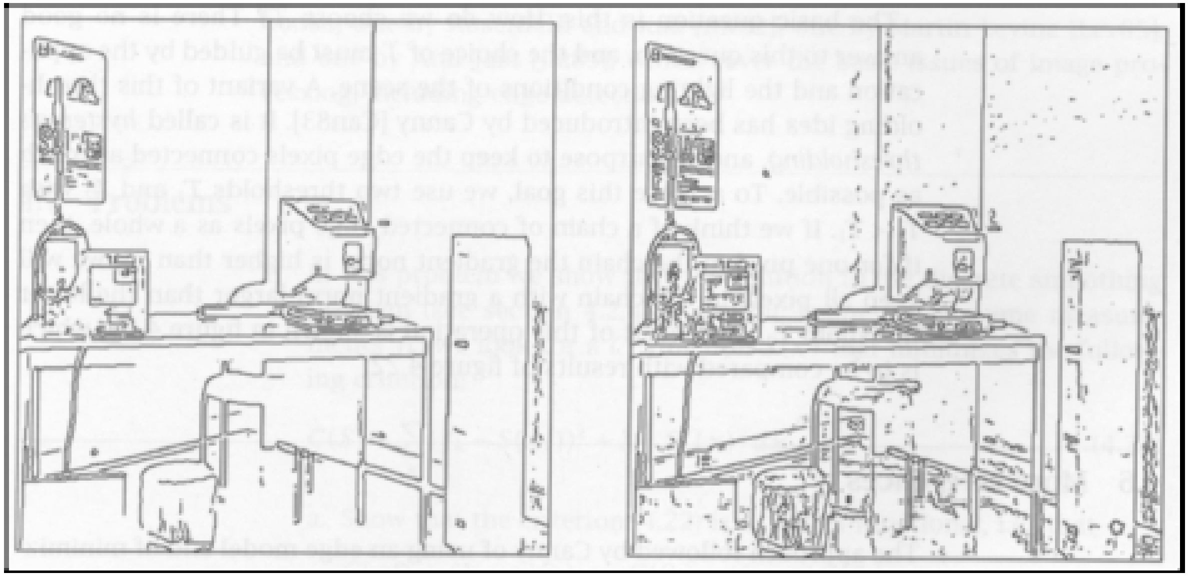
Thresholding



Different thresholds
Applied to gradient value

adapted from Martial Hebert, CMU

Non-local Maxima Suppression



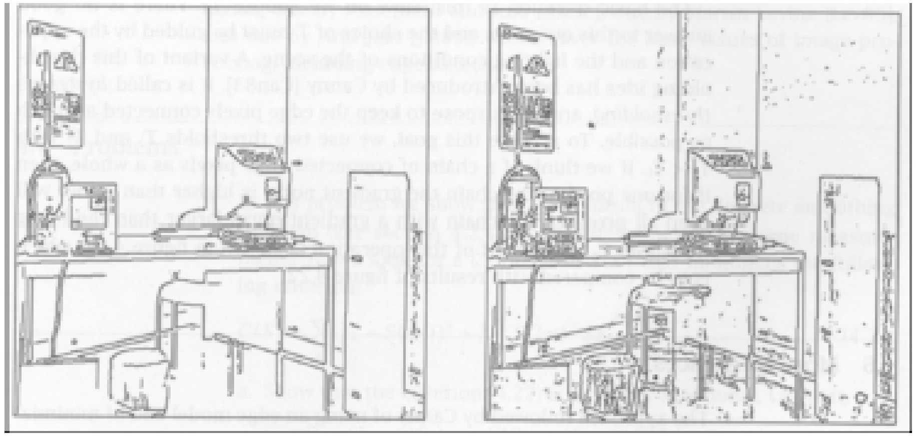
T = 15 T = 5

Two threshold applied to gradient magnitude

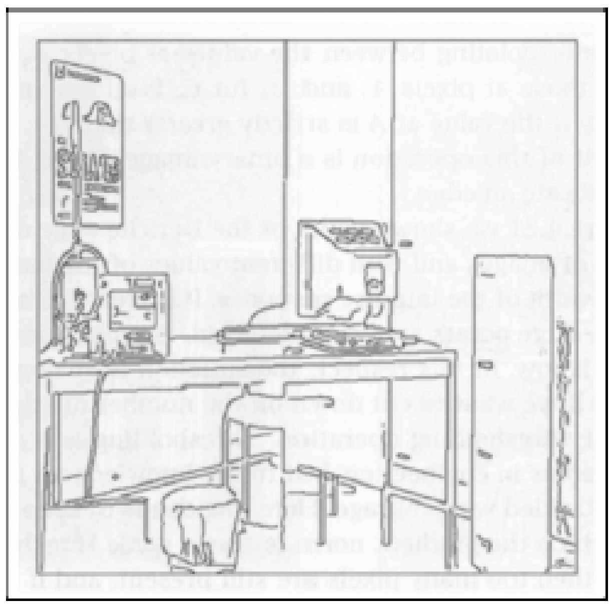
adapted from Martial Hebert, CMU

Hysteresis Thresholding

T=15



T=5

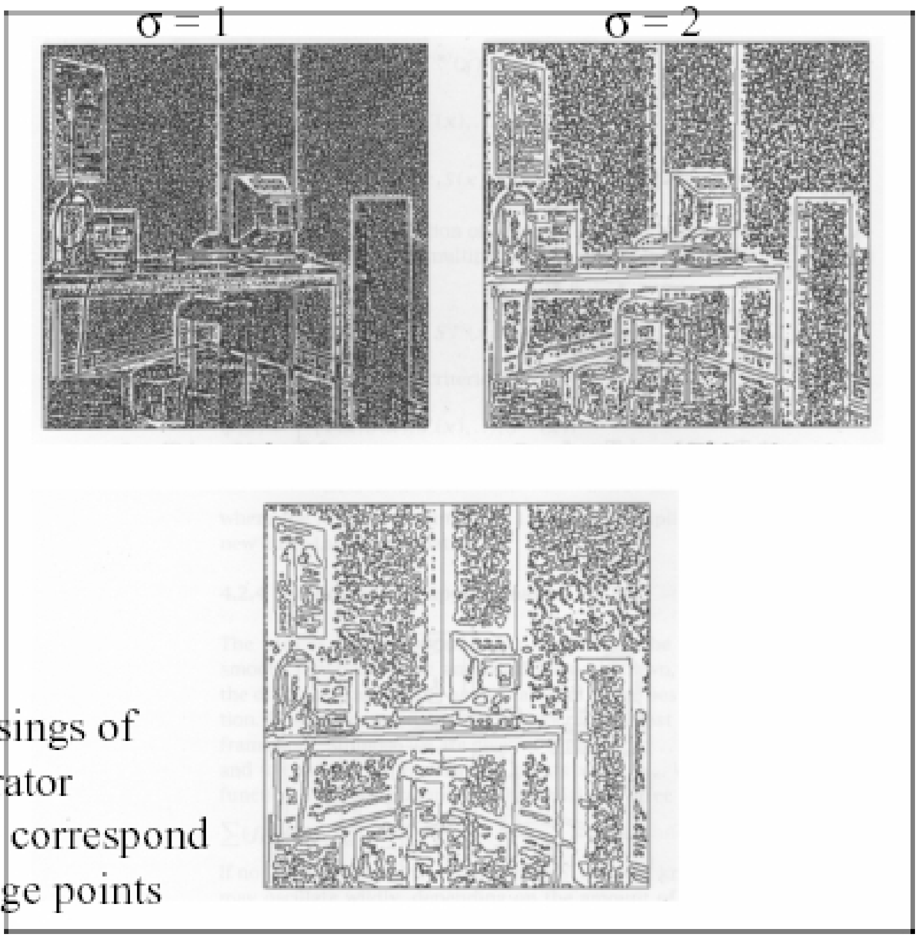


Hysteresis
T_h=15 T_l = 5

Hysteresis
thresholding

adapted from Martial Hebert, CMU

LOG Operator



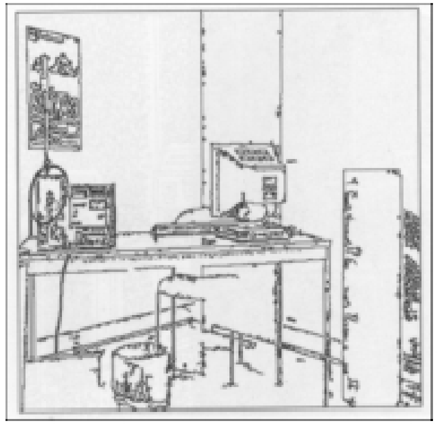
Zero-crossings of LOG operator
→ Do not correspond only to edge points

σ = 3

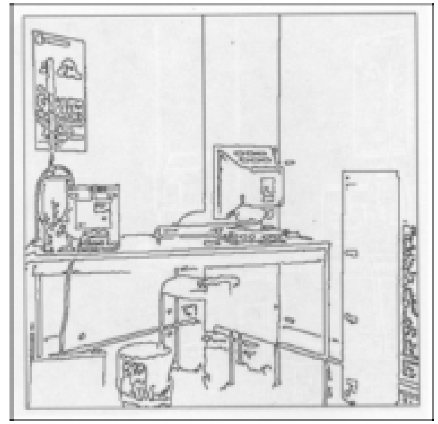
adapted from Martial Hebert, CMU

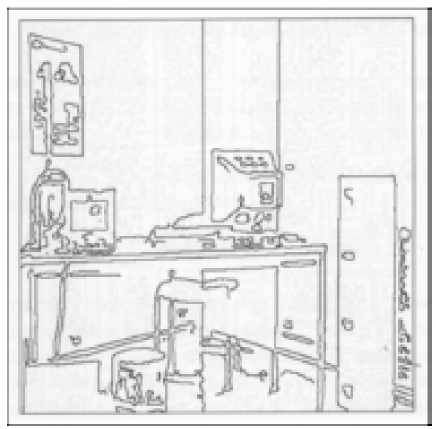
LOG Operator

$\sigma = 1$



$\sigma = 2$



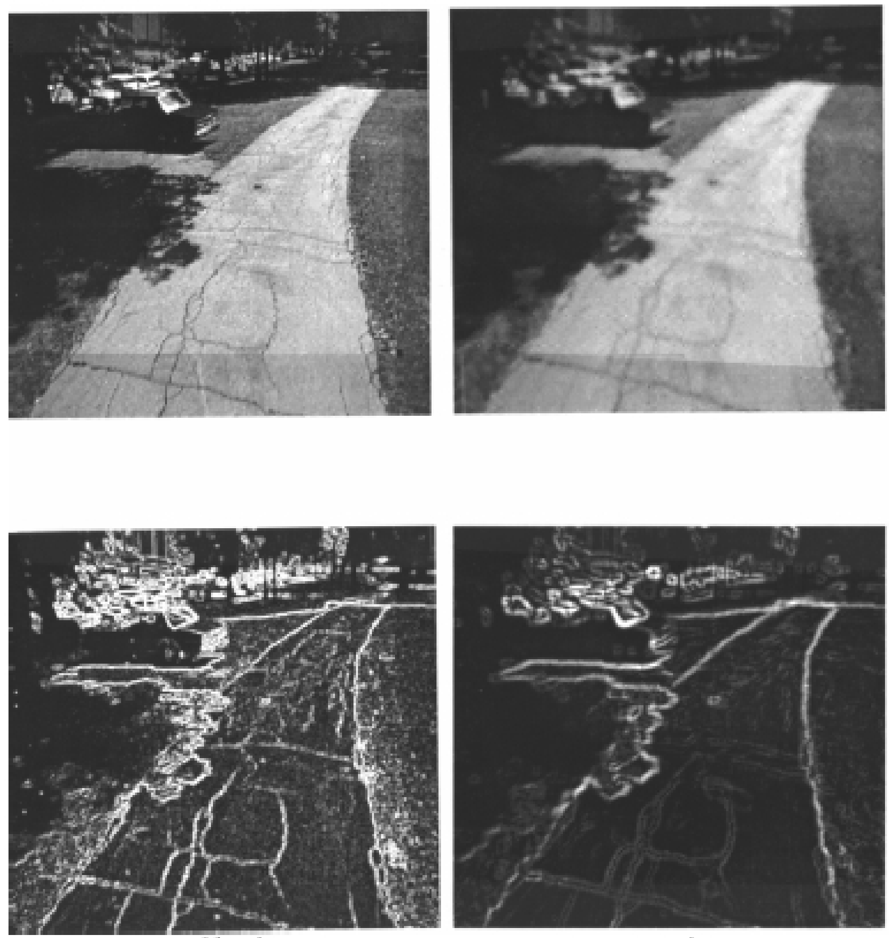


Combination zero-crossings and gradient

$\sigma = 3$

adapted from Martial Hebert, CMU

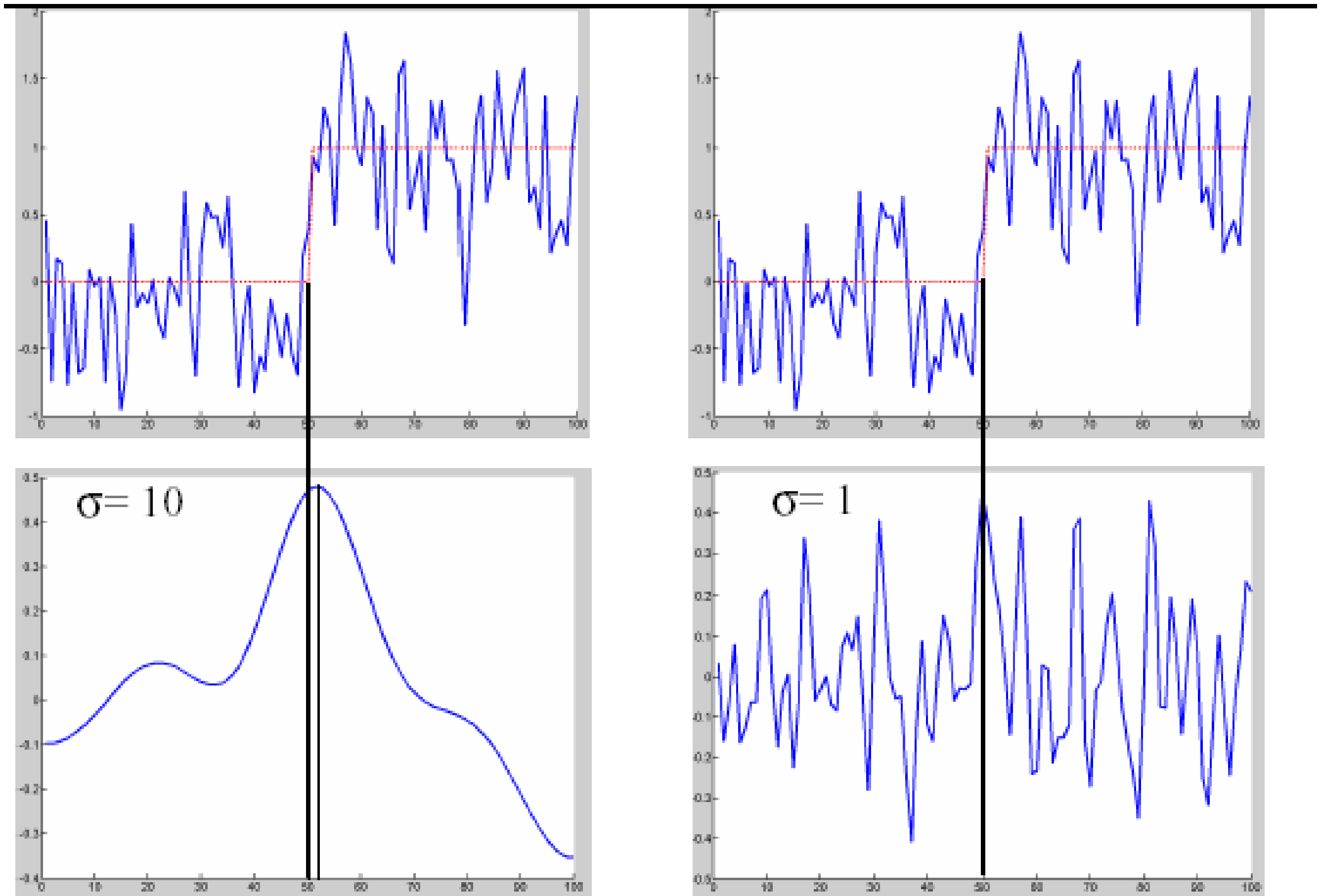
Effect of sigma – Detection vs. Localization



Small sigma Large sigma

adapted from Martial Hebert, CMU

Effect of sigma



Large σ → Good detection (high SNR)
Poor localization

Small σ → Poor detection (low SNR)
Good localization

adapted from Martial Hebert, CMU

Canny’s Edge detection

Several criteria can be chosen to characterize the performance of an edge detector

Good detection, i.e. robustness to noise
Good localization
Uniqueness to response

Canny takes two factors into account in designing the edge detector

A model of the kind of edges to be detected
A quantitative definition of the performance this edge detector is supposed to have

adapted from Martial Hebert, CMU

Canny's Edge detection

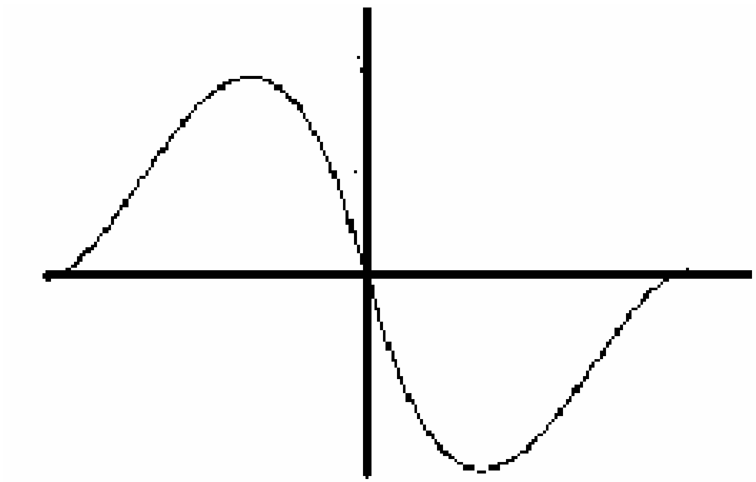
- Given a filter f, define the two objective functions:

  $\Lambda(f)$ large if f produces good localization

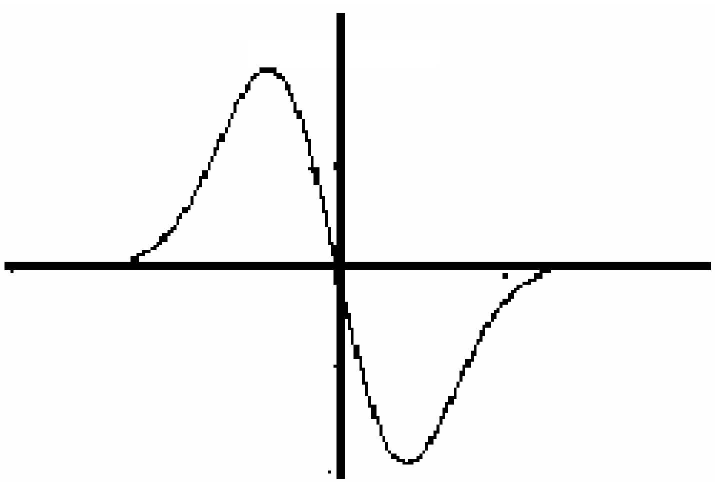
  $\Sigma(f)$ large if f produces good detection (high SNR)
- Problem: Find a family of filters f that maximizes the compromise criterion

$$\Lambda(f)\Sigma(f)$$

  under the constraint that a single peak is generated by a step edge
- Solution: Unique solution, a close approximation is the Gaussian derivative filter!



Canny



Derivative of Gaussian

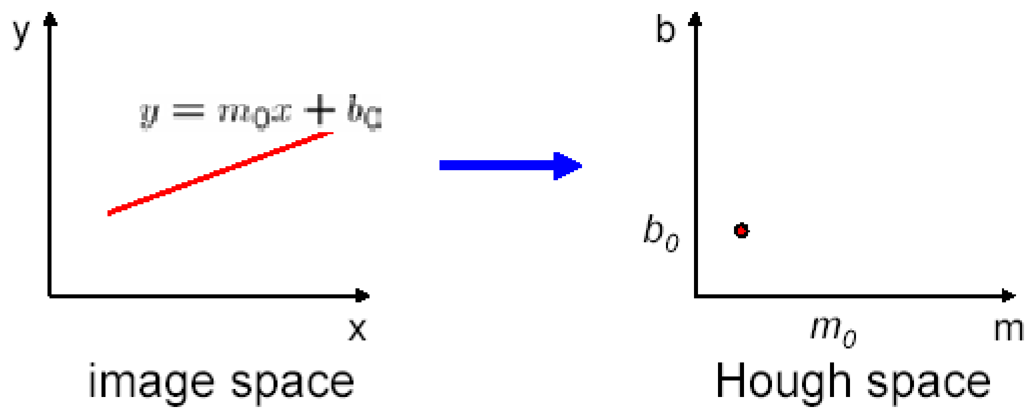
adapted from Martial Hebert, CMU

An edge is not a line


How can we detect ***lines*** ?

adapted from Steven Seitz, University of Washington

Hough Transform

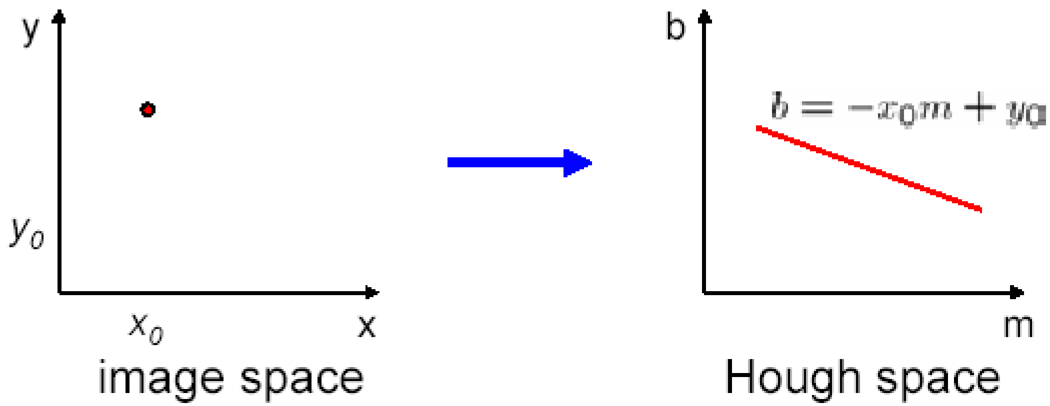


Connection between image (x,y) and Hough (m,b) spaces

- A line in the image corresponds to a point in Hough space
- To go from image space to Hough space:
  - given a set of points (x,y), find all (m,b) such that y = mx + b

adapted from Steven Seitz, University of Washington

Hough Transform



Connection between image (x,y) and Hough (m,b) spaces

- A line in the image corresponds to a point in Hough space
- To go from image space to Hough space:
  - given a set of points (x,y), find all (m,b) such that y = mx + b
- What does a point (x_0, y_0) in the image space map to?
  - A: the solutions of $b = -x_0 m + y_0$
  - this is a line in Hough space

adapted from Steven Seitz, University of Washington

Hough Transform

Typically use a different parameterization

$$d = x\cos\theta + y\sin\theta$$

- d is the perpendicular distance from the line to the origin
- θ is the angle this perpendicular makes with the x axis
- Why?

Basic Hough transform algorithm

1. Initialize H[d, θ]=0
2. for each edge point I[x,y] in the image

   for θ = 0 to 180

   $d = x\cos\theta + y\sin\theta$

   H[d, θ] += 1
3. Find the value(s) of (d, θ) where H[d, θ] is maximum
4. The detected line in the image is given by $d = x\cos\theta + y\sin\theta$

adapted from Steven Seitz, University of Washington