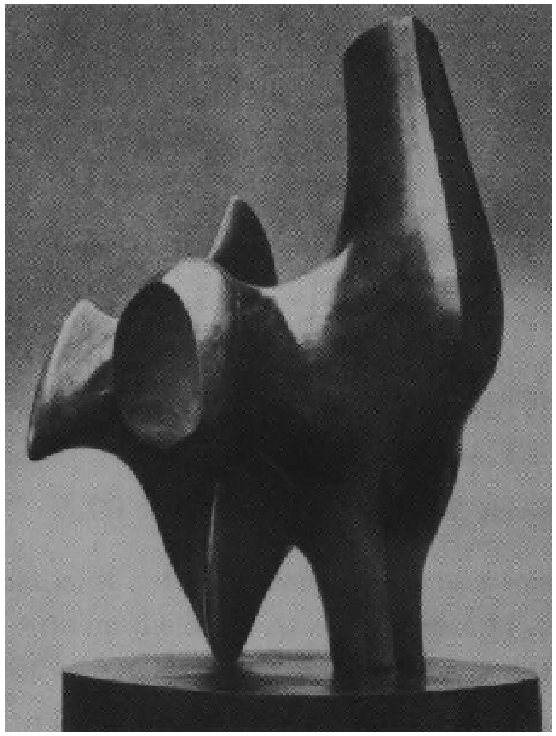
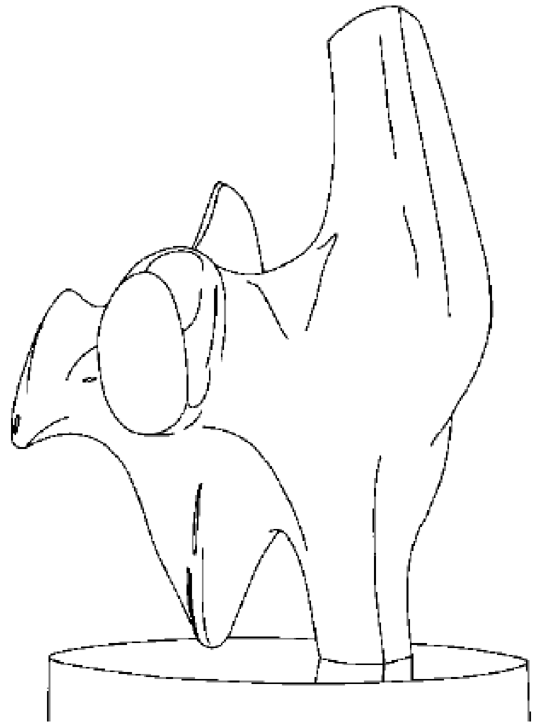
Edge Detection

CS 111

Slides from Cornelia Fermüller and Marc Pollefeys

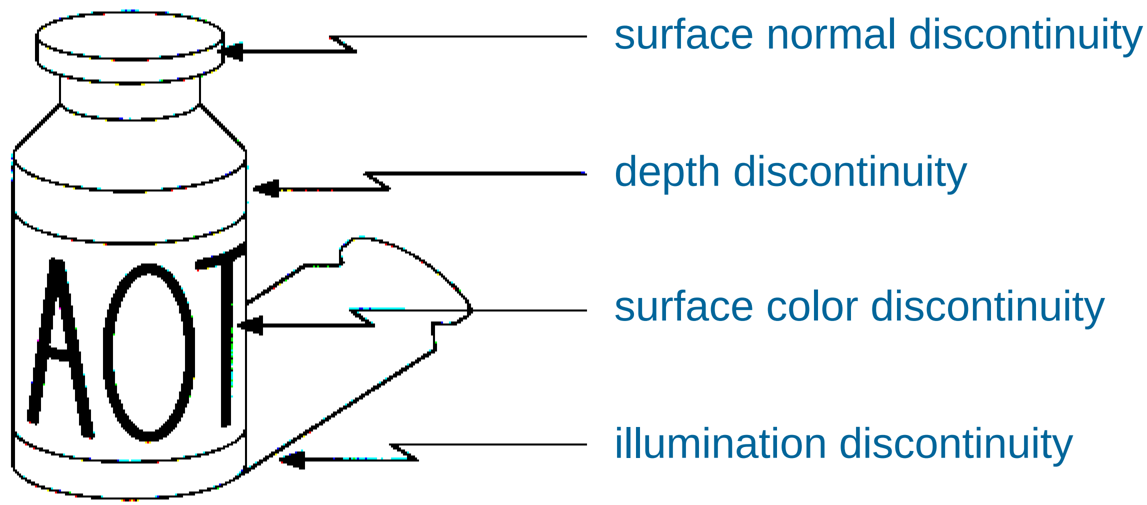
Edge detection





- 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

1. Detection of short linear edge segments (edgels)
2. Aggregation of edgels into extended edges
3. Maybe parametric description

Edge is Where Change Occurs

- Change is measured by derivative in 1D
- Biggest change, derivative has maximum magnitude
- Or 2nd derivative is zero.

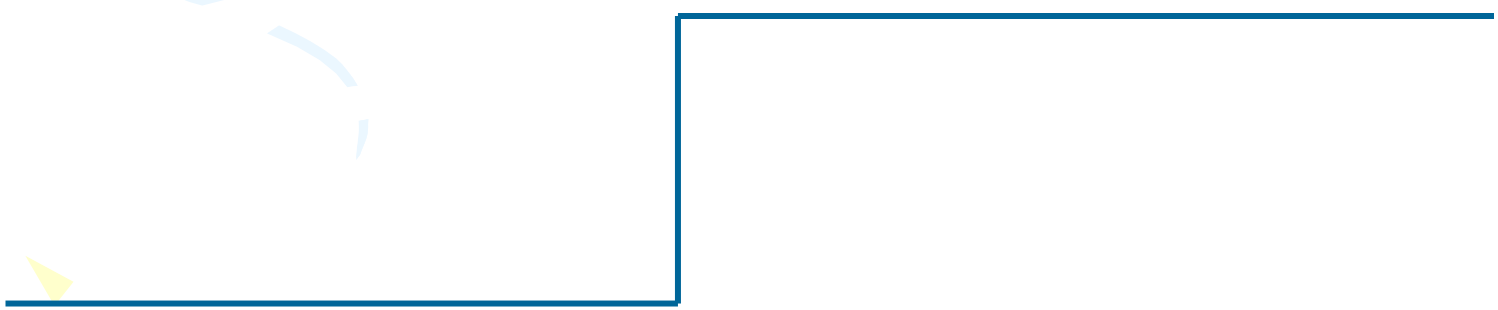
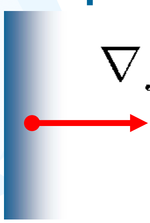


Image gradient

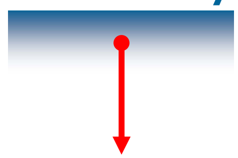
- The gradient of an image:

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

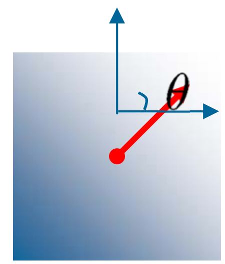
- The gradient points in the direction of most rapid change in intensity



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



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



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

- The gradient direction is given by:

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

  – Perpendicular to the edge

- The *edge strength* is given by the magnitude

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

How discrete gradient?

- By finite differences

  $f(x+1, y) - f(x, y)$

  $f(x, y+1) - f(x, y)$

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)

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

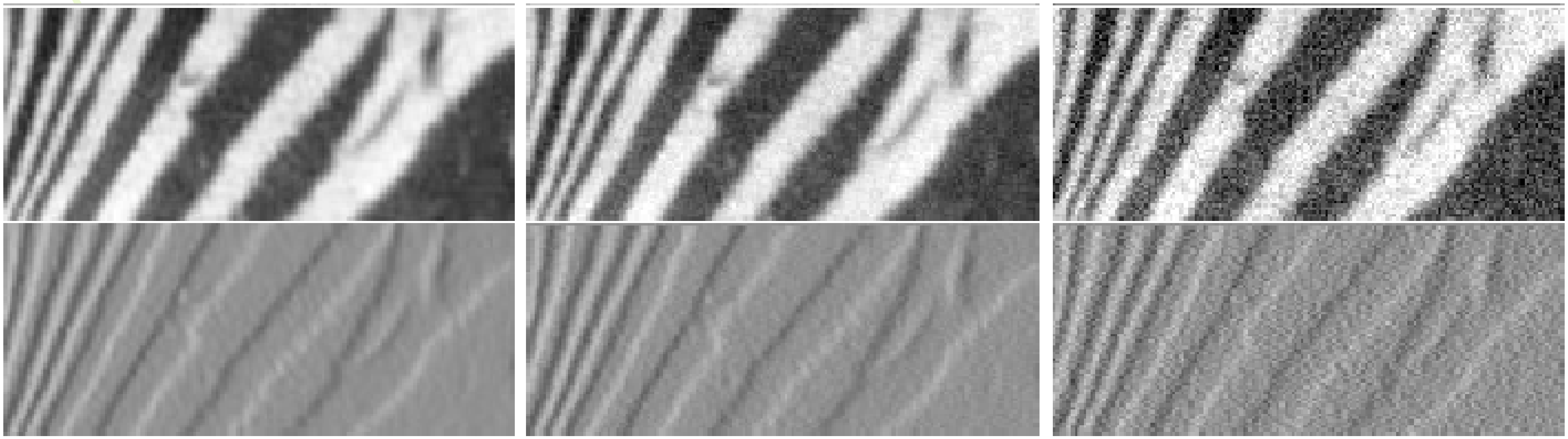
(c)

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

(d)

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

Finite differences responding to noise



Increasing noise ->
(this is zero mean additive gaussian noise)

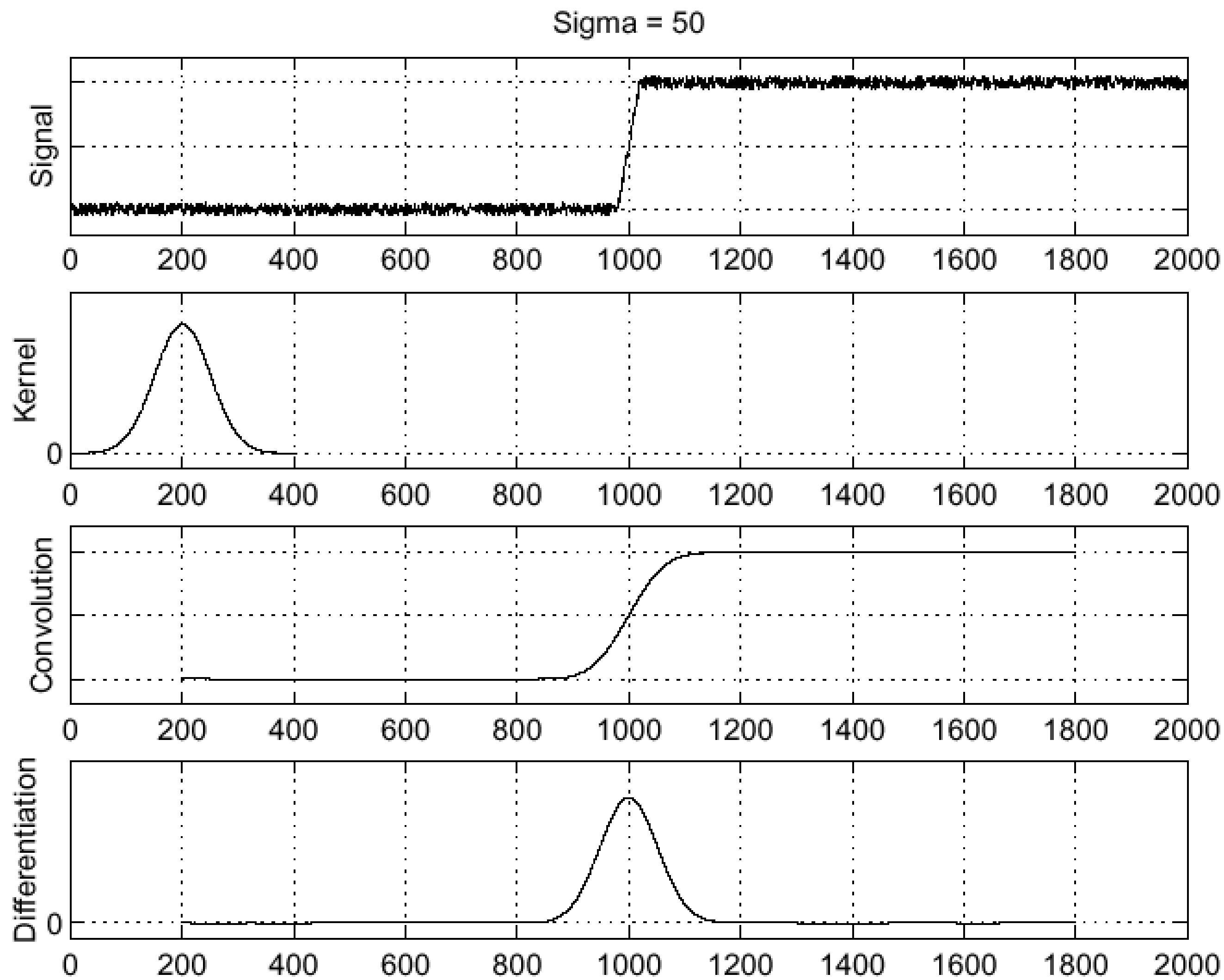
Solution: smooth first

f

h

$h \star f$

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



- Look for peaks in $\frac{\partial}{\partial x}(h \star f)$

Derivative theorem

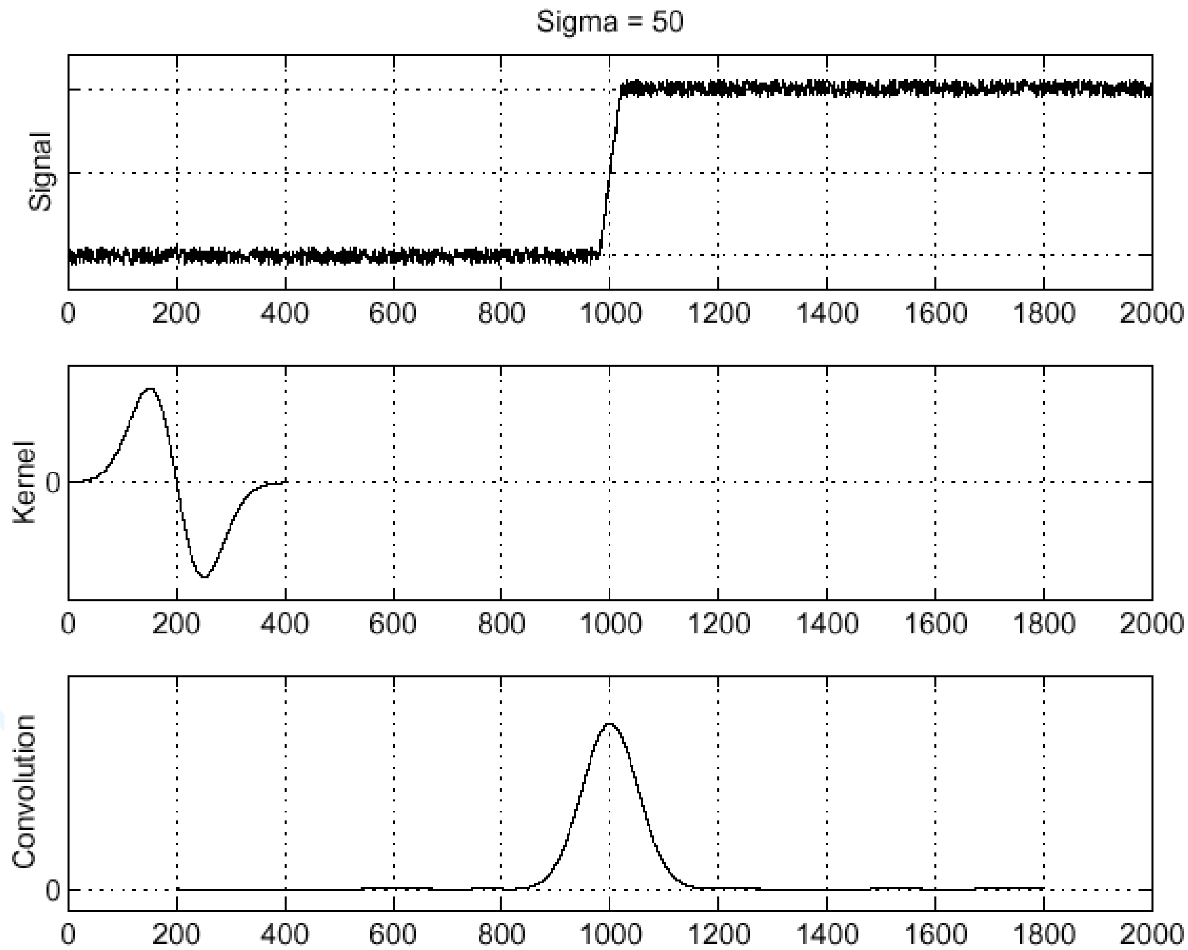
$$\frac{\partial}{\partial x}(h \star f) = (\frac{\partial}{\partial x}h) \star f$$

- This saves us one operation:

f

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

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



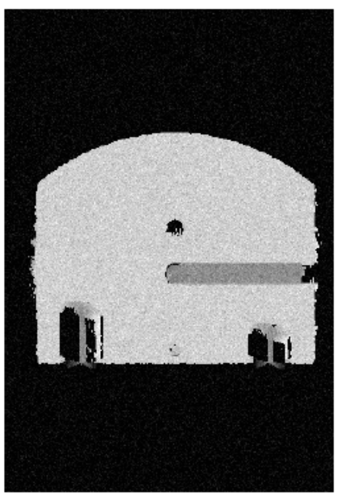
Results

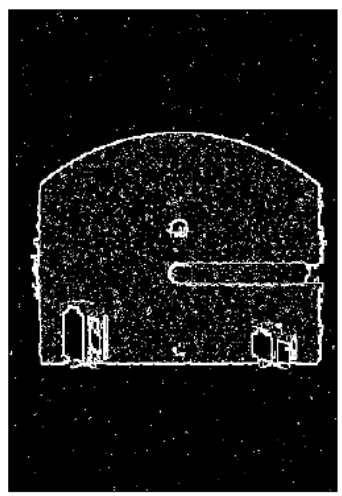
Original

Convolution with Sobel

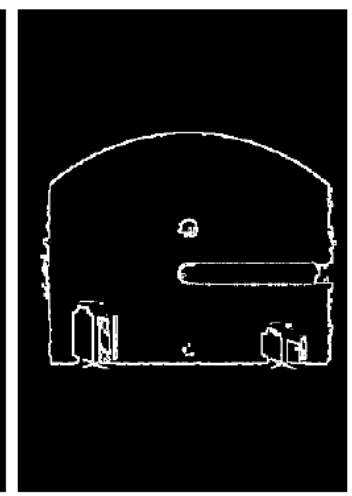
Thresholding (Value = 64)

Thresholding (Value = 96)



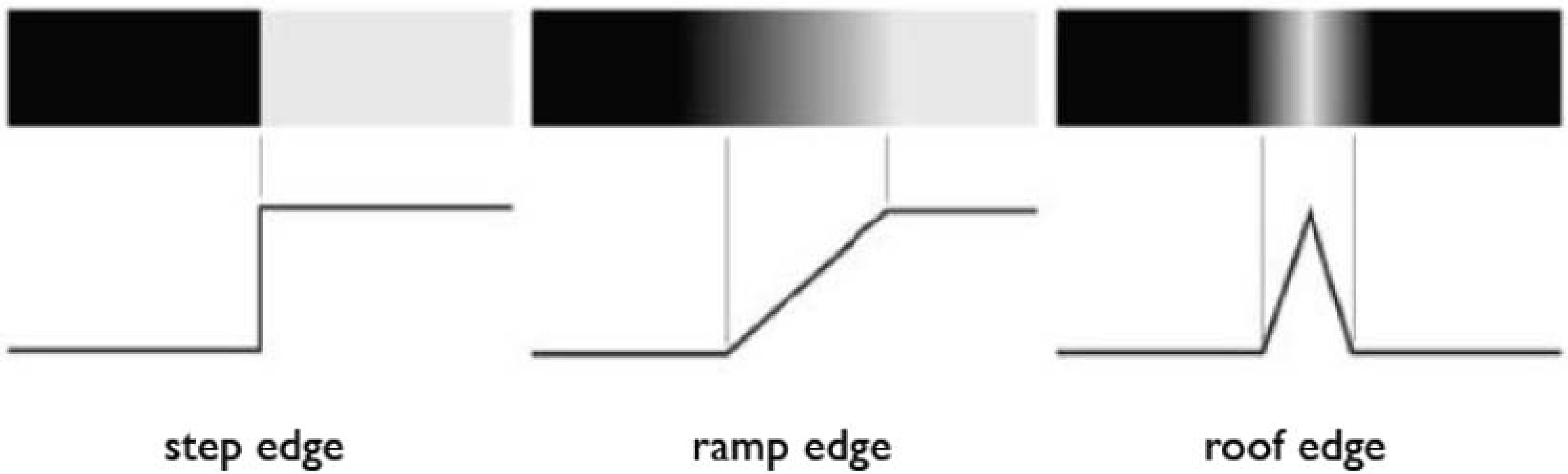


Without Gaussian



With Gaussian

Problems: Gradient Based Edges



Poor Localization
(Trigger response in
multiple adjacent pixels)

- Different response for different direction edges
- Thresholding value favors certain directions over others
  - Can miss oblique edges more than horizontal or vertical edges
  - False negatives

Second derivative zero

- How to find second derivative?
- $f(x+1, y) - 2f(x,y) + f(x-1,y)$
- In 2D
- What is an edge?
  - Look for zero crossings
  - With high contrast
  - Laplacian Kernel

0	−1	0
−1	4	−1
0	−1	0

−1	−1	−1
−1	8	−1
−1	−1	−1

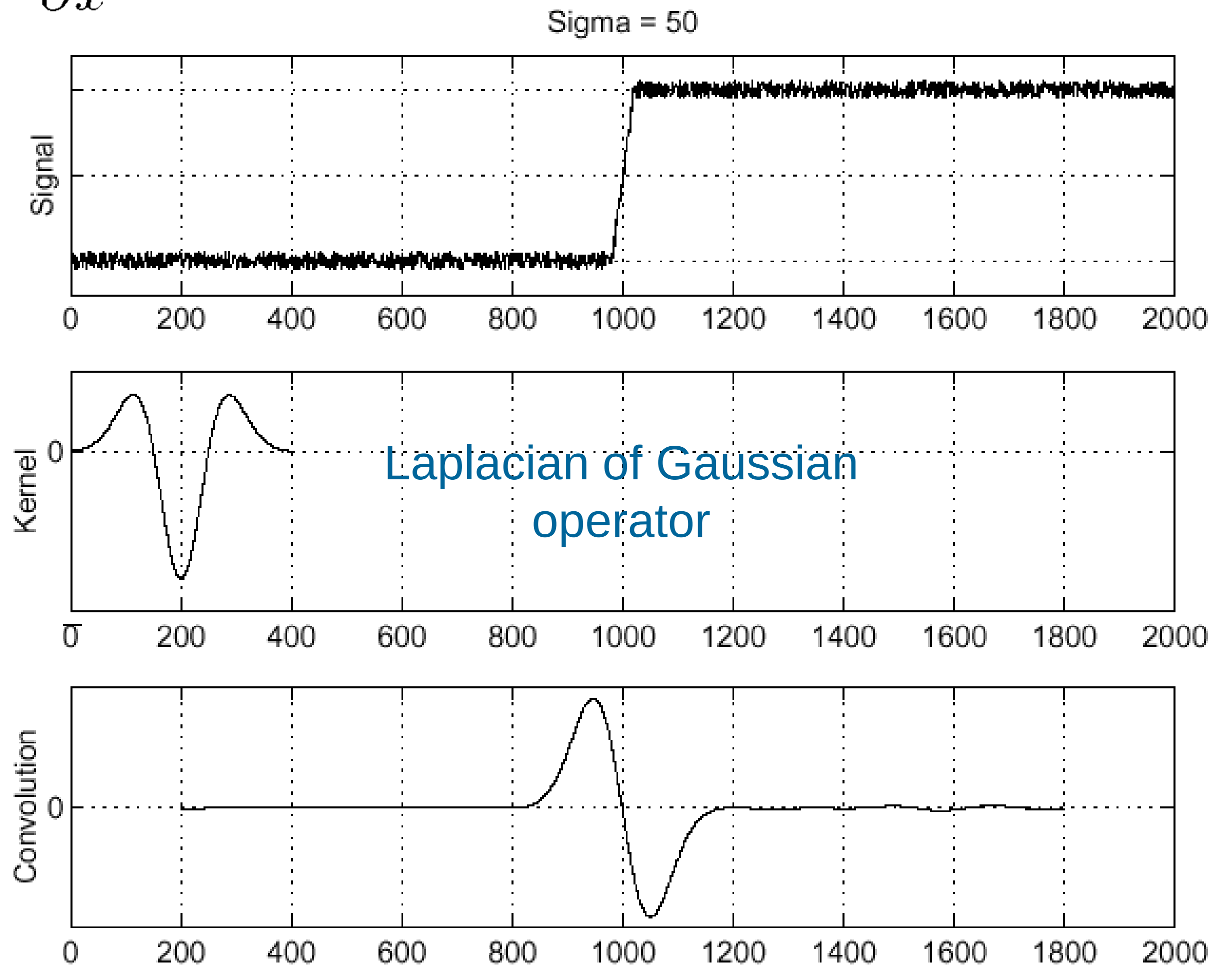
Laplacian of Gaussian

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

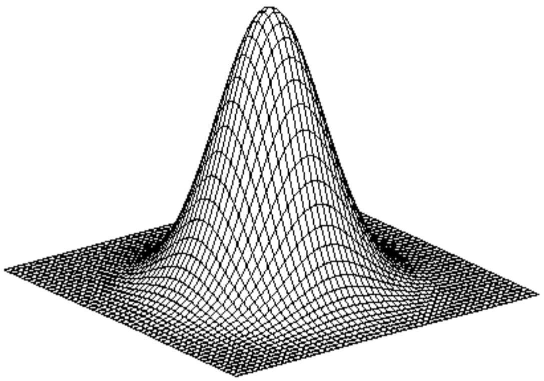
f

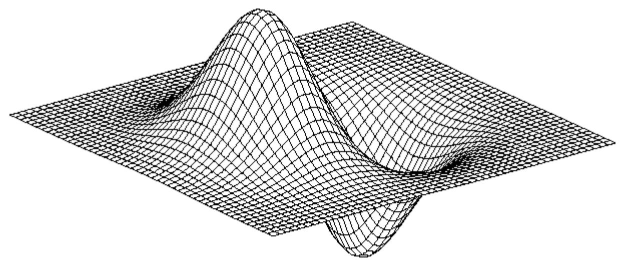
$\frac{\partial^2}{\partial x^2}h$

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

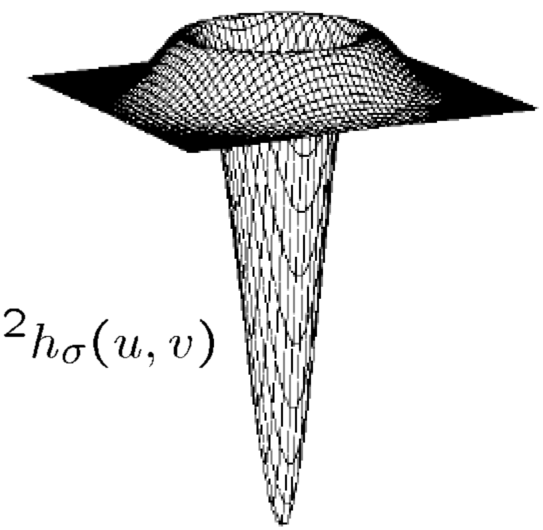


2D edge detection filters





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:

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

Edge detection by subtraction

original

Edge detection by subtraction

smoothed (5x5 Gaussian)

Edge detection by subtraction

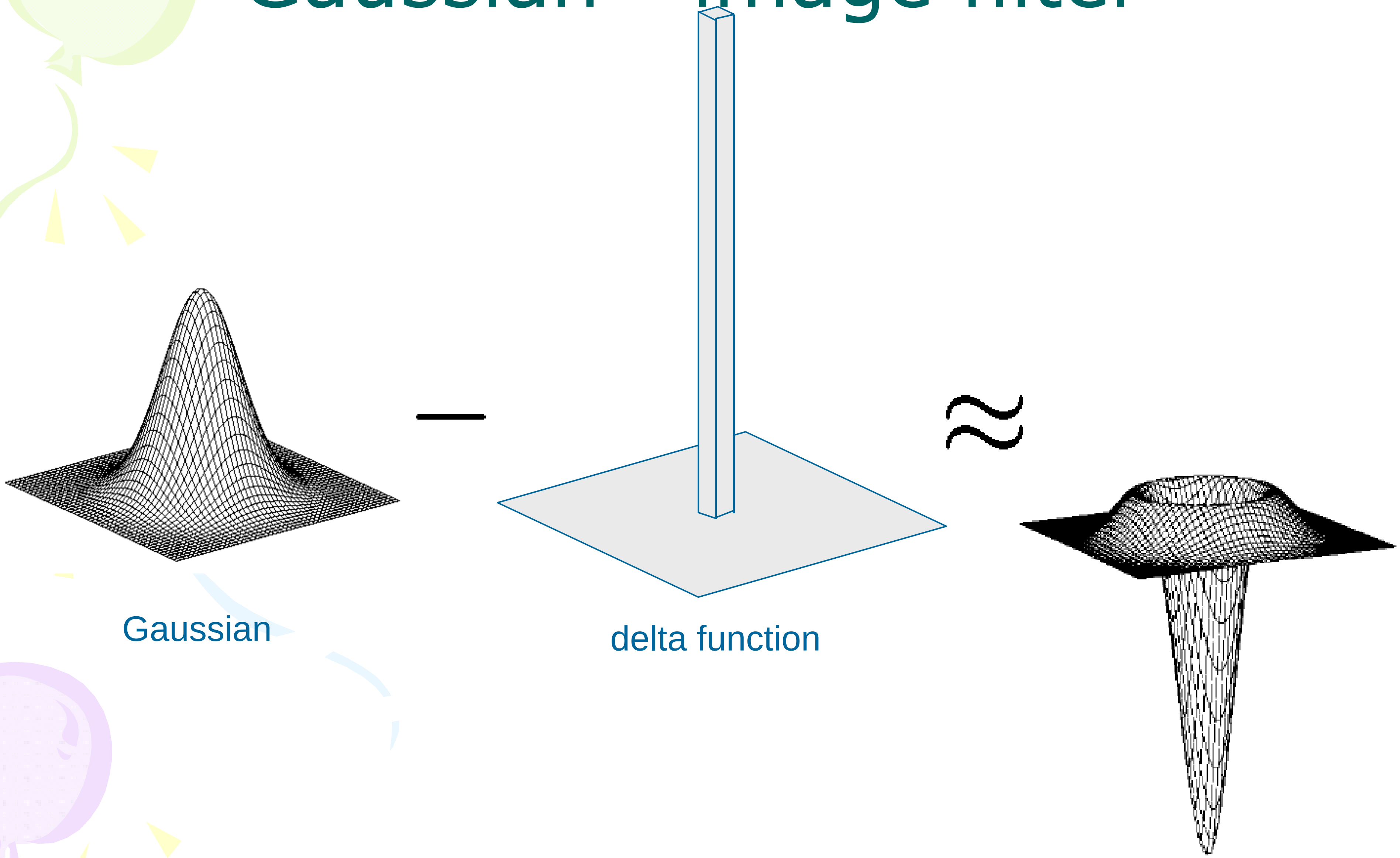
smoothed – original

(scaled by 4, offset +128)

Why does this work?

filter demo

Gaussian - image filter



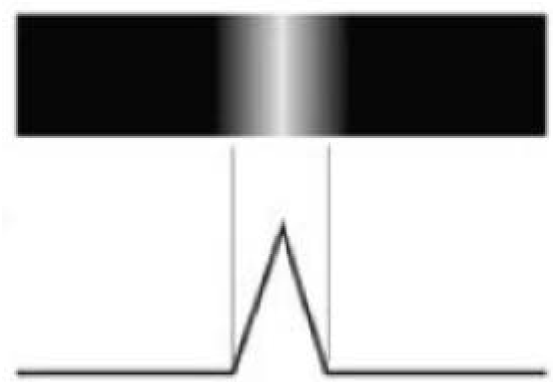
Gaussian

delta function

Laplacian of Gaussian

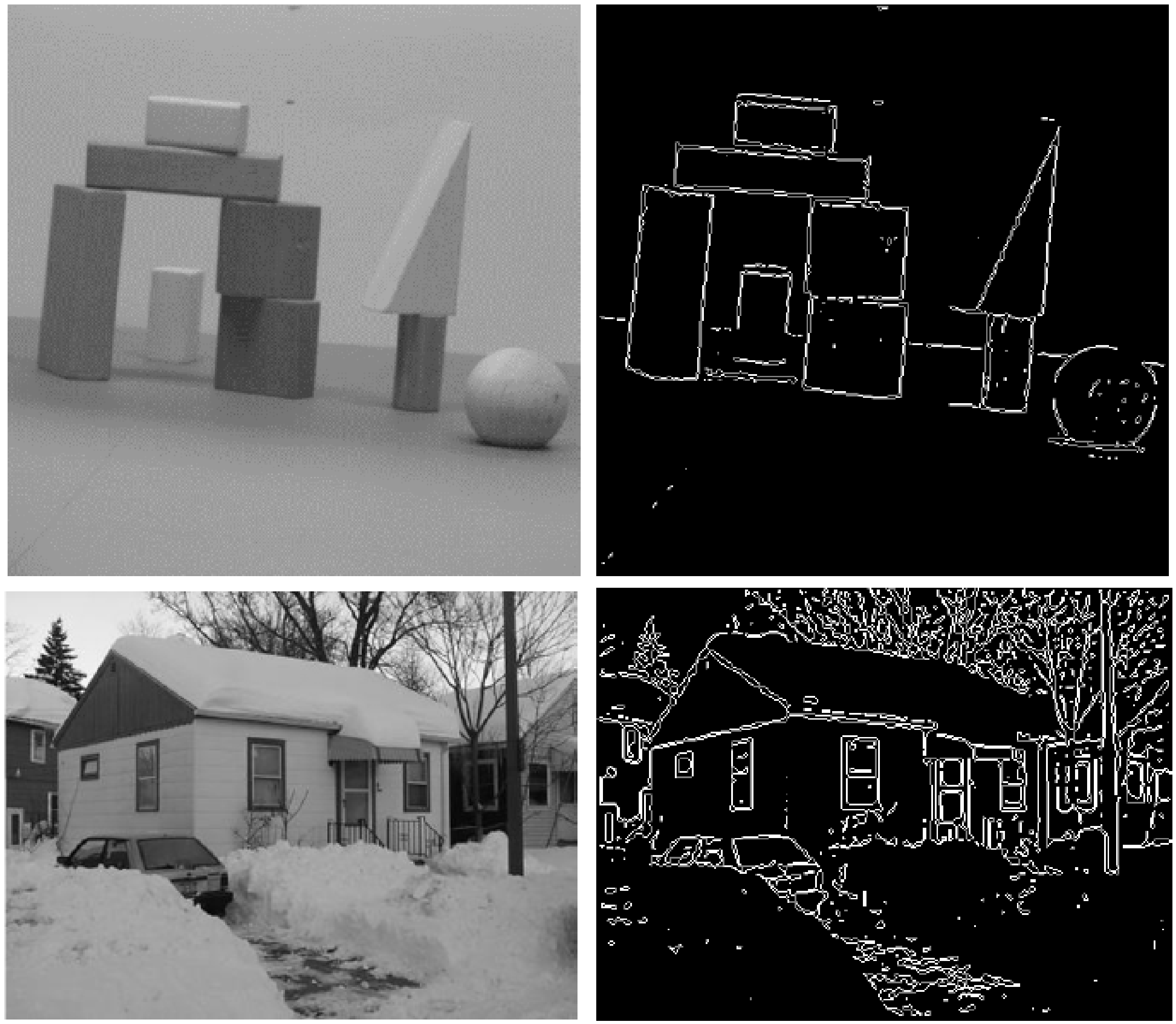
Pros and Cons

+ Good localizations due to zero crossings
+ Responds similarly to all different edge orientation
- Two zero crossings for roof edges
    - Spurious edges
    - False positives



roof edge

Examples



Optimal Edge Detection: Canny

- Assume:
  - Linear filtering
  - Additive Gaussian noise
- Edge detector should have:
  - Good Detection. Filter responds to edge, not noise.
  - Good Localization: detected edge near true edge.
  - Minimal Response: one per edge
- Detection/Localization trade-off
  - More smoothing improves detection
  - And hurts localization.

Canny Edge Detector

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

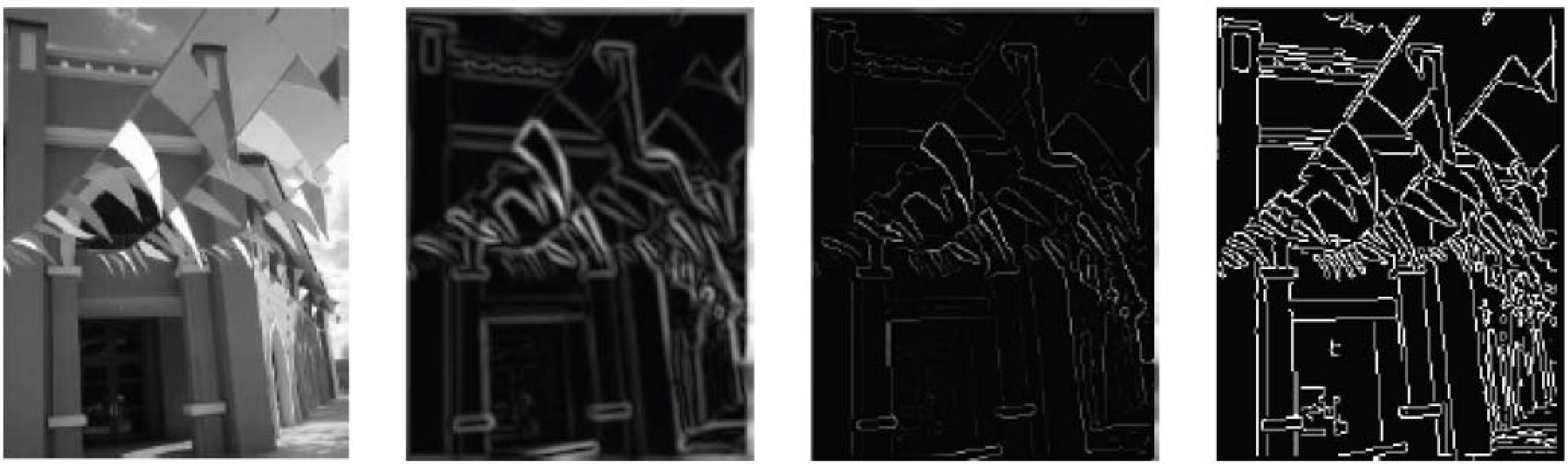
Non-Maxima Supression

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

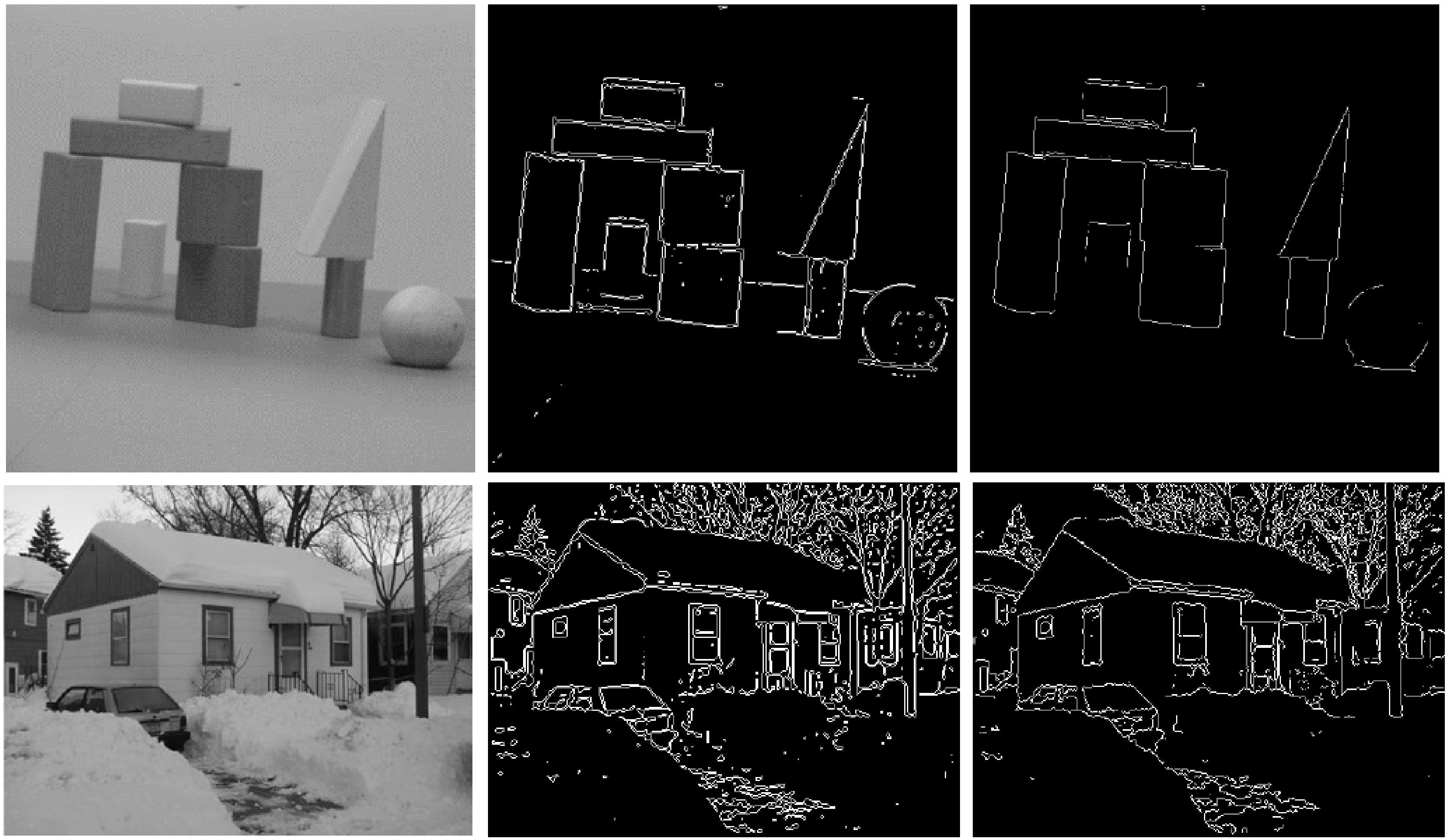
Hysteresis

- Avoid streaking near threshold value
- Define two thresholds – L , H
  - If less than L, not an edge
  - If greater than H, strong edge
  - If between L and H, weak edge
    - Analyze connectivity to mark is either non-edge or strong edge
    - Removes spurious edges

Four Steps



Comparison with Laplacian Based



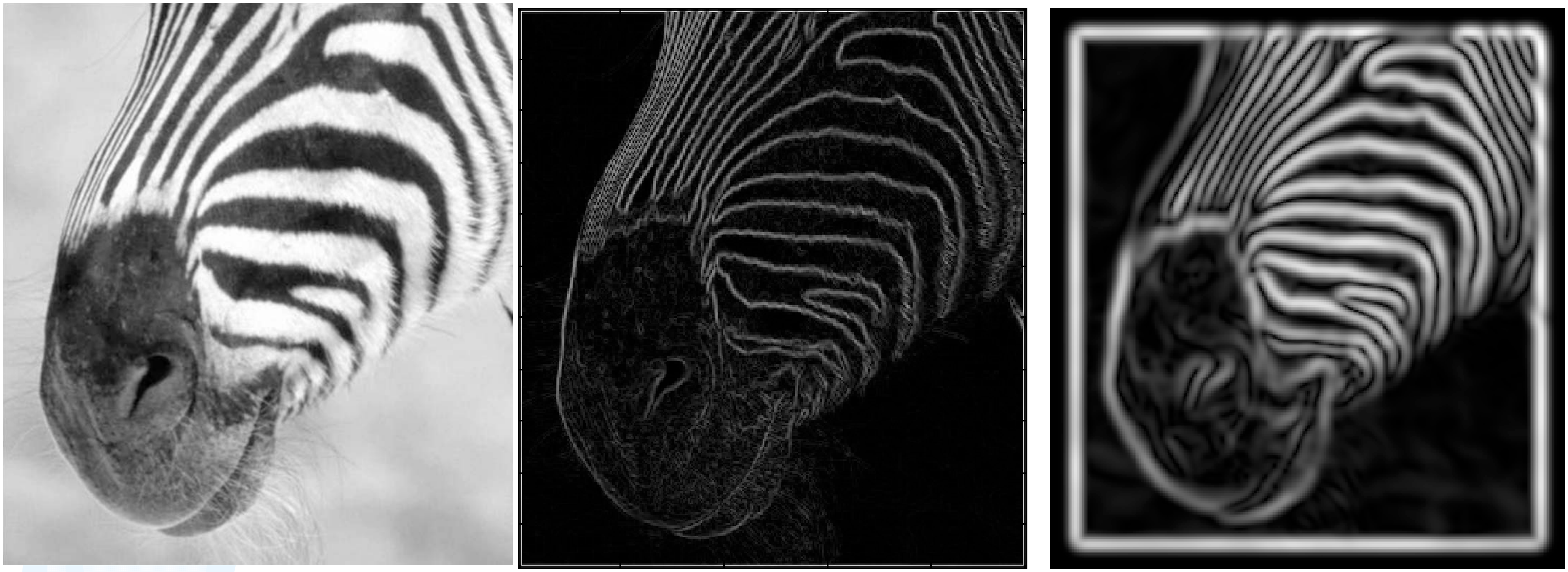
Effect of Smoothing kernel size)

original

Canny with $\sigma = 1$

Canny with $\sigma = 2$

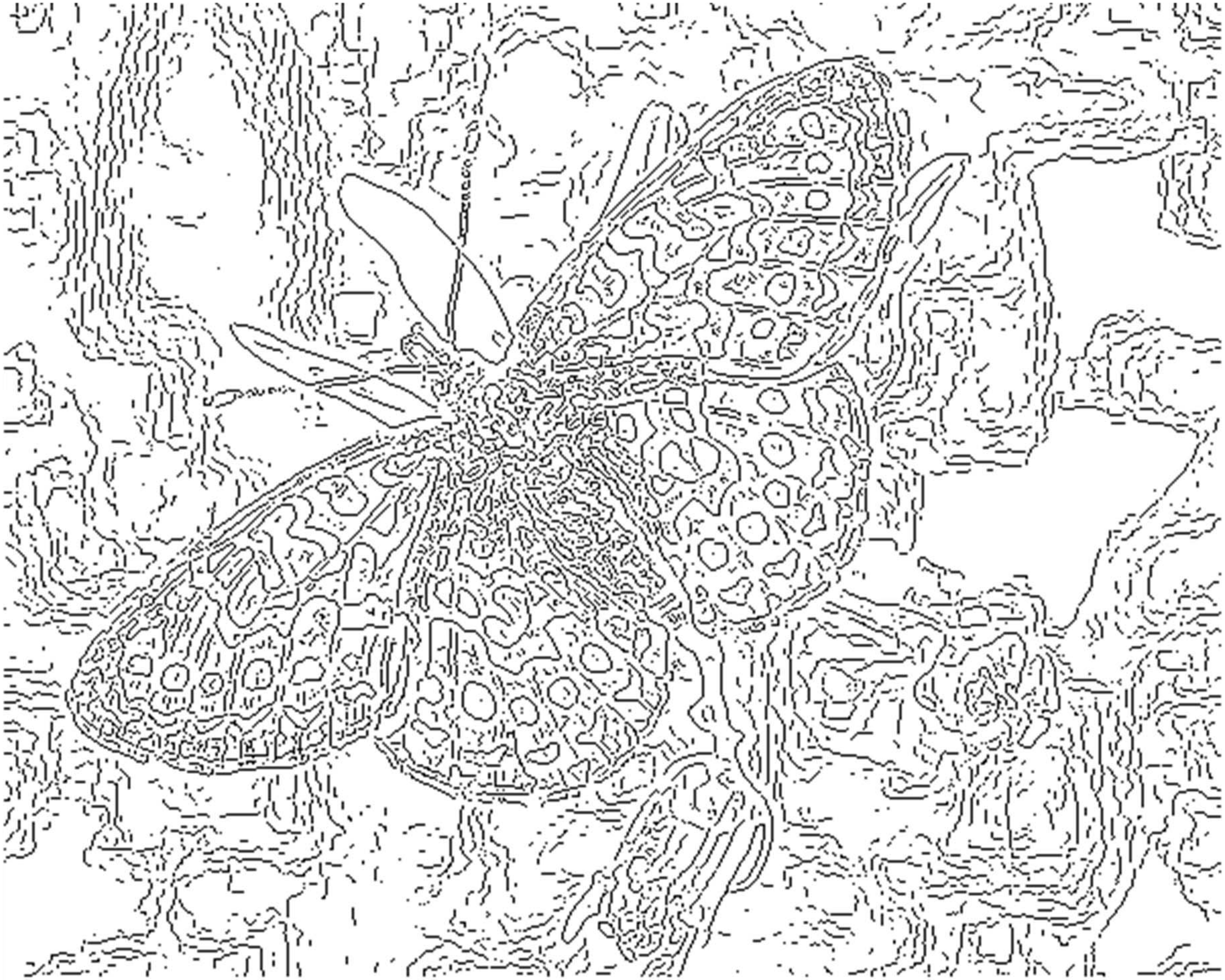
- The choice σ depends what is desired
  - large σ detects large scale edges
  - small σ detects fine features



Multi-resolution Edge Detection

- Smoothing
- Eliminates noise edges.
- Makes edges smoother.
- Removes fine detail.

(Forsyth & Ponce)

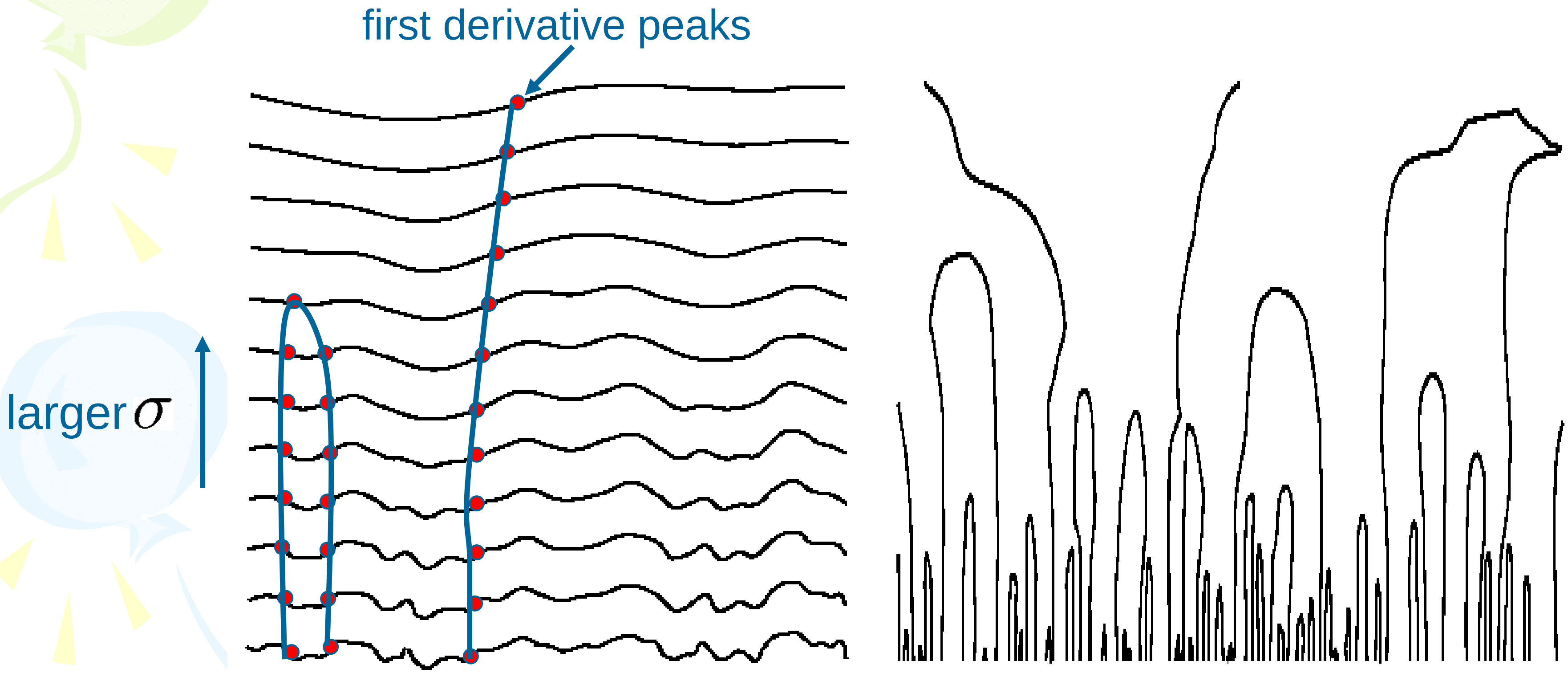


fine scale
high
threshold

coarse scale, high threshold

coarse
scale
low
threshold

Scale space (Witkin 83)



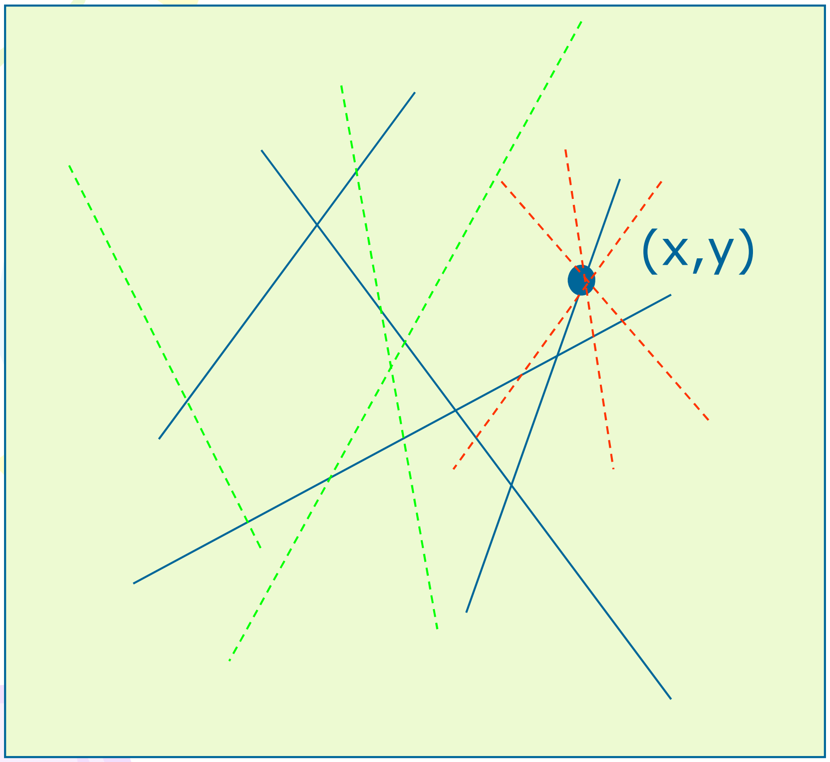
Gaussian filtered signal

- Properties of scale space (with 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

Identifying parametric edges

- Can we identify lines?
- Can we identify curves?
- More general
  - Can we identify circles/ellipses?
- Voting scheme called Hough Transform

Hough Transform



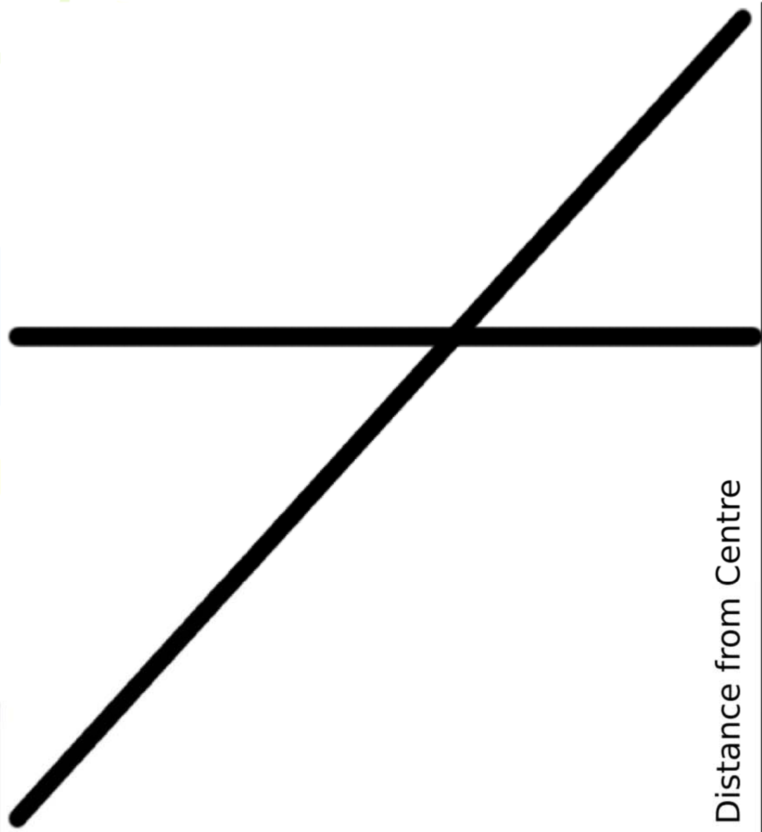
- Only a few lines can pass through (x,y)
  - $mx+b$
- Consider (m,b) space
- Red lines are given by a line in that space
  - $b = y - mx$
- Each point defines a line in the Hough space
- Each line defines a point (since same m,b)

How to identify lines?

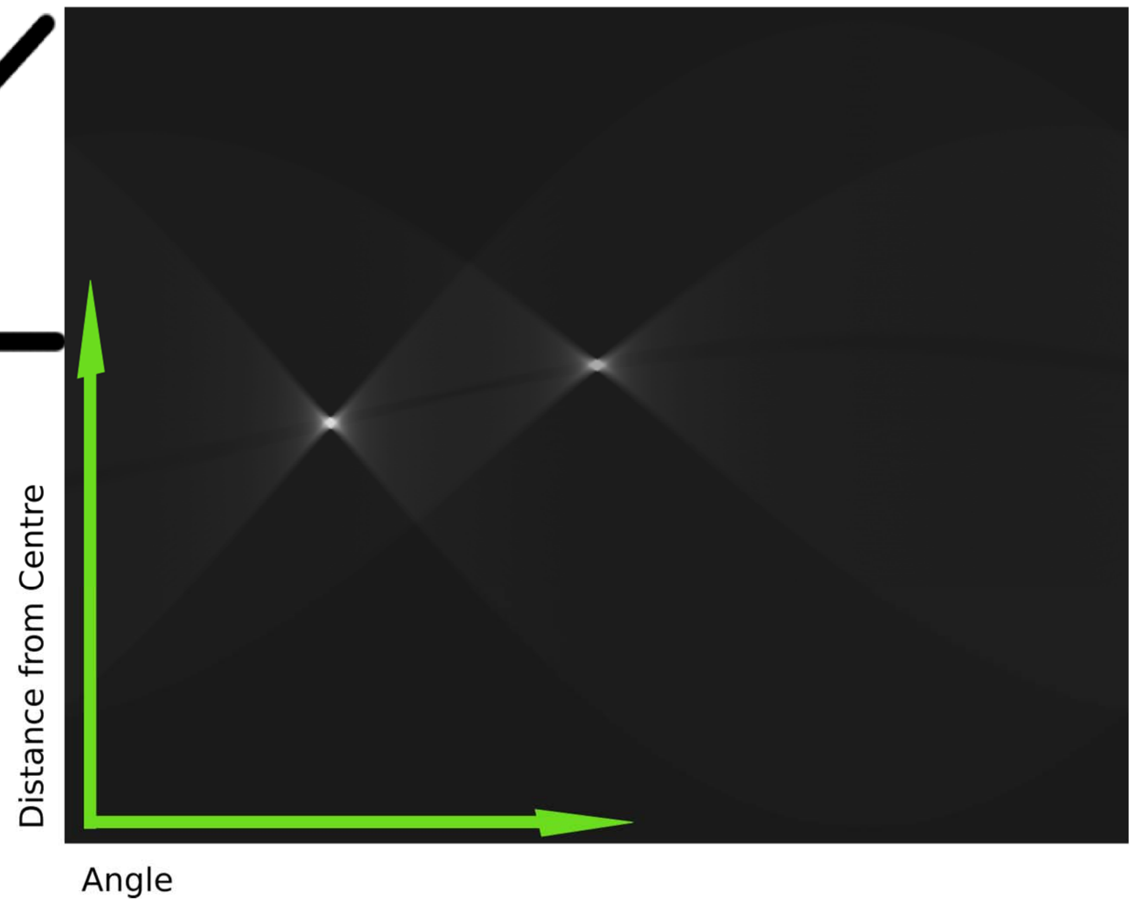
- For each edge point
  - Add intensity to the corresponding line in Hough space
- Each edge point votes on the possible lines through them
- If a line exists in the image space, that point in Hough space will get many votes and hence high intensity
- Find maxima in Hough space
- Find lines by equations y – mx+b

Example

Input Image

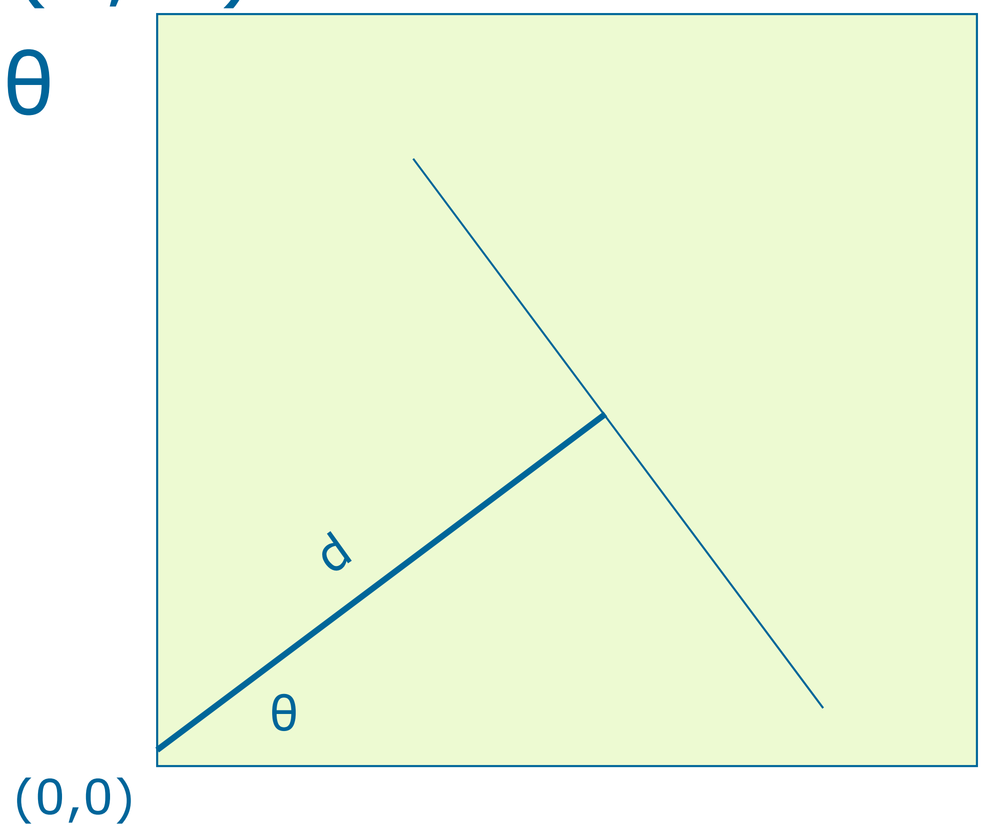


Rendering of Transform Results



Problem with (m,b) space

- Vertical lines have infinite m
- Polar notation of (d, θ)
- $d = x\cos\theta + y\sin\theta$



Basic Hough Transform

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

   for θ = 0 to 180

   d = xcosθ + ysinθ

   H[d, θ] += 1

3. Find the value(s) of (d, θ) for max H[d, θ]

A similar procedure can be used for identifying circles, squares, or other shape with appropriate change in Hough parameterization.