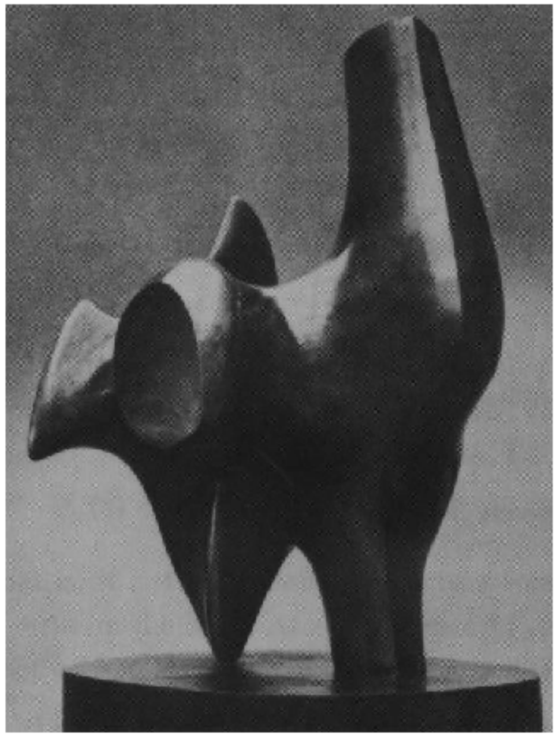
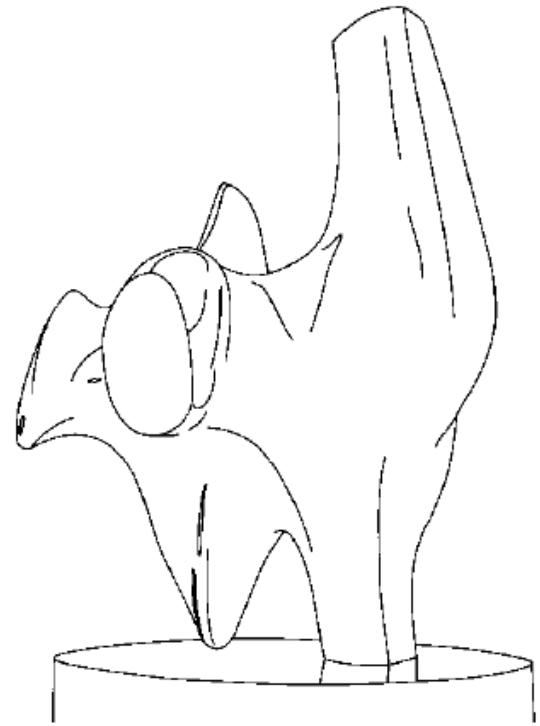
EIE4512 - Digital Image Processing
Edge Detection

香港中文大學(深圳)
The Chinese University of Hong Kong, Shenzhen

Zhen Li
lizhen@cuhk.edu.cn
School of Science and Engineering
The Chinese University of Hong Kong, Shen Zhen
March 5 -- 7, 2019

Edge detection

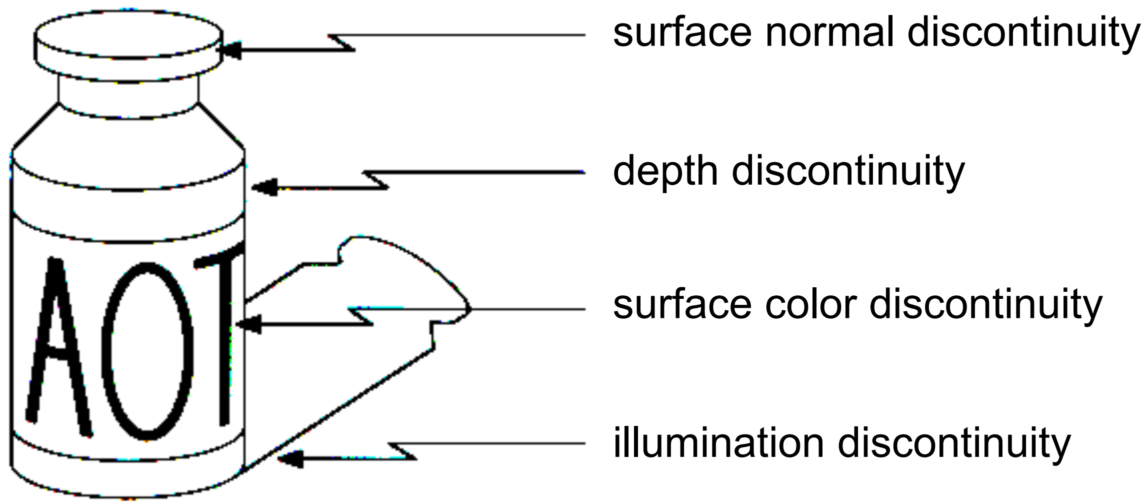




Convert a 2D image into a set of curves

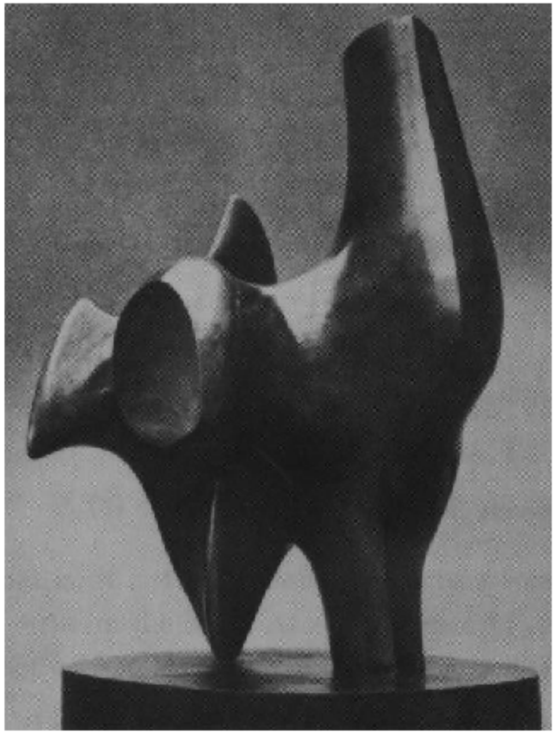
- Extracts salient features of the scene
- More compact than pixels

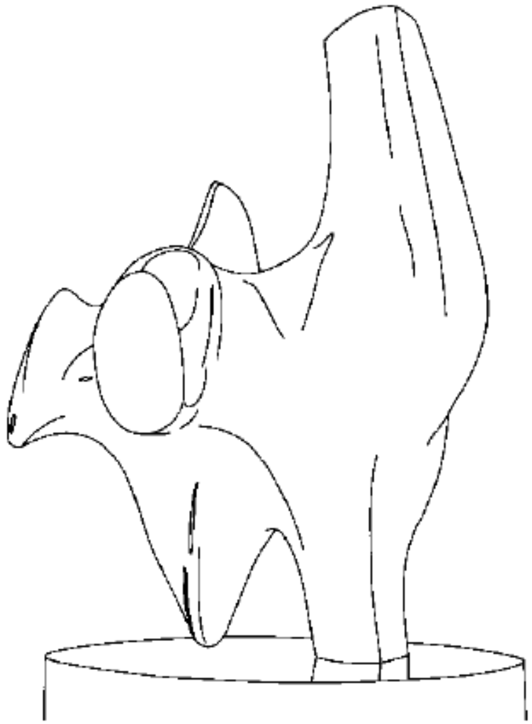
Origin of Edges



Edges are caused by a variety of factors

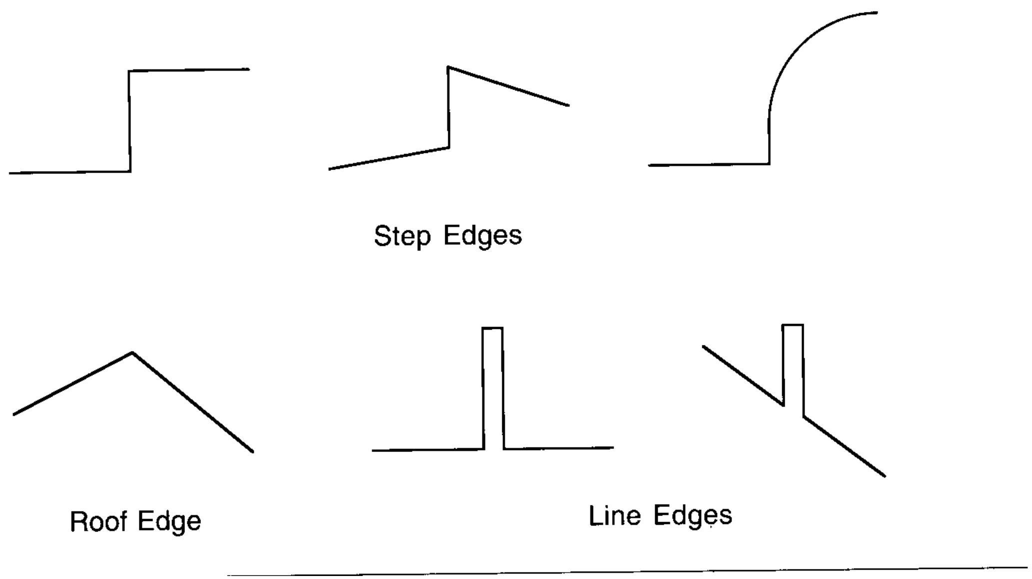
Edge detection





How can you tell that a pixel is on an edge?

Profiles of image intensity edges



Edge detection

1. Detection of short linear edge segments (edgels)

2. Aggregation of edgels into extended edges
   (maybe parametric description)

Edgel detection

- Difference operators
- Parametric-model matchers

Edge is Where Change Occurs

Change is measured by derivative in 1D

Biggest change, derivative has maximum magnitude

Or 2^{nd} derivative is zero.

