

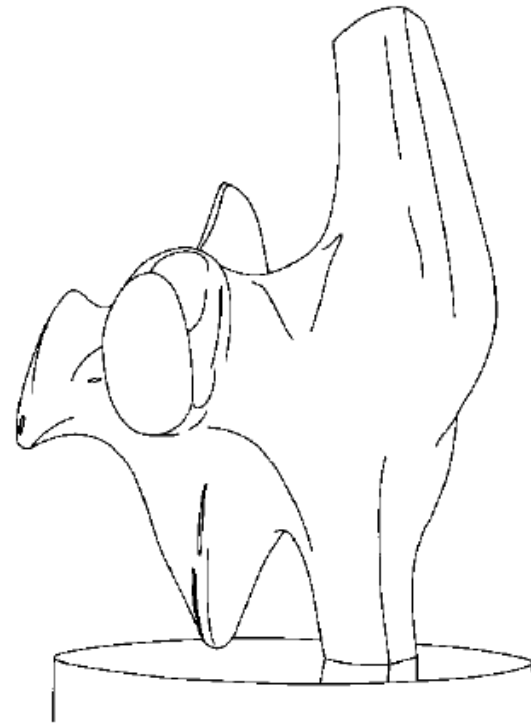
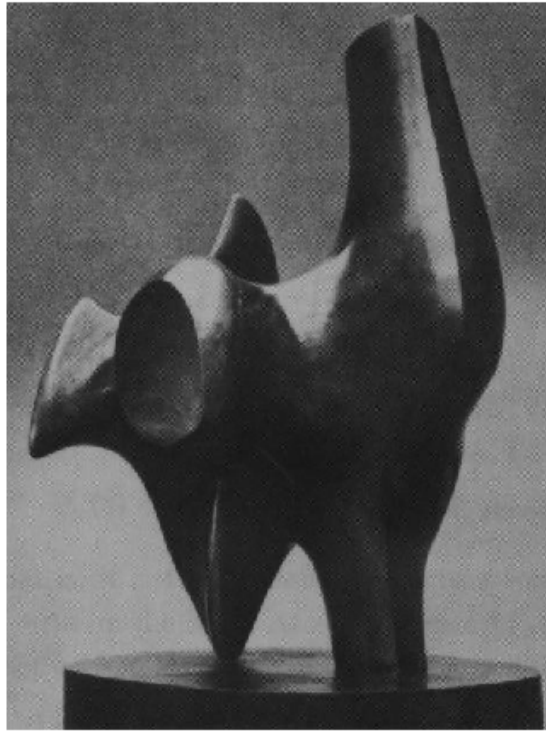


# Other Linear Filters

CS 211A

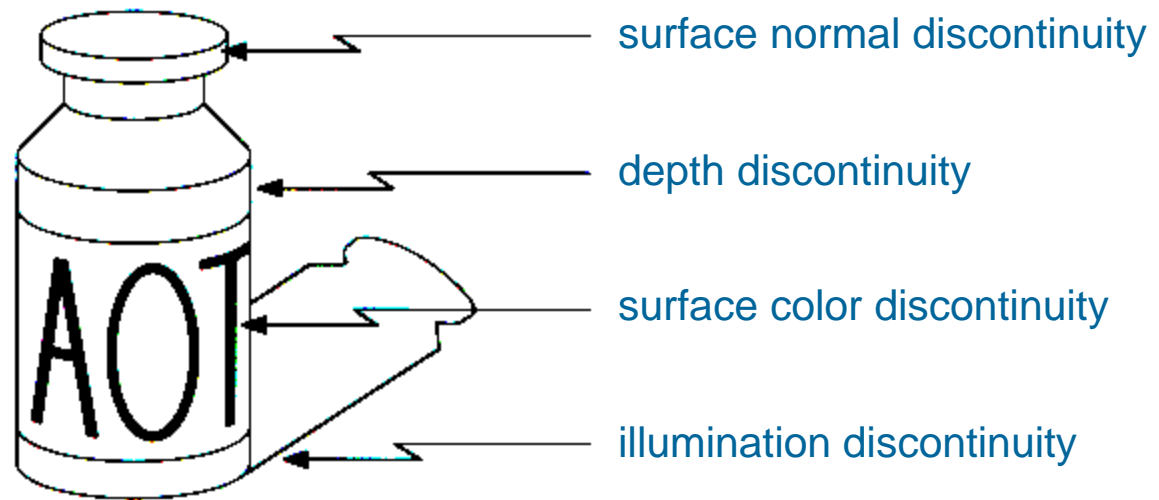
Slides from Cornelia Fermüller and Marc Pollefeys

# 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



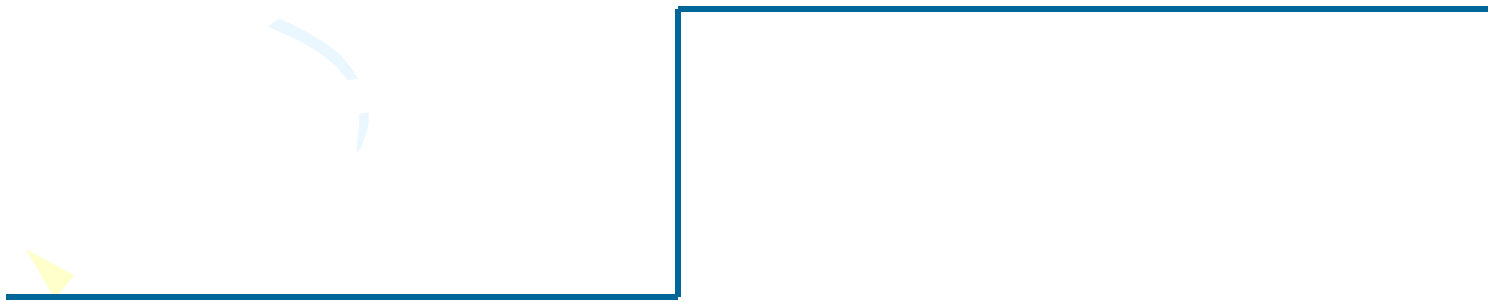
# Edge detection

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  $2^{\text{nd}}$  derivative is zero.

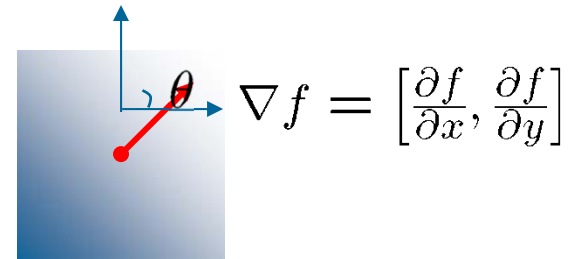
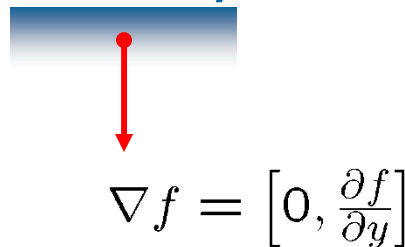
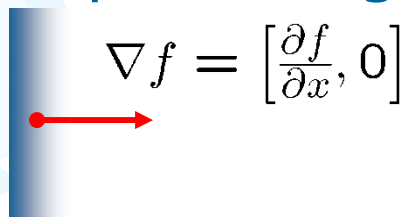


# Image gradient

- 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



- The gradient direction is given by:

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

– Perpendicular to the edge

- The *edge strength* is given by the magnitude

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

Three balloons in green, blue, and purple are positioned on the left side of the slide. Each balloon has a string and several small yellow triangular flags attached to it. The green balloon is at the top, the blue one is in the middle, and the purple one is at the bottom. They are arranged in a slightly diagonal line from top-left to bottom-left.

# 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

$$\frac{1}{8} \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

$s_x$

$$\frac{1}{8} \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

$s_y$

- 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

$\Delta_1$

0 1  
-1 0

$\Delta_2$

1 0  
0 -1

(a)

$\Delta_1$

-1 0 1  
-1 0 1  
-1 0 1

$\Delta_2$

1 1 1  
0 0 0  
-1 -1 -1

(b)

$\Delta_1$

-1 0 1  
-2 0 2  
-1 0 1

$\Delta_2$

1 2 1  
0 0 0  
-1 -2 -1

(c)

$\Delta_1$

-3 -1 1 3  
-3 -1 1 3  
-3 -1 1 3  
-3 -1 1 3

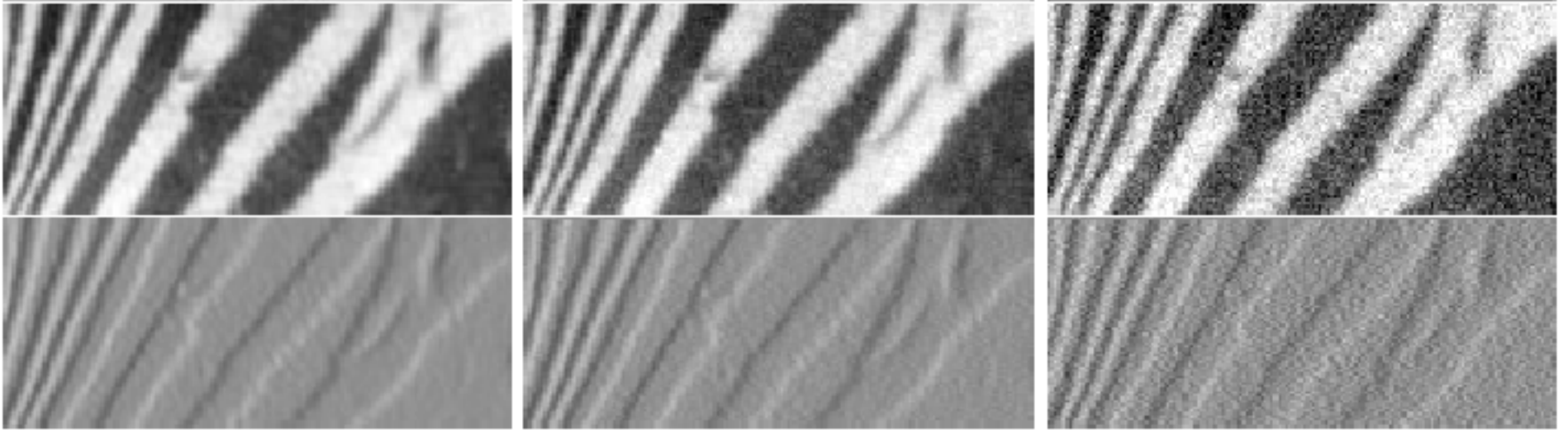
$\Delta_2$

3 3 3 3  
1 1 1 1  
-1 -1 -1 -1  
-3 -3 -3 -3

(d)

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

# Finite differences responding to noise

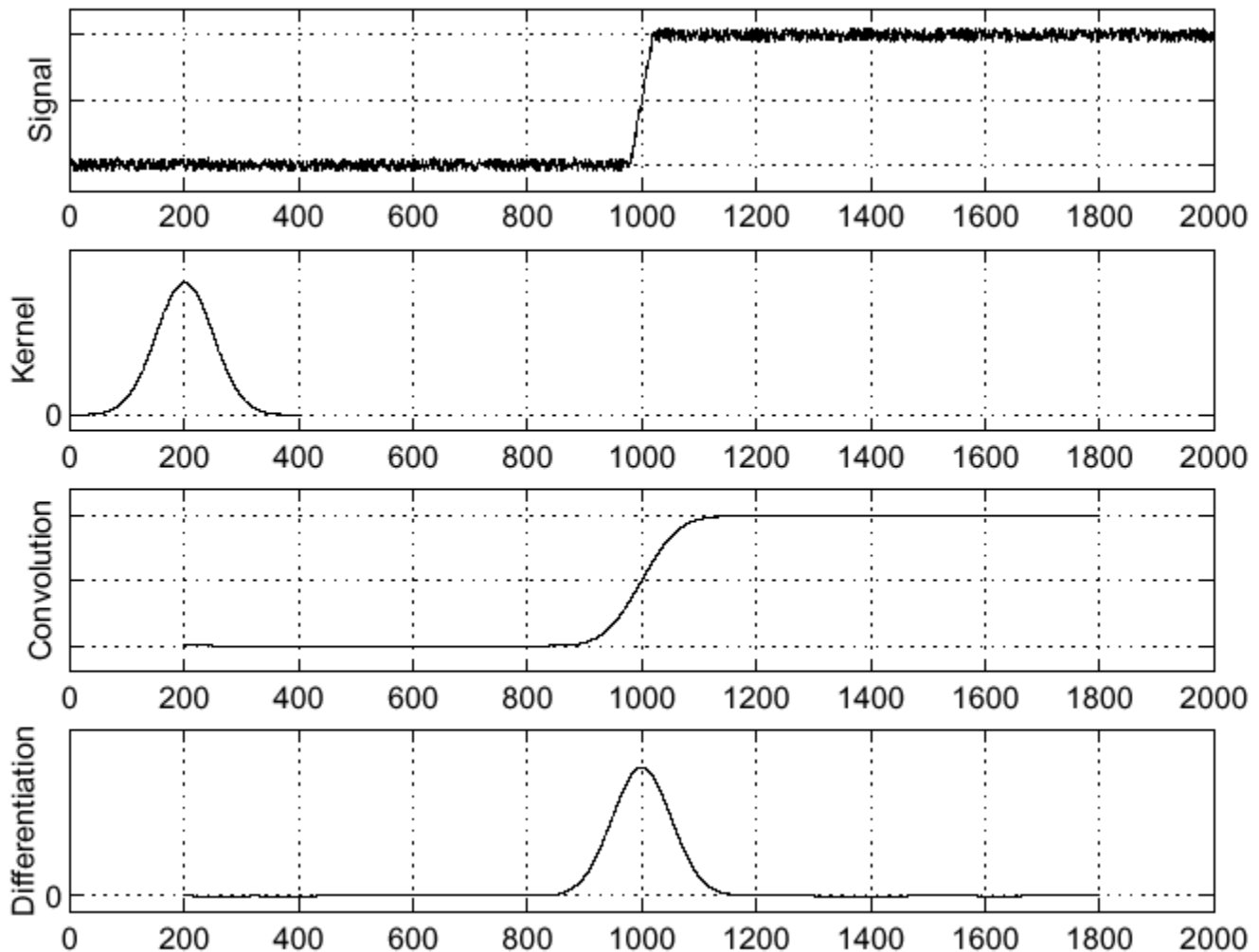


Increasing noise ->

(this is zero mean additive gaussian noise)

# Solution: smooth first

Sigma = 50



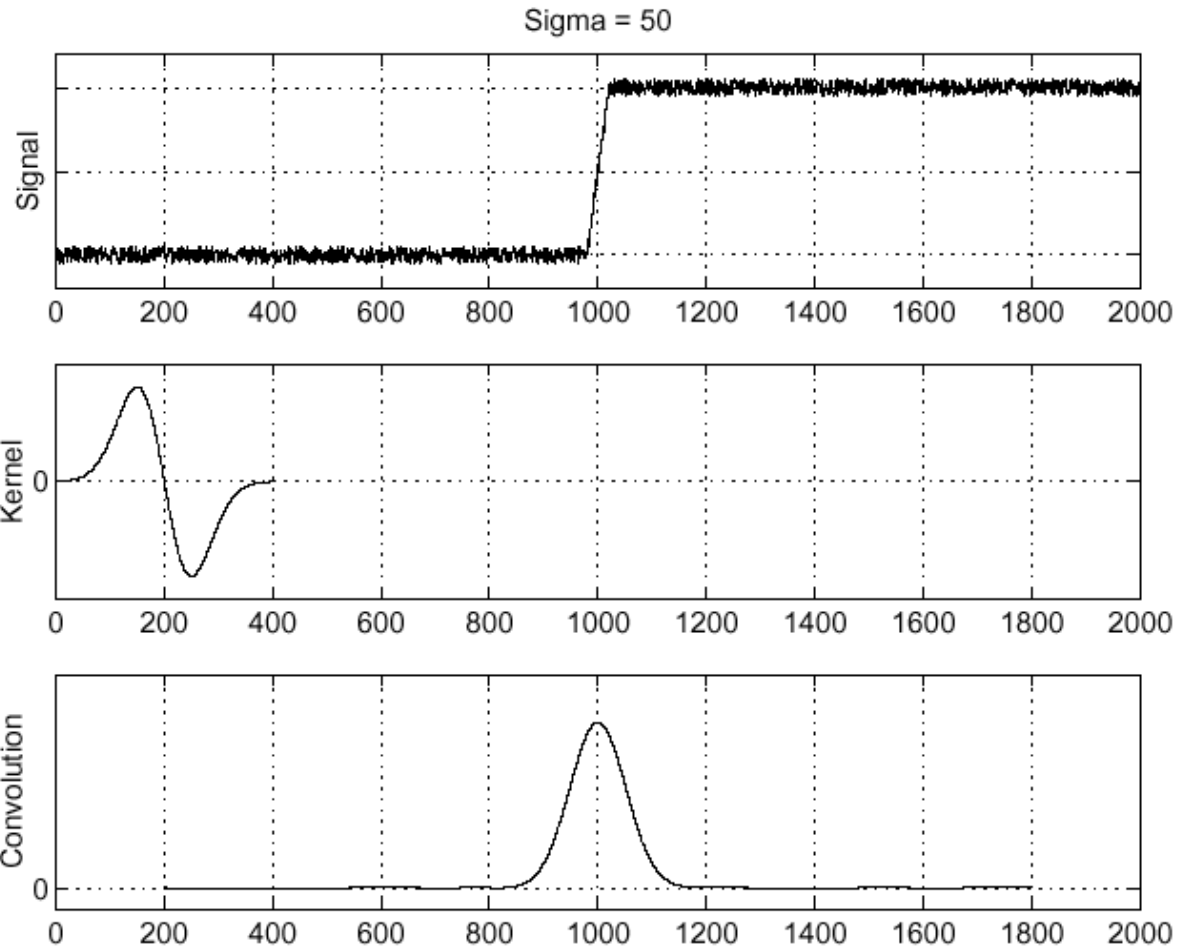
- Look for peaks in

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

# Derivative theorem

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

- This saves us one operation:



# Results



Original



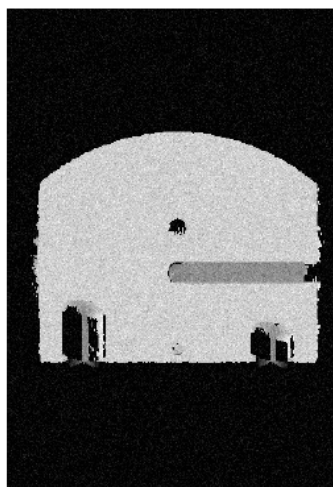
Convolution  
with Sobel



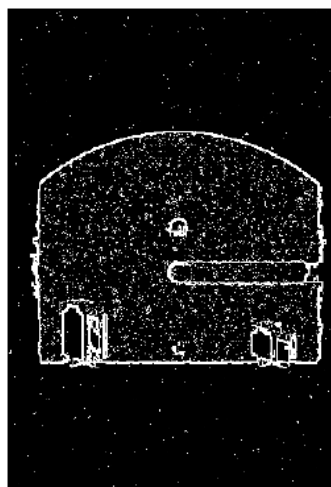
Thresholding  
(Value = 64)



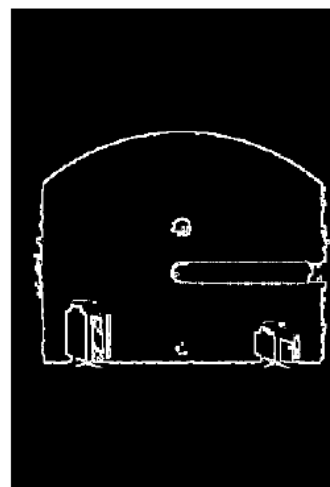
Thresholding  
(Value = 96)



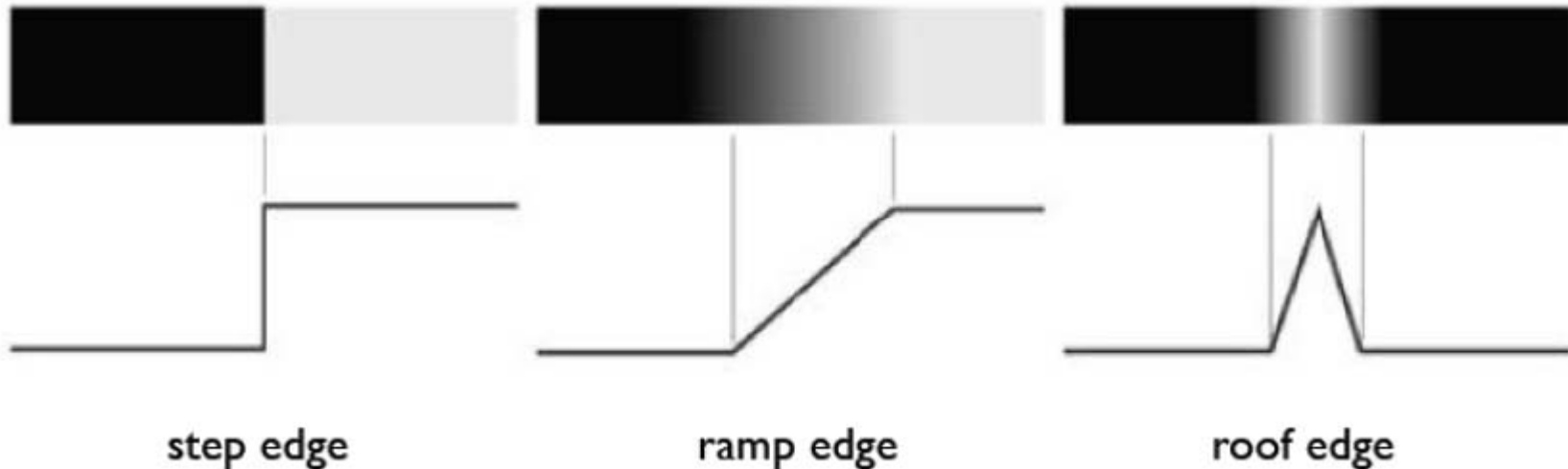
Without Gaussian



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

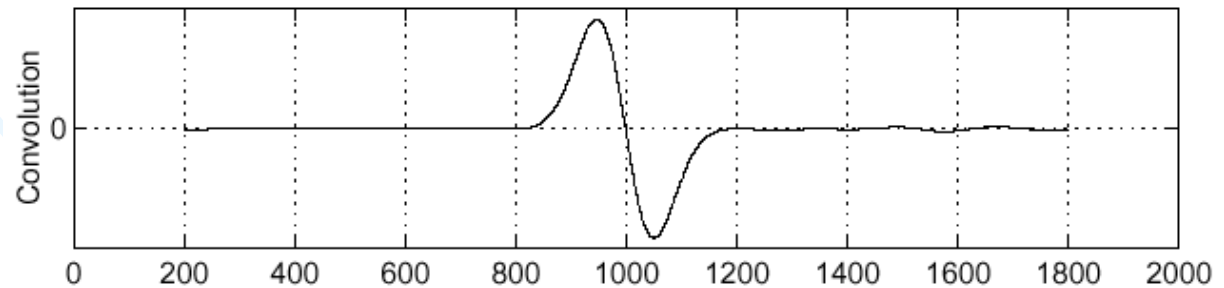
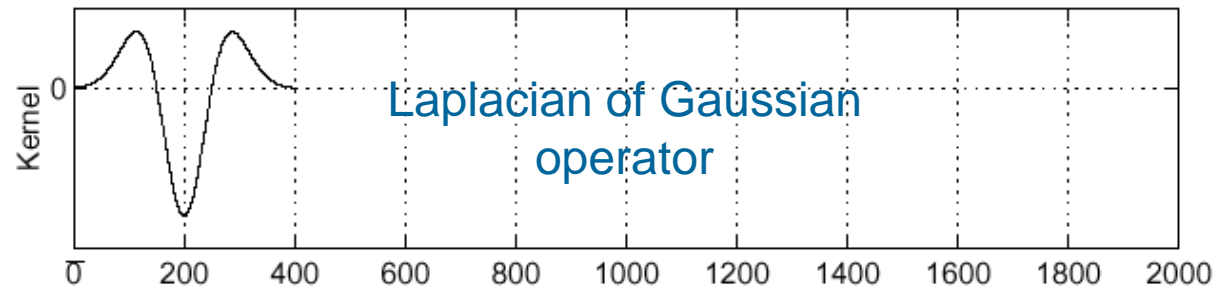
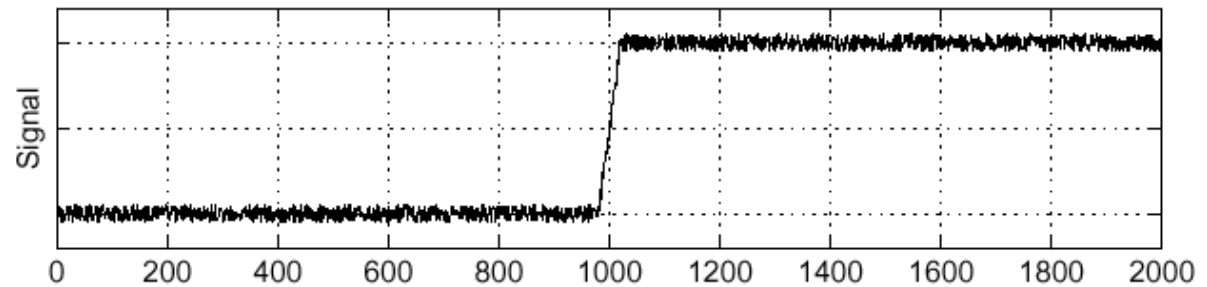
- Consider  $\frac{\partial^2}{\partial x^2}(h \star f)$

$f$

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

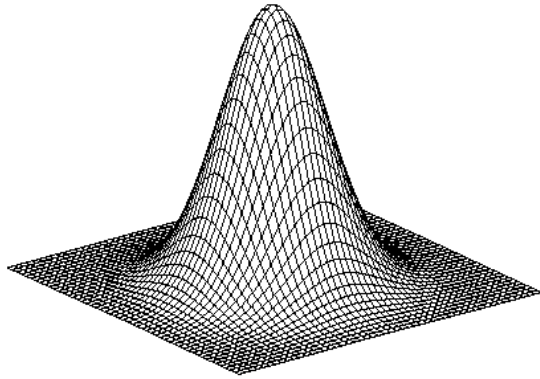
$(\frac{\partial^2}{\partial x^2}h) \star f$

Sigma = 50



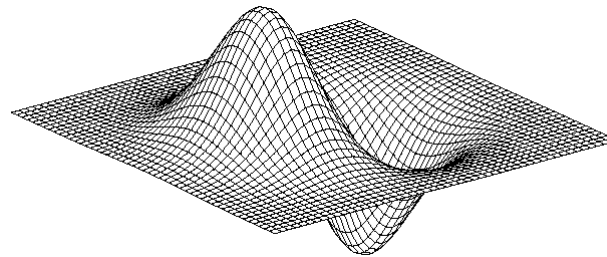


# 2D edge detection filters



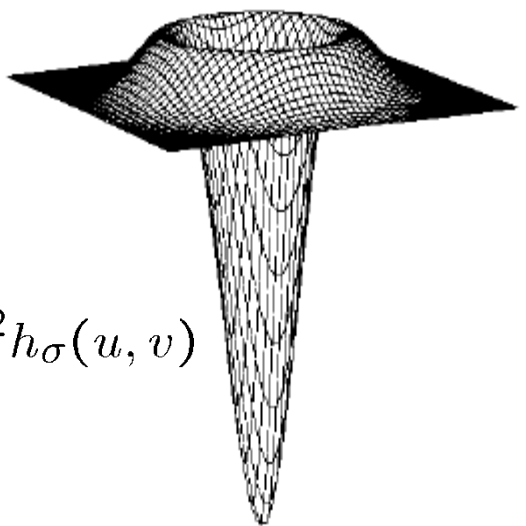
Gaussian

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



derivative of Gaussian

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



Laplacian of Gaussian

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

•  $\nabla^2$

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



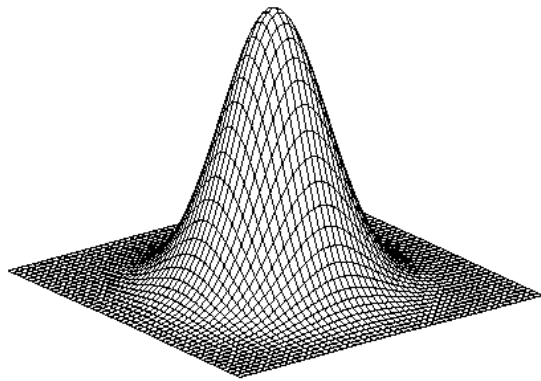
smoothed – original

(scaled by 4, offset +128)

Why does  
this work?

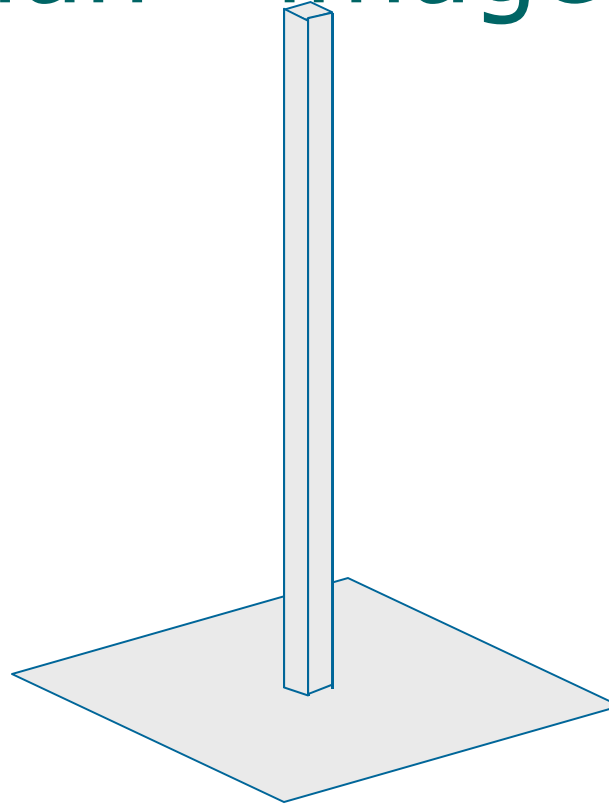
filter demo

# Gaussian - image filter



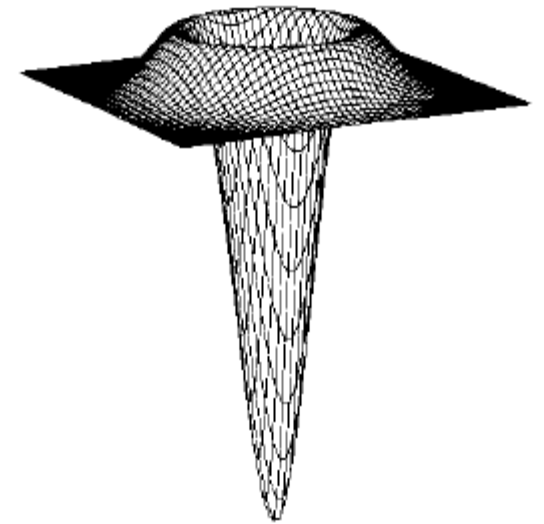
Gaussian

—



delta function

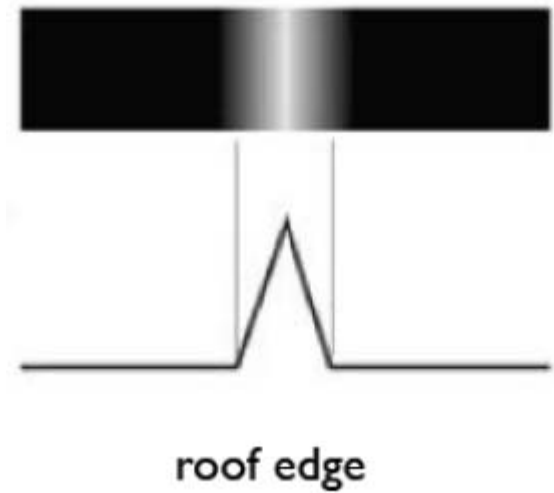
$\approx$



Laplacian of Gaussian

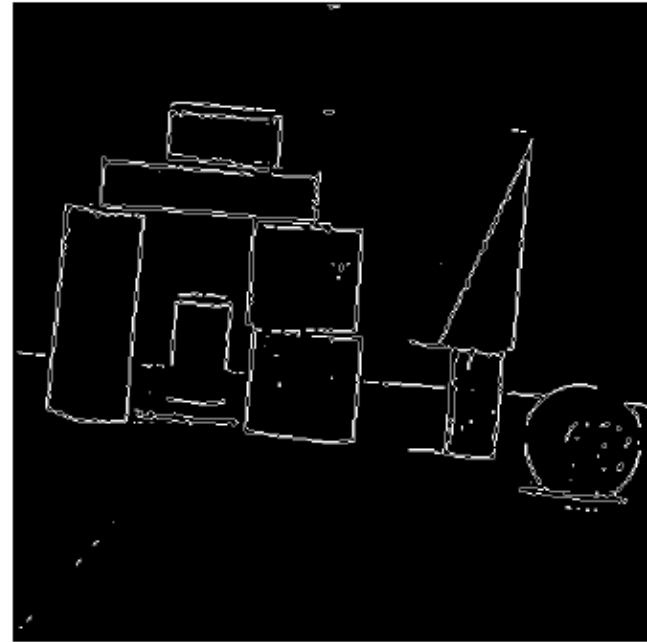
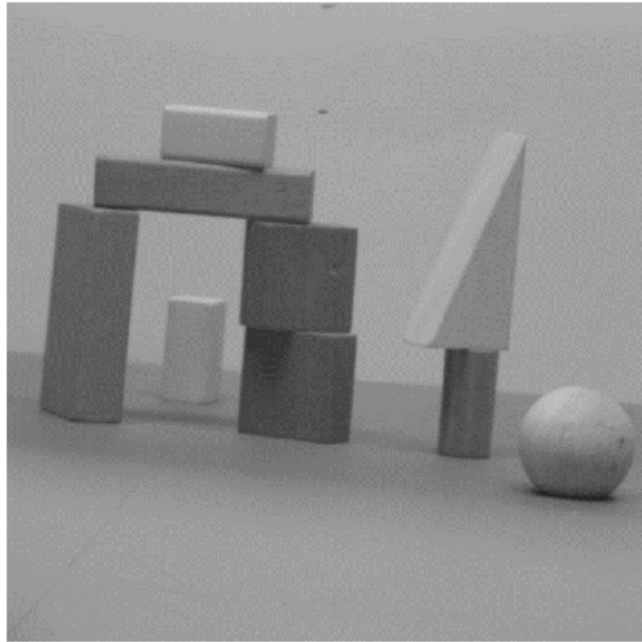
# Pros and Cons

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





# 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
- 

Three balloons in green, blue, and purple are positioned on the left side of the slide, each with yellow triangular streamers. The green balloon is at the top, the blue one in the middle, and the purple one at the bottom.

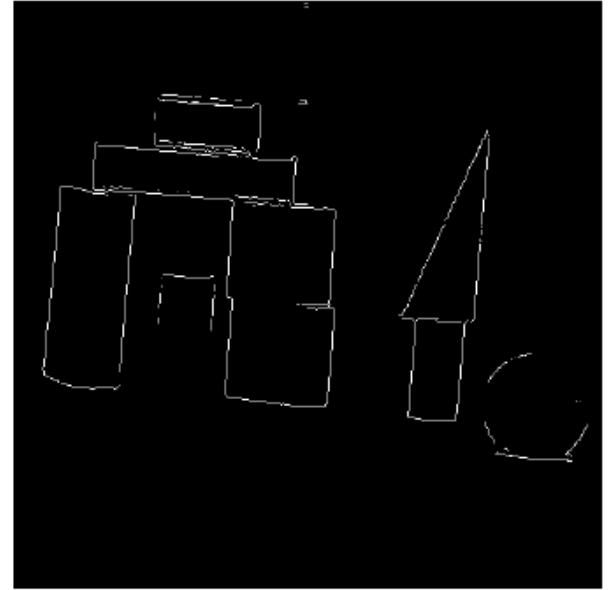
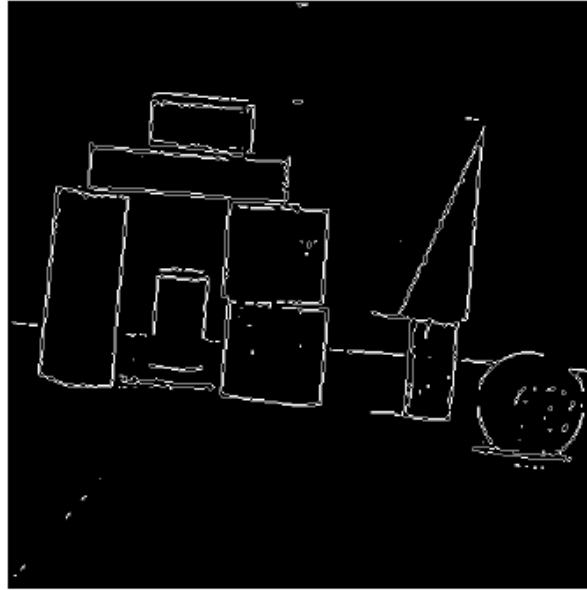
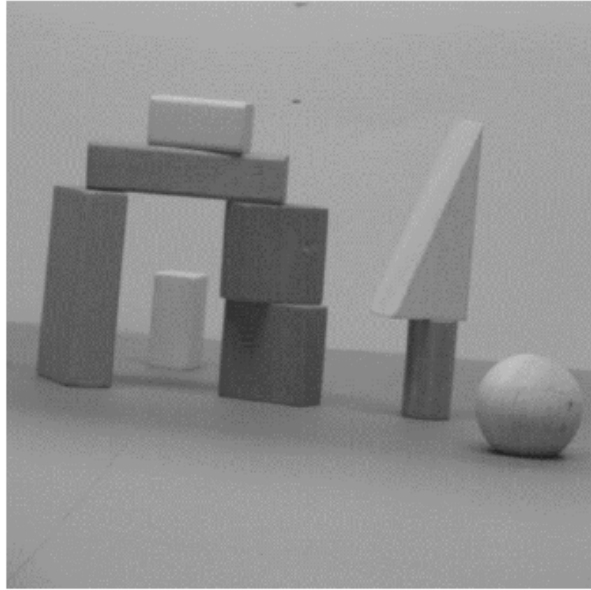
# Hysteresis

- 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



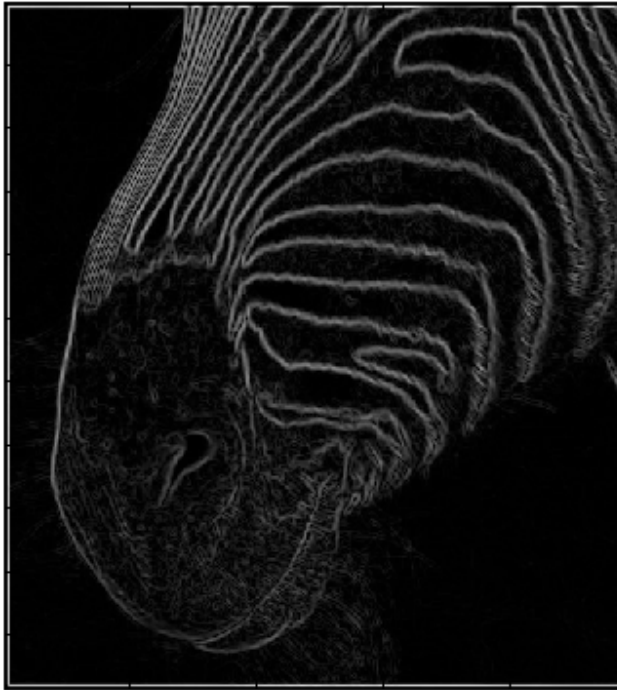
Canny with  $\sigma = 1$



Canny with  $\sigma = 2$

- The choice of  $\sigma$  depends on what is desired
  - large  $\sigma$  detects large scale edges
  - small  $\sigma$  detects fine features