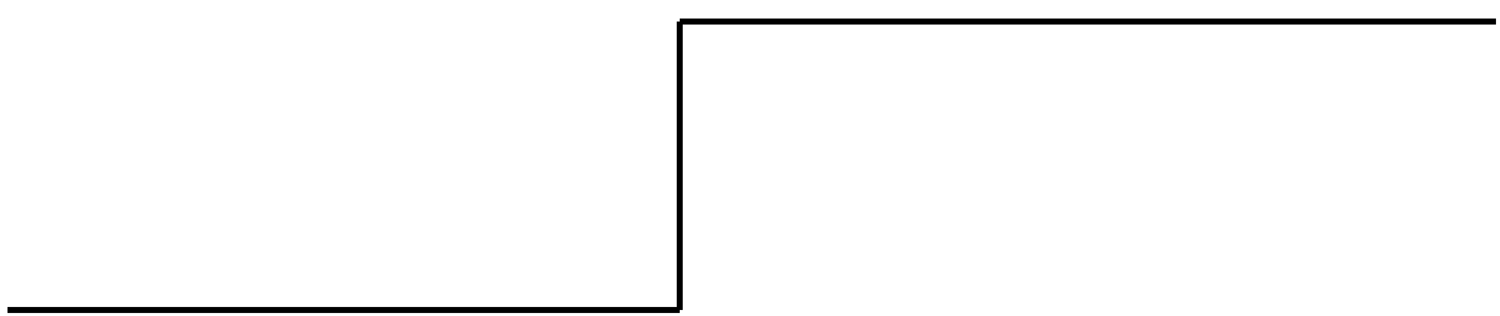
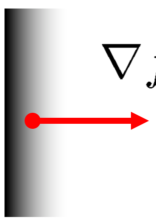


Image gradient

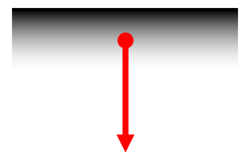
The gradient of an image:

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

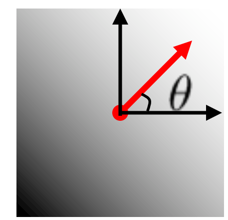
The gradient points in the direction of most rapid change in intensity



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



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



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

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

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

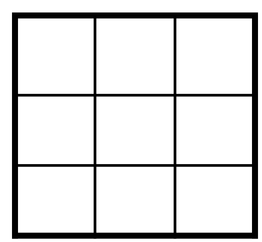
The discrete gradient

How can we differentiate a *digital* image f[x,y]?

- Option 1: reconstruct a continuous image, then take gradient
- Option 2: take discrete derivative (finite difference)

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

How would you implement this as a cross-correlation?



H

The Sobel operator

Better approximations of the derivatives exist

- The *Sobel* operators below are very commonly used

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

$$s_x \qquad\qquad\qquad s_y$$

- The standard defn. of the Sobel operator omits the 1/8 term
  - doesn't make a difference for edge detection
  - the 1/8 term **is** needed to get the right gradient value, however

Gradient operators

$$\Delta_1 \quad \begin{matrix} 0 & 1 \\ -1 & 0 \end{matrix} \qquad \Delta_2 \quad \begin{matrix} 1 & 0 \\ 0 & -1 \end{matrix}$$

(a)

$$\Delta_1 \quad \begin{matrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{matrix} \qquad \Delta_2 \quad \begin{matrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{matrix}$$

(b)

$$\Delta_1 \quad \begin{matrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{matrix} \qquad \Delta_2 \quad \begin{matrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{matrix}$$

(c)

$$\Delta_1 \quad \begin{matrix} -3 & -1 & 1 & 3 \\ -3 & -1 & 1 & 3 \\ -3 & -1 & 1 & 3 \\ -3 & -1 & 1 & 3 \end{matrix} \qquad \Delta_2 \quad \begin{matrix} 3 & 3 & 3 & 3 \\ 1 & 1 & 1 & 1 \\ -1 & -1 & -1 & -1 \\ -3 & -3 & -3 & -3 \end{matrix}$$

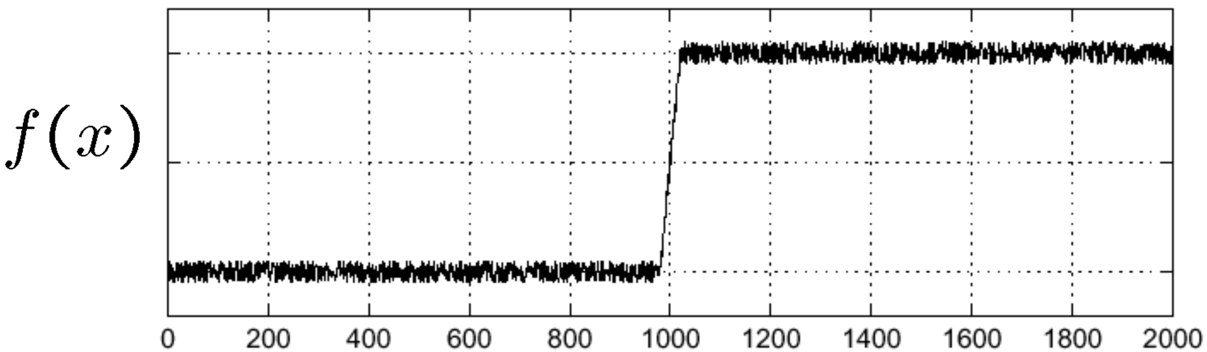
(d)

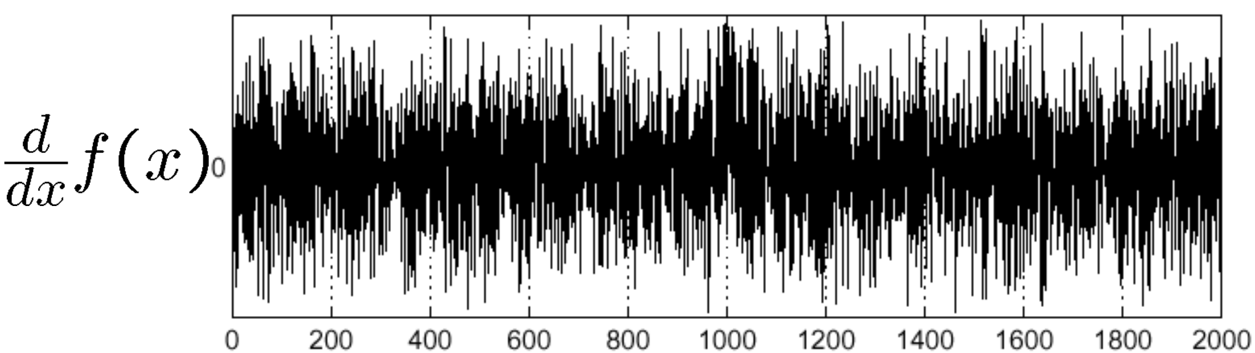
(a): Roberts' cross operator (b): 3x3 Prewitt operator
(c): Sobel operator (d) 4x4 Prewitt operator

Effects of noise

Consider a single row or column of the image

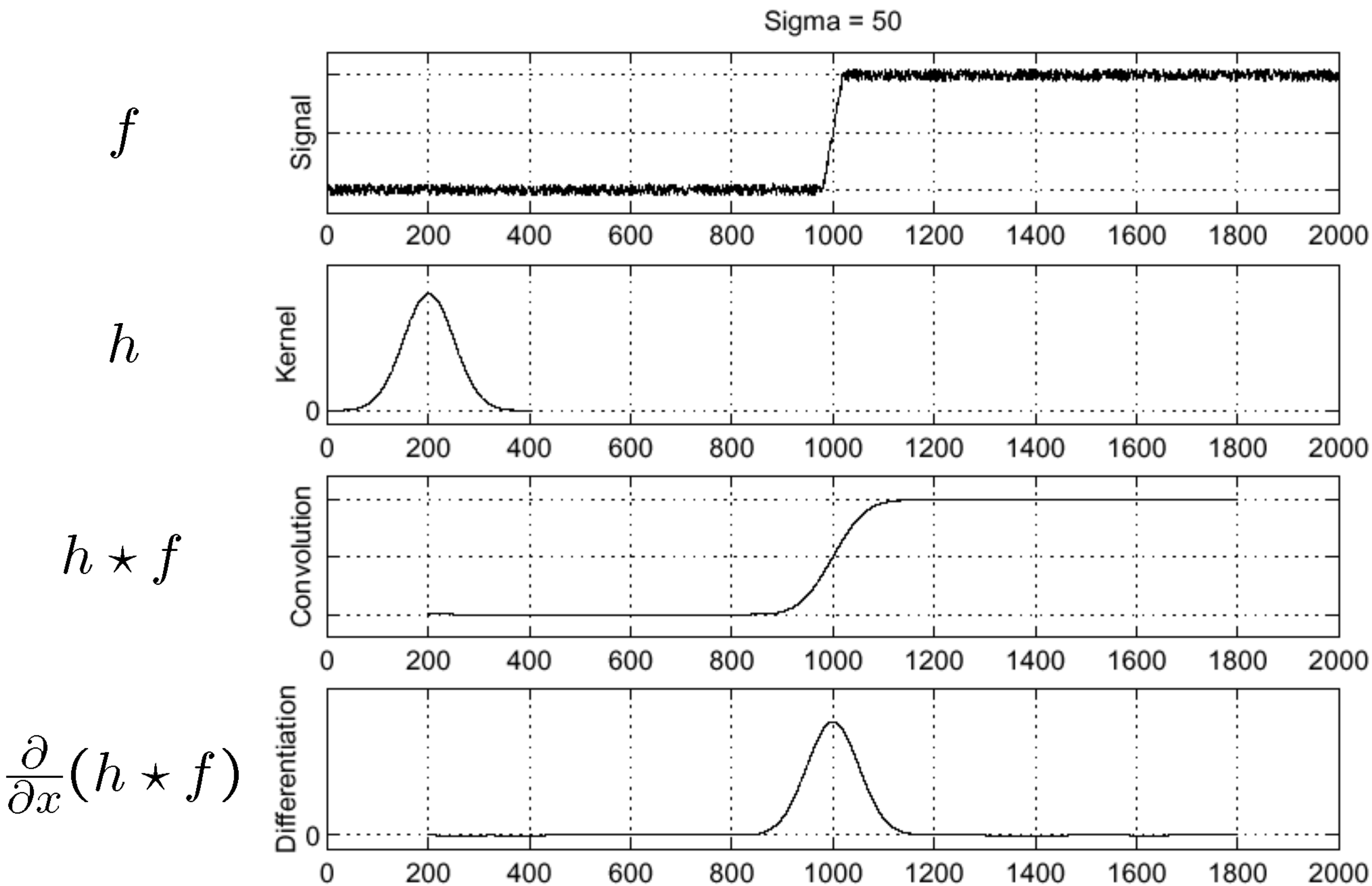
- Plotting intensity as a function of position gives a signal





Where is the edge?

Solution: smooth first



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

Derivative theorem of convolution

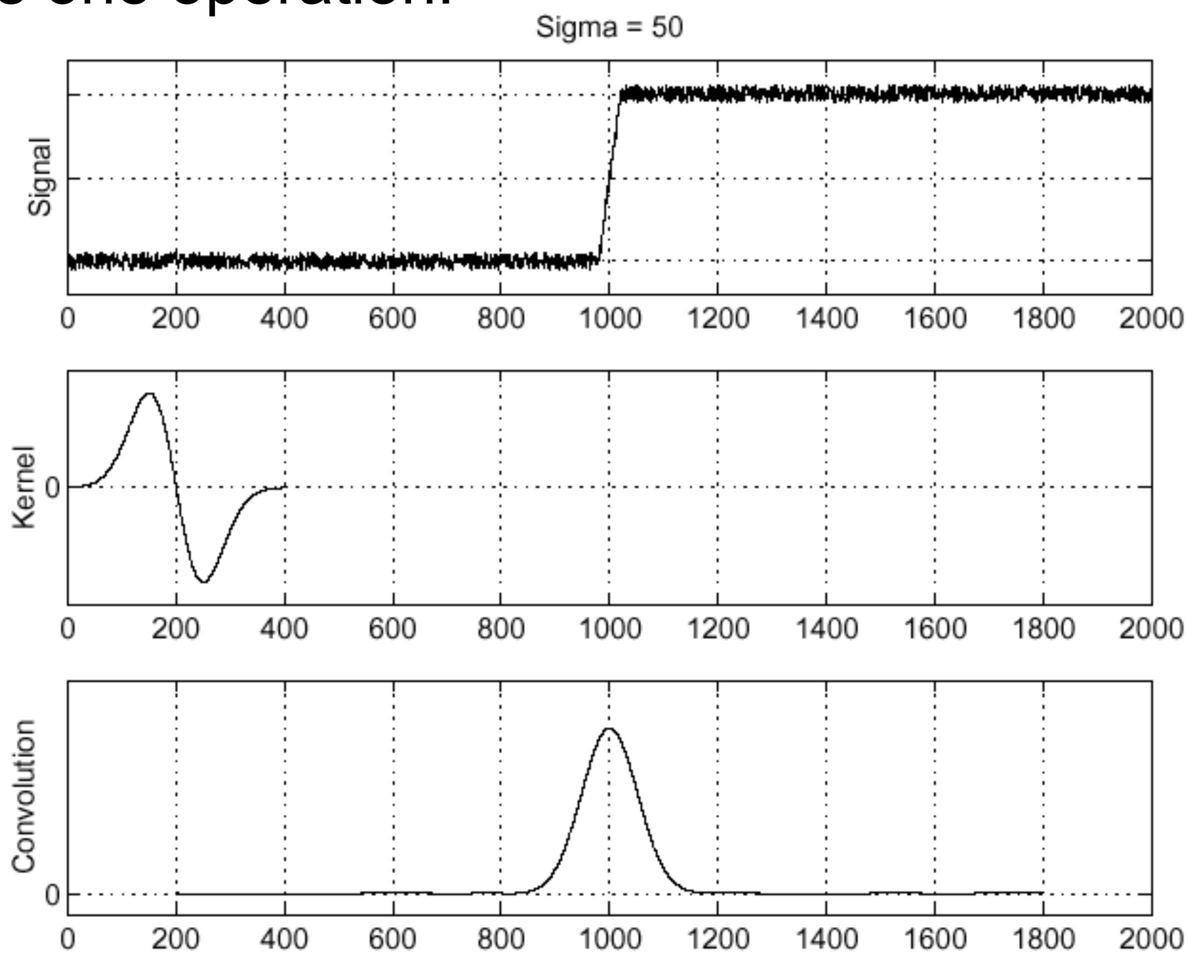
$$\frac{\partial}{\partial x}(h \star f) = (\frac{\partial}{\partial x}h) \star f$$

This saves us one operation:

f

$\frac{\partial}{\partial x}h$

$(\frac{\partial}{\partial x}h) \star f$



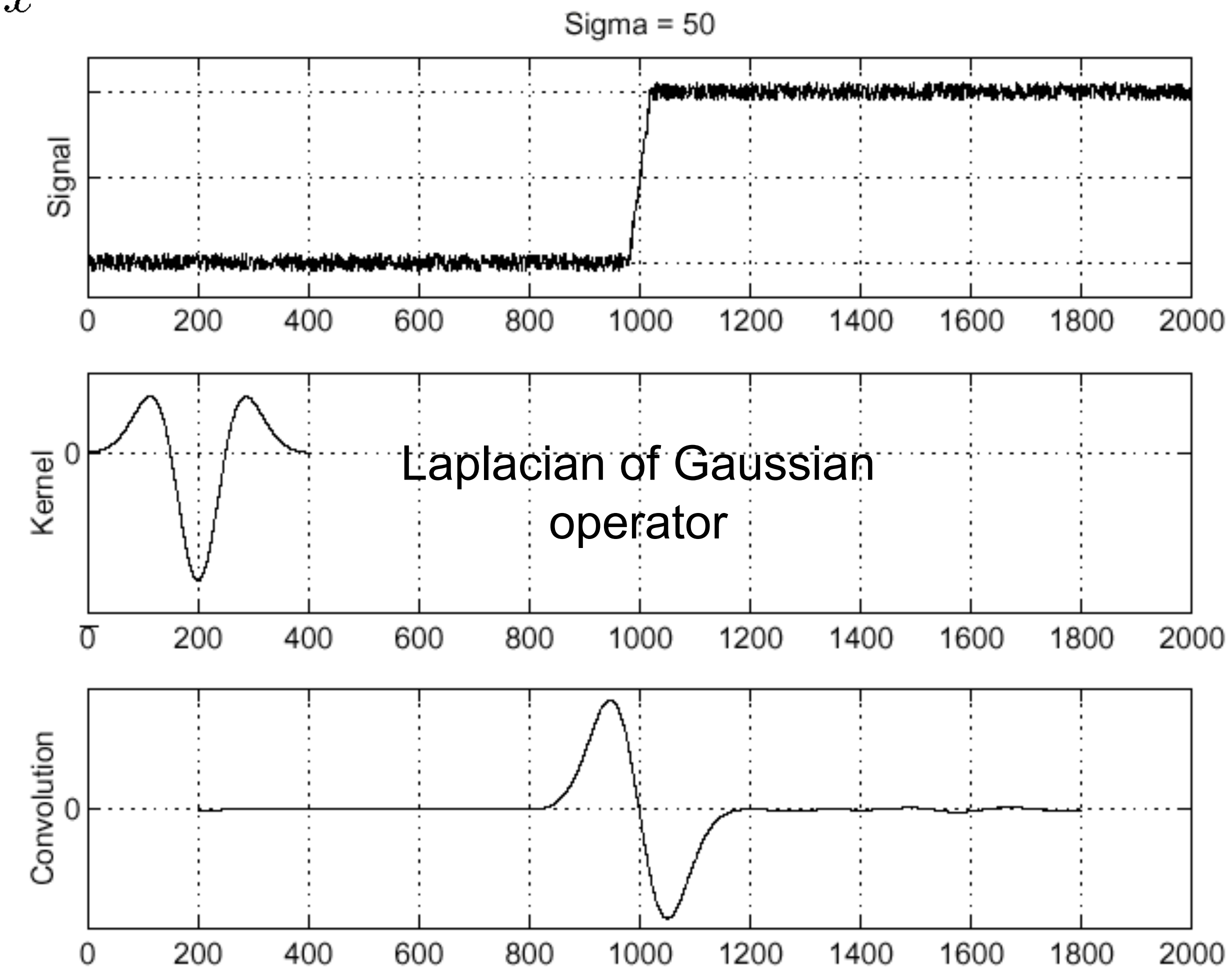
Laplacian of Gaussian

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

f

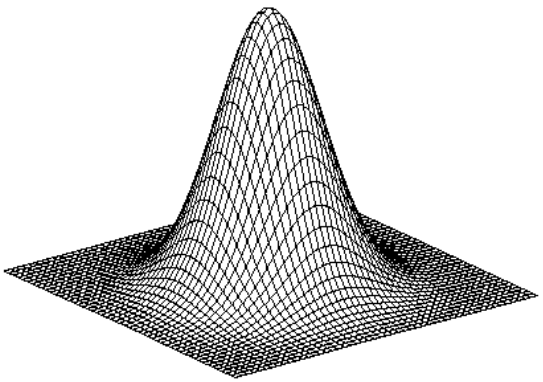
$\frac{\partial^2}{\partial x^2}h$

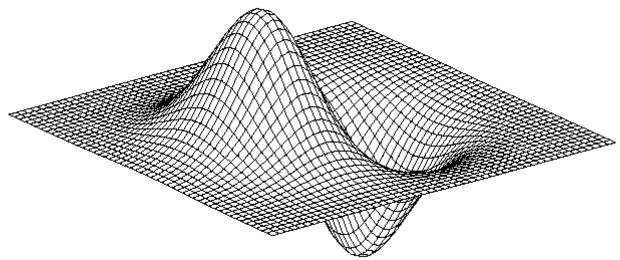
$(\frac{\partial^2}{\partial x^2}h) \star f$



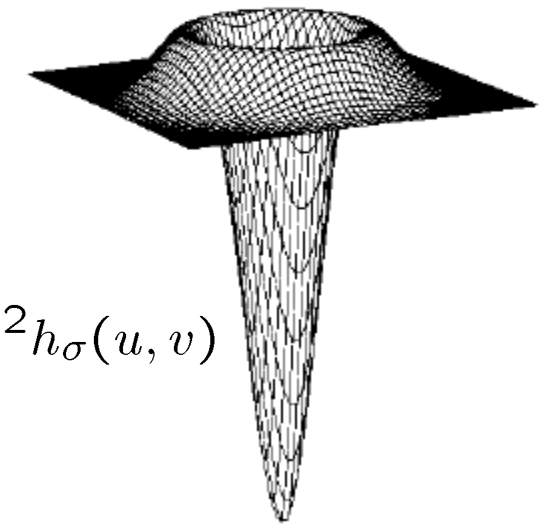
Where is the edge? Zero-crossings of bottom graph

2D edge detection filters





Laplacian of Gaussian



Gaussian

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

derivative of Gaussian

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

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

∇^2 is the **Laplacian** operator:

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

Optimal Edge Detection: Canny

Assume:

- Linear filtering
- Additive iid Gaussian noise

Edge detector should have:

- Good Detection. Filter responds to edge, not noise.
- Good Localization: detected edge near true edge.
- Single Response: one per edge.

Optimal Edge Detection: Canny (continued)

Optimal Detector is approximately Derivative of Gaussian.

Detection/Localization trade-off

- More smoothing improves detection
- And hurts localization.

This is what you might guess from (detect change) + (remove noise)

The Canny edge detector

original image (Lena)

The Canny edge detector

norm of the gradient

The Canny edge detector

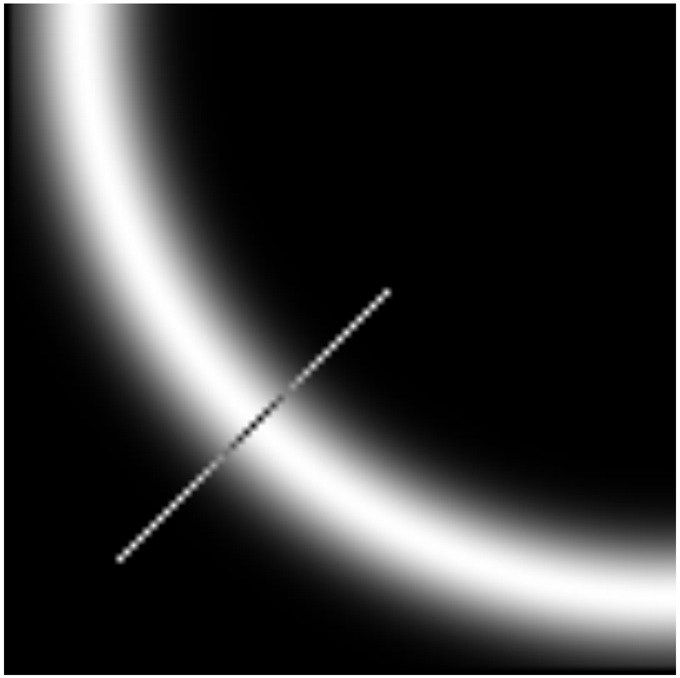
thresholding

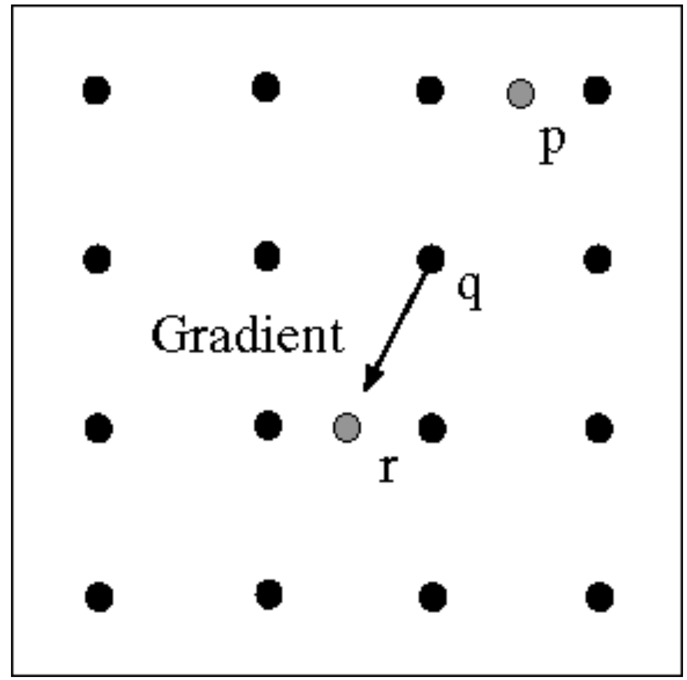
The Canny edge detector

thinning

(non-maximum suppression)

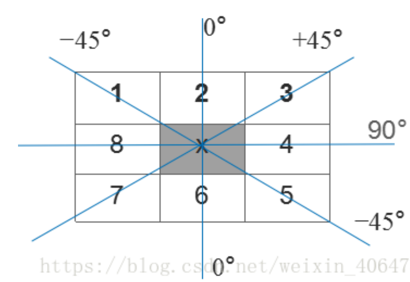
Non-maximum suppression



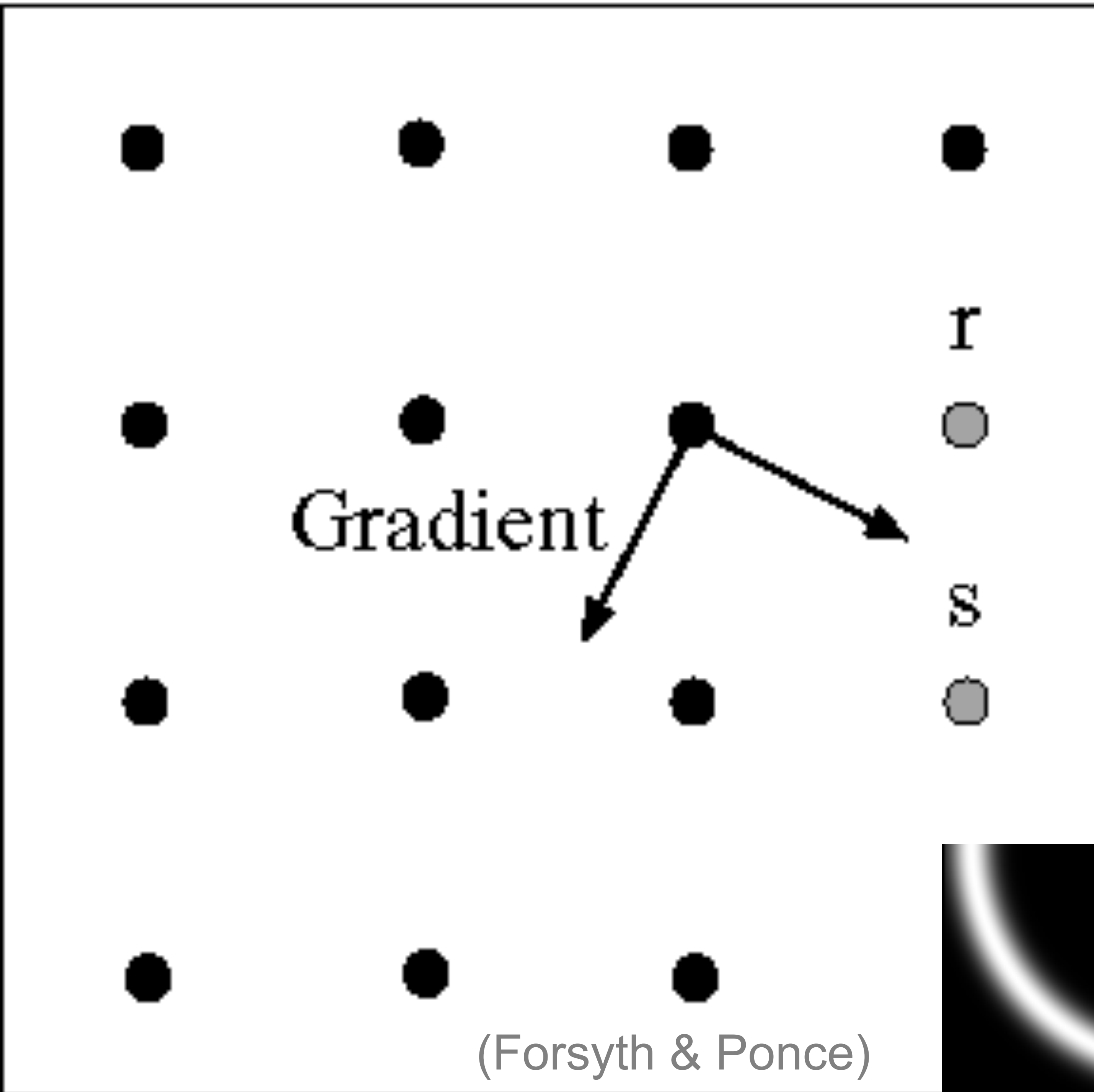


Check if pixel is local maximum along gradient direction

- requires checking interpolated pixels p and r

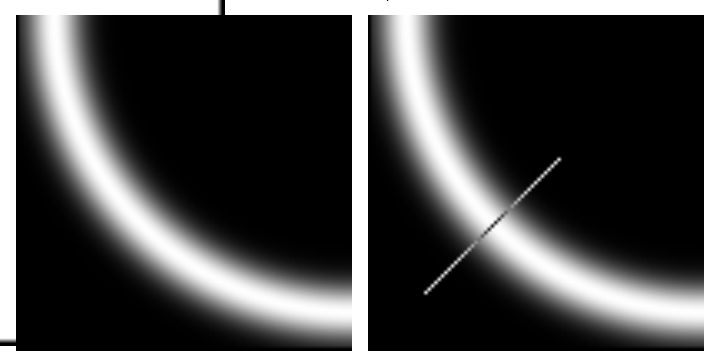


Predicting the next edge point



(Forsyth & Ponce)

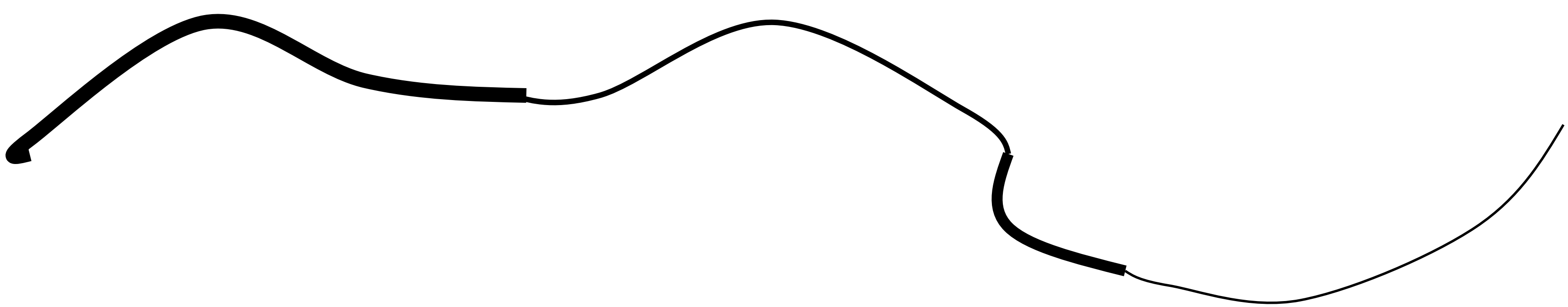
Assume the marked point is an edge point. Then we construct the tangent to the edge curve (which is normal to the gradient at that point) and use this to predict the next points (here either r or s).



Hysteresis

Check that maximum value of gradient value is sufficiently large

- drop-outs? use **hysteresis**
  - use a high threshold to start edge curves and a low threshold to continue them.



Edge detection by subtraction

original

Edge detection by subtraction

smoothed (5x5 Gaussian)

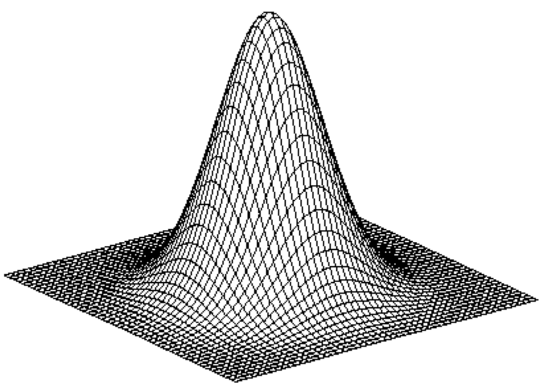
Edge detection by subtraction

smoothed – original
(scaled by 4, offset +128)

Why does this work?

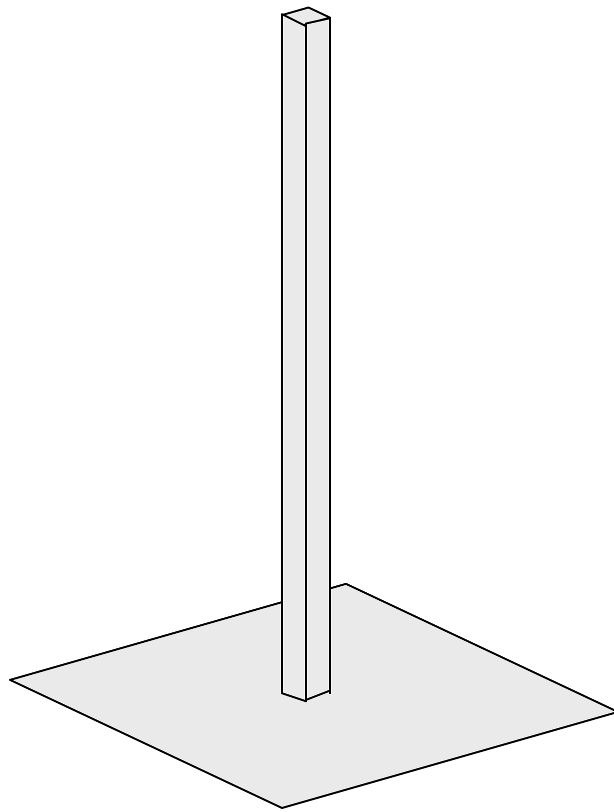
filter demo

Gaussian - image filter



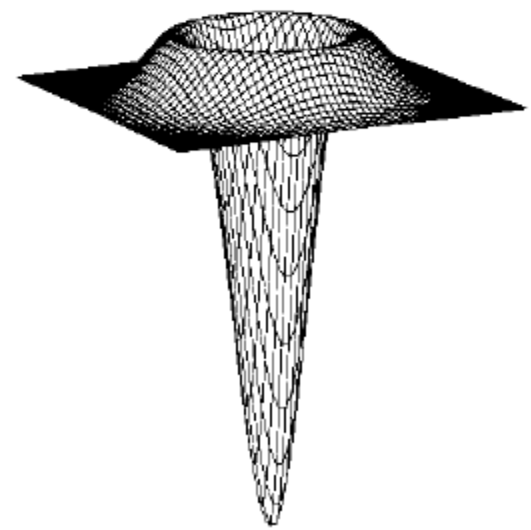
Gaussian

—



delta function

≈



Laplacian of Gaussian

Process of Canny edge detection algorithm

The Process of Canny edge detection algorithm can be broken down to 5 different steps:

1. Apply Gaussian filter to smooth the image in order to remove the noise

2. Find the intensity gradients of the image

3. Apply non-maximum suppression to get rid of spurious response to edge detection

4. Apply double threshold to determine potential edges

5. Track edge by hysteresis: Finalize the detection of edges by suppressing all the other edges that are weak and not connected to strong edges.

Effect of σ (Gaussian kernel size)

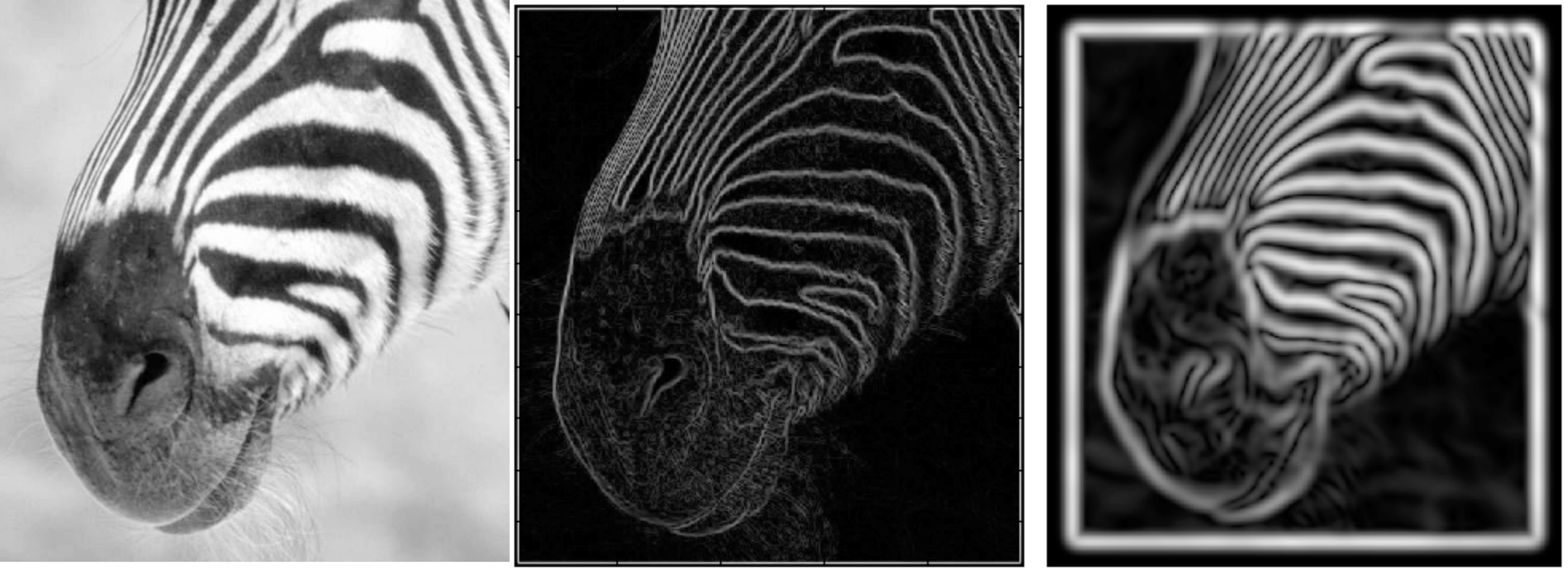
original

Canny with $\sigma = 1$

Canny with $\sigma = 2$

The choice of σ depends on desired behavior

- large σ detects large scale edges
- small σ detects fine features



Scale

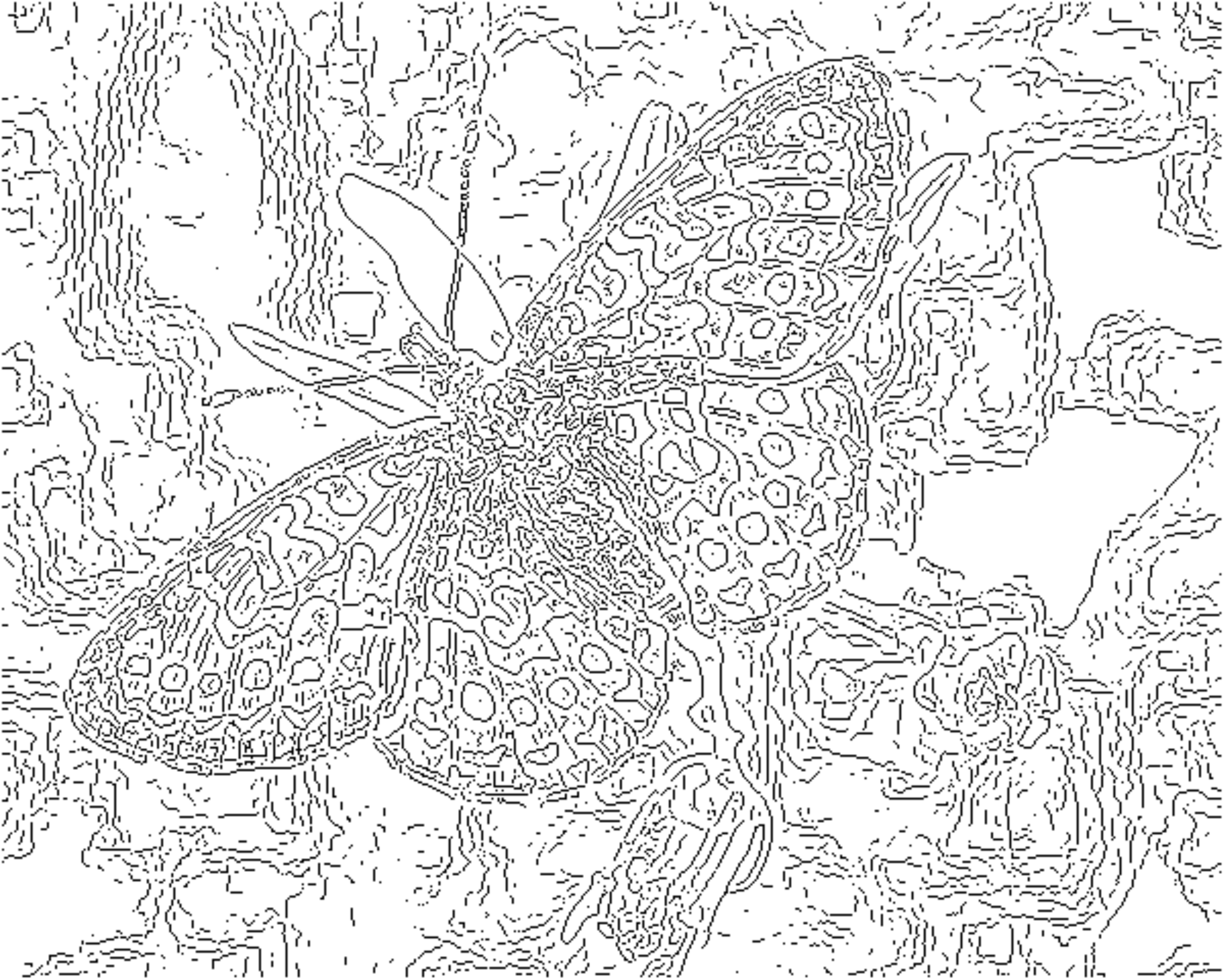
Smoothing

Eliminates noise edges.

Makes edges smoother.

Removes fine detail.

(Forsyth & Ponce)

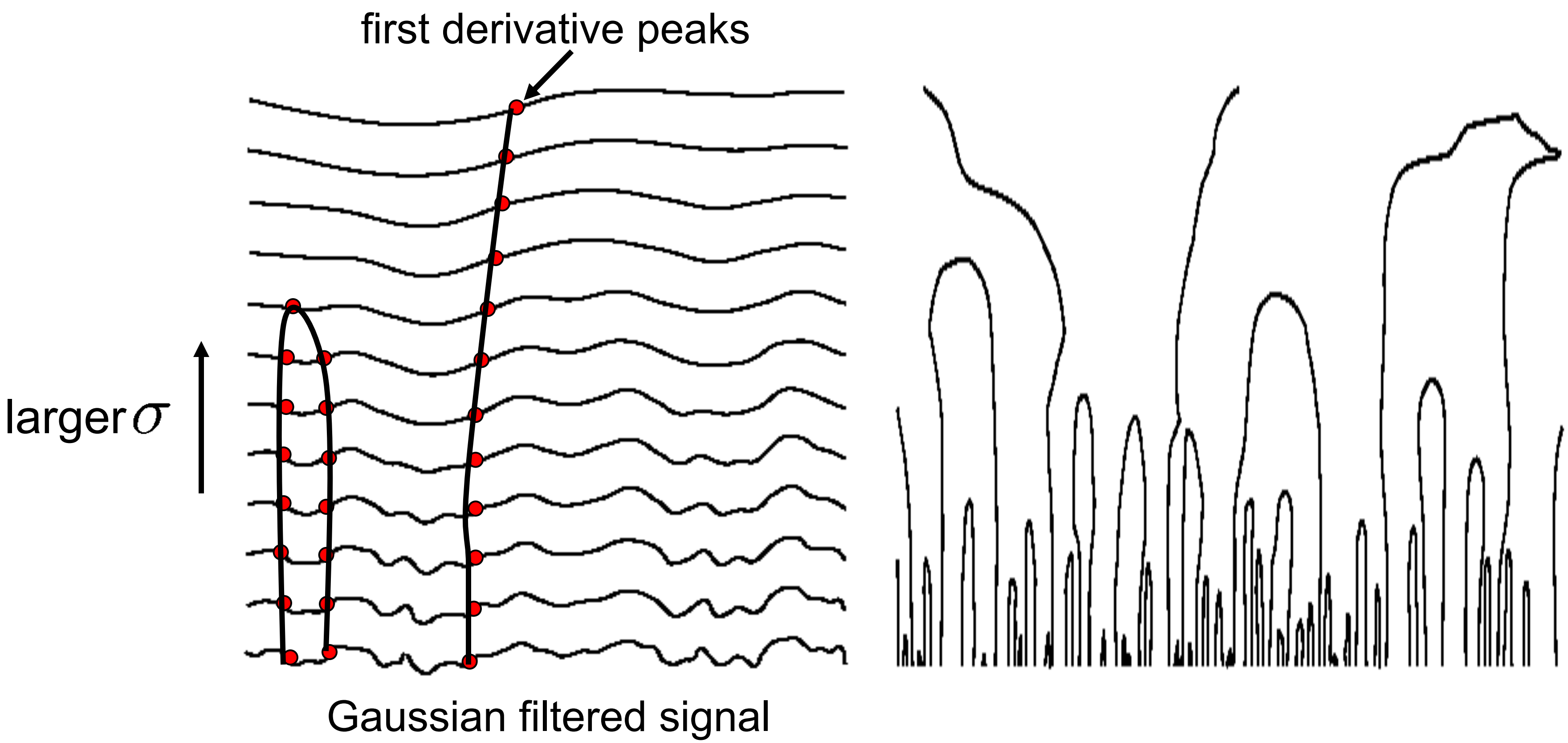


fine scale
high
threshold

coarse
scale,
high
threshold

coarse
scale
low
threshold

Scale space (Witkin 83)



Properties of scale space (w/ Gaussian smoothing)

- edge position may shift with increasing scale (σ)
- two edges may merge with increasing scale
- an edge may ***not*** split into two with increasing scale

Canny edge detection example

An edge is not a line...

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

Finding lines in an image

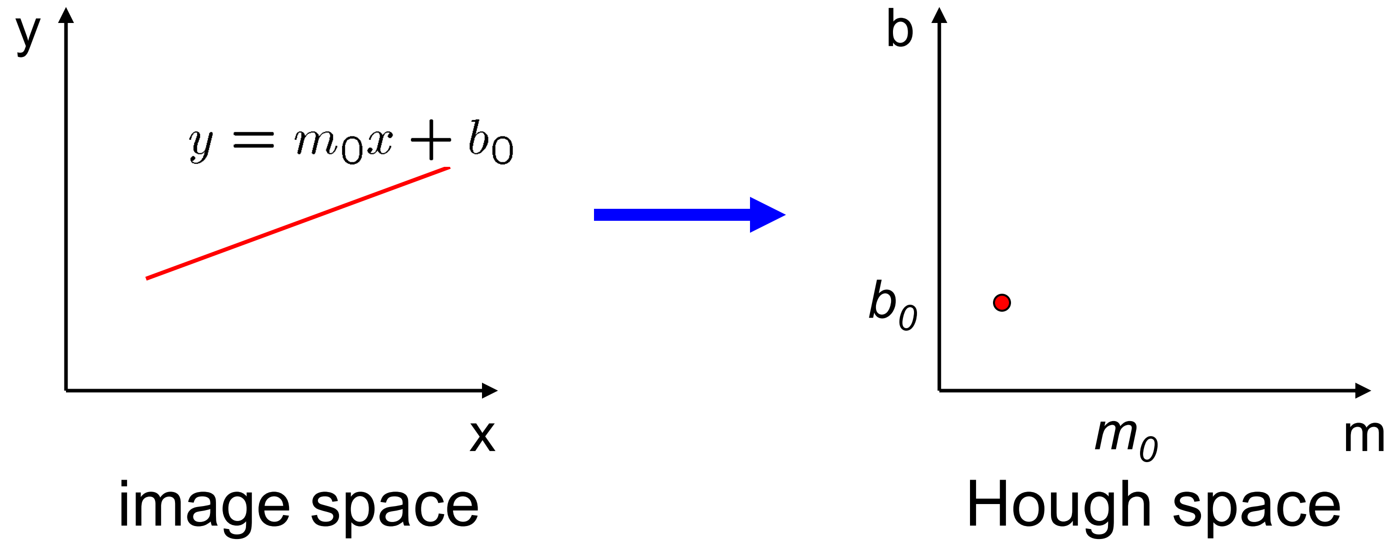
Option 1:

- Search for the line at every possible position/orientation
- What is the cost of this operation?

Option 2:

- Use a voting scheme: Hough transform

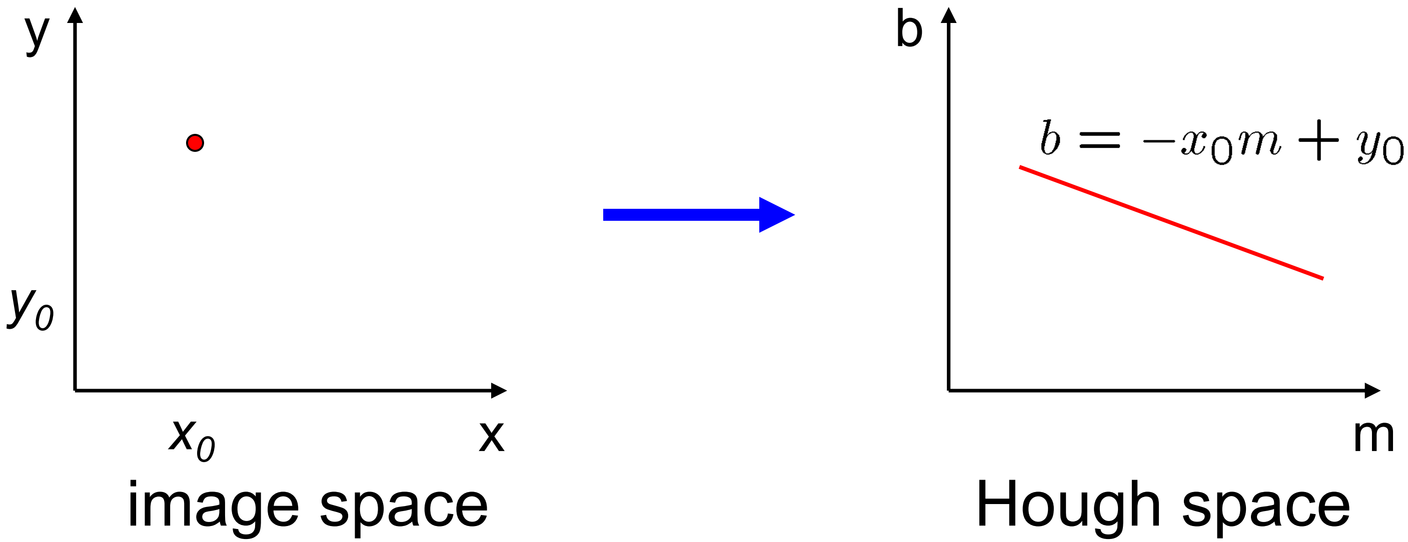
Finding lines in an image



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

Finding lines in an image



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

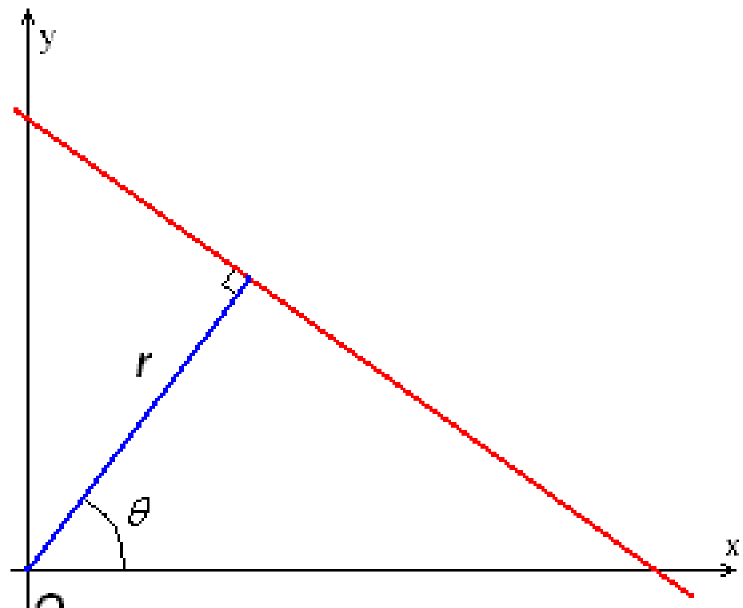
- A line in the image corresponds to a point in Hough space
- To go from image space to Hough space:
  - given a set of points (x,y), find all (m,b) such that y = mx + b
- What does a point (x_0, y_0) in the image space map to?
  - A: the solutions of $b = -x_0 m + y_0$
  - this is a line in Hough space

Hough transform algorithm

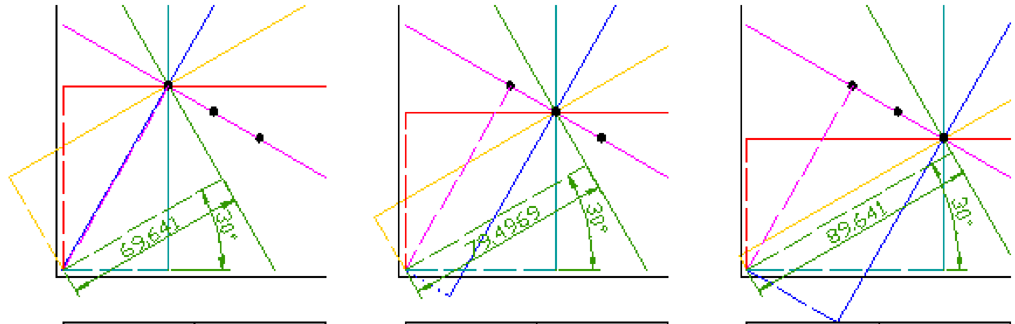
Typically use a different parameterization

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

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



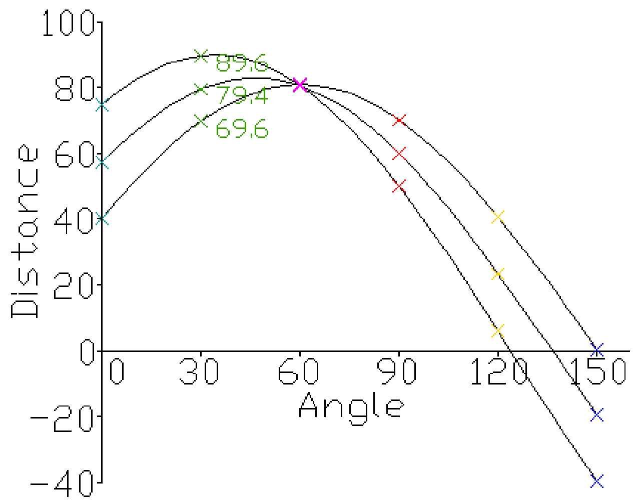
Hough transform example



Angle	Dist.
0	40
30	69.6
60	81.2
90	70
120	40.6
150	0.4

Angle	Dist.
0	57.1
30	79.5
60	80.5
90	60
120	23.4
150	−19.5

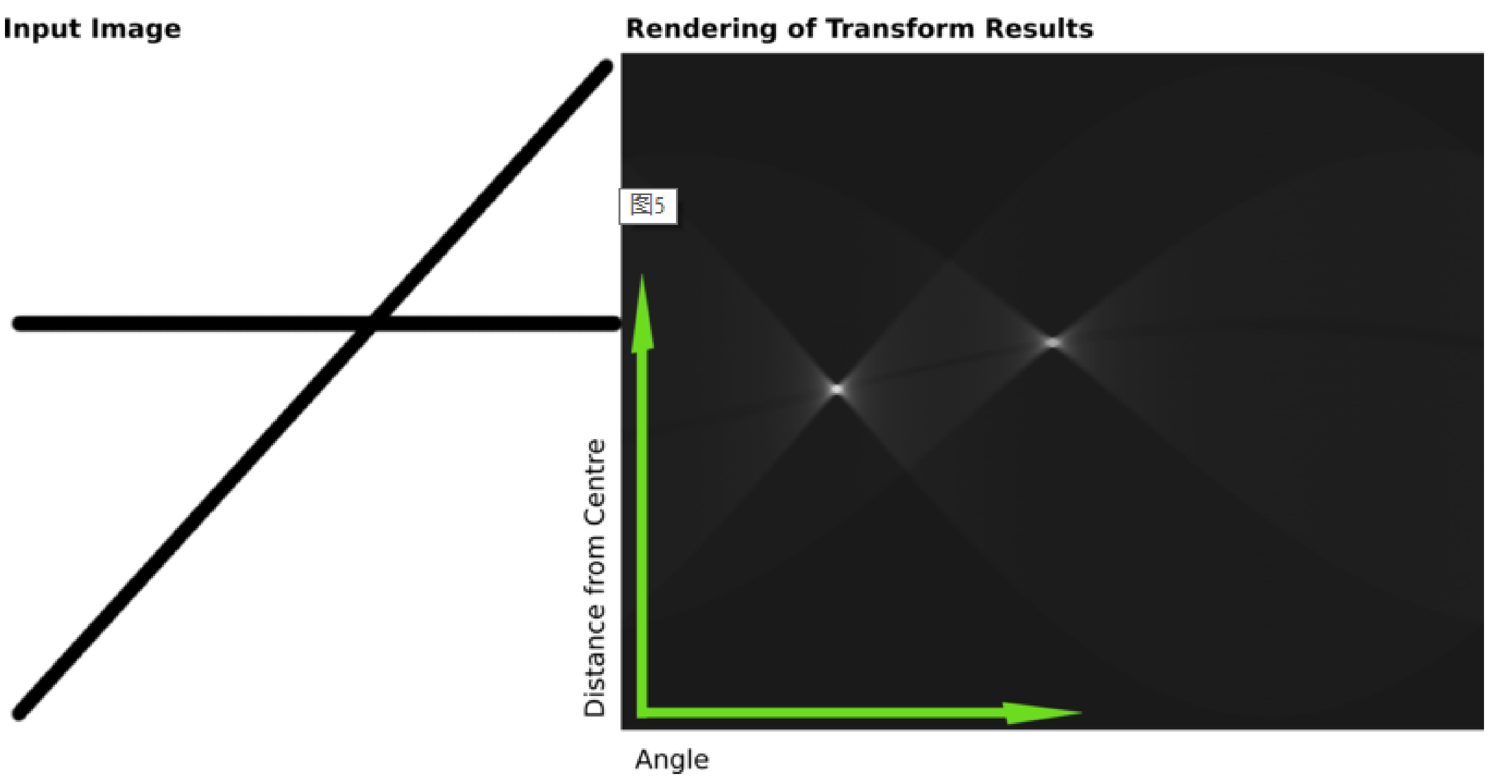
Angle	Dist.
0	74.6
30	89.6
60	80.6
90	50
120	6.0
150	−39.6



<u>A point in spatial domain has been transfered to a sin curve in Hough space</u>

<u>Line in spatial domain has been transfered to a point in Hough space.</u>

Hough transform example



Inverse Hough transform:

$$y = \frac{-cos(\theta_l)}{sin(\theta_l)}x + \frac{\rho_l}{sin(\theta_l)}$$

Hough transform algorithm

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

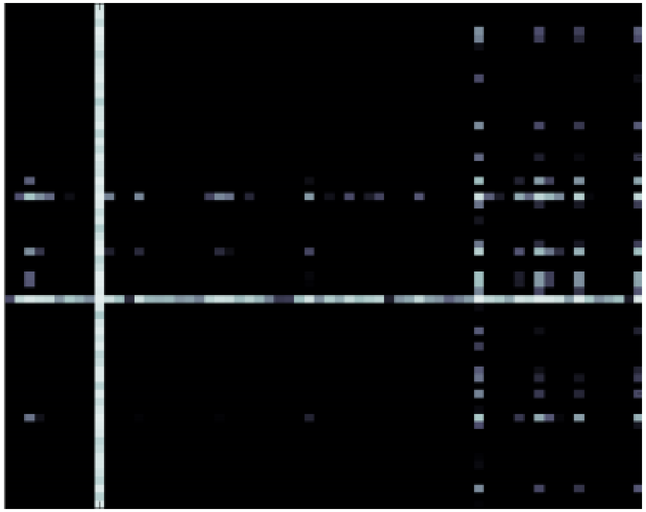
   for θ = 0 to 180

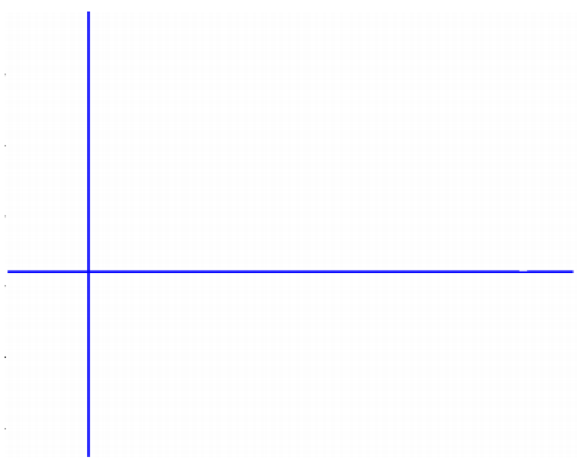
   $$d = x cos\theta + y sin\theta$$

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

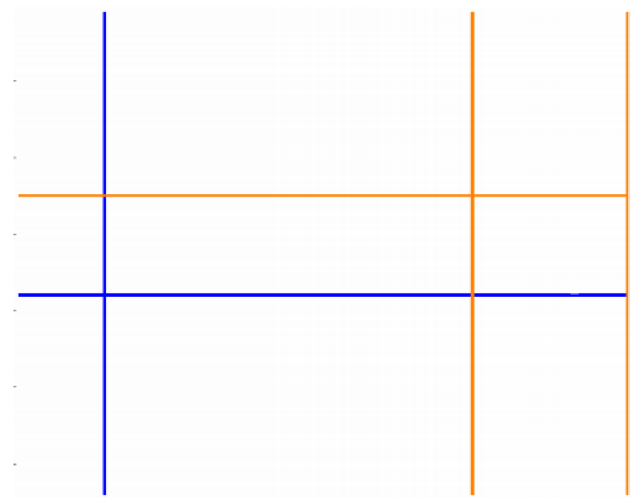
What's the running time (measured in # votes)?

Hough transform algorithm example





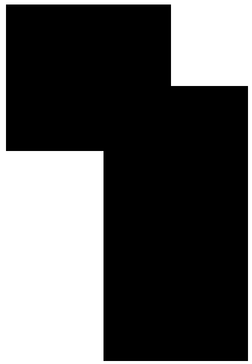
Threshold = 30



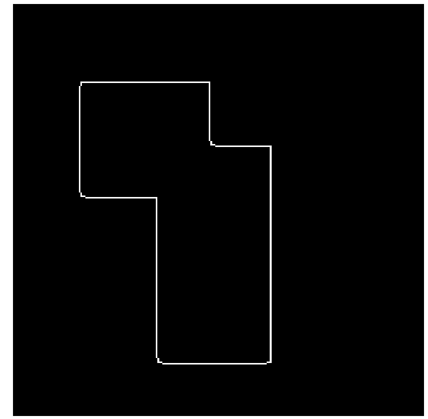
Threshold = 20

https://blog.csdn.net/YuYunTan/article/details/80141392

Hough transform algorithm example

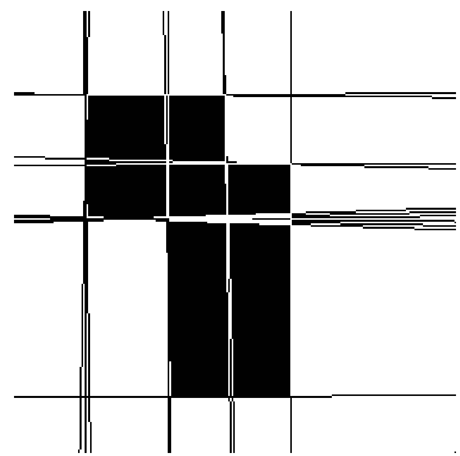


Original



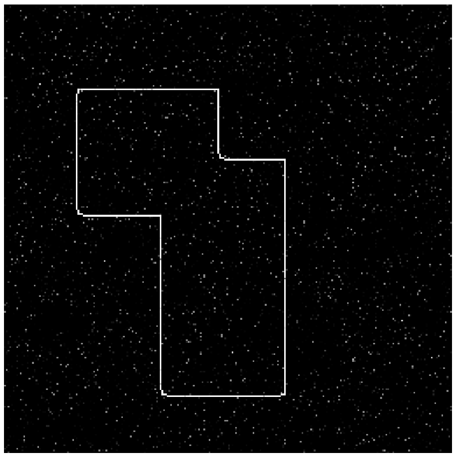
Candy edge

Hough Transform

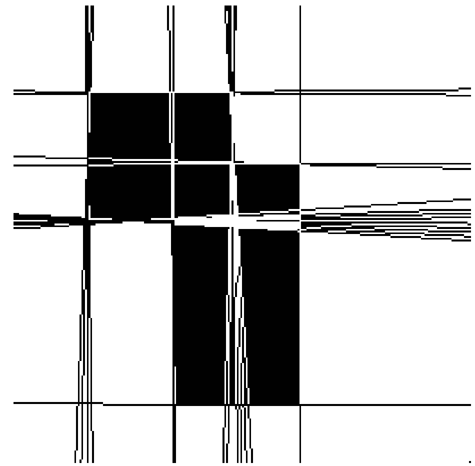


Inverse Hough Transform

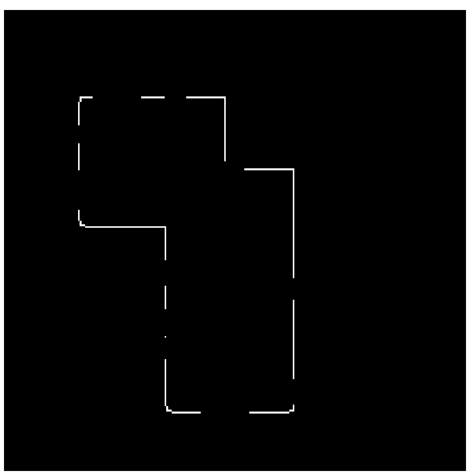
Hough transform algorithm robustness



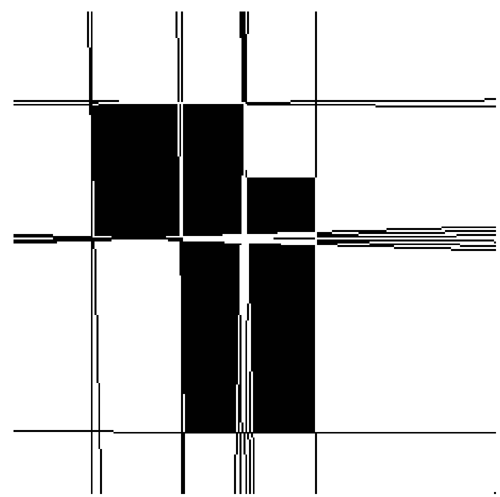
Original+salt-pepper



Inverse Hough Transform



Original with disconnection



Inverse Hough Transform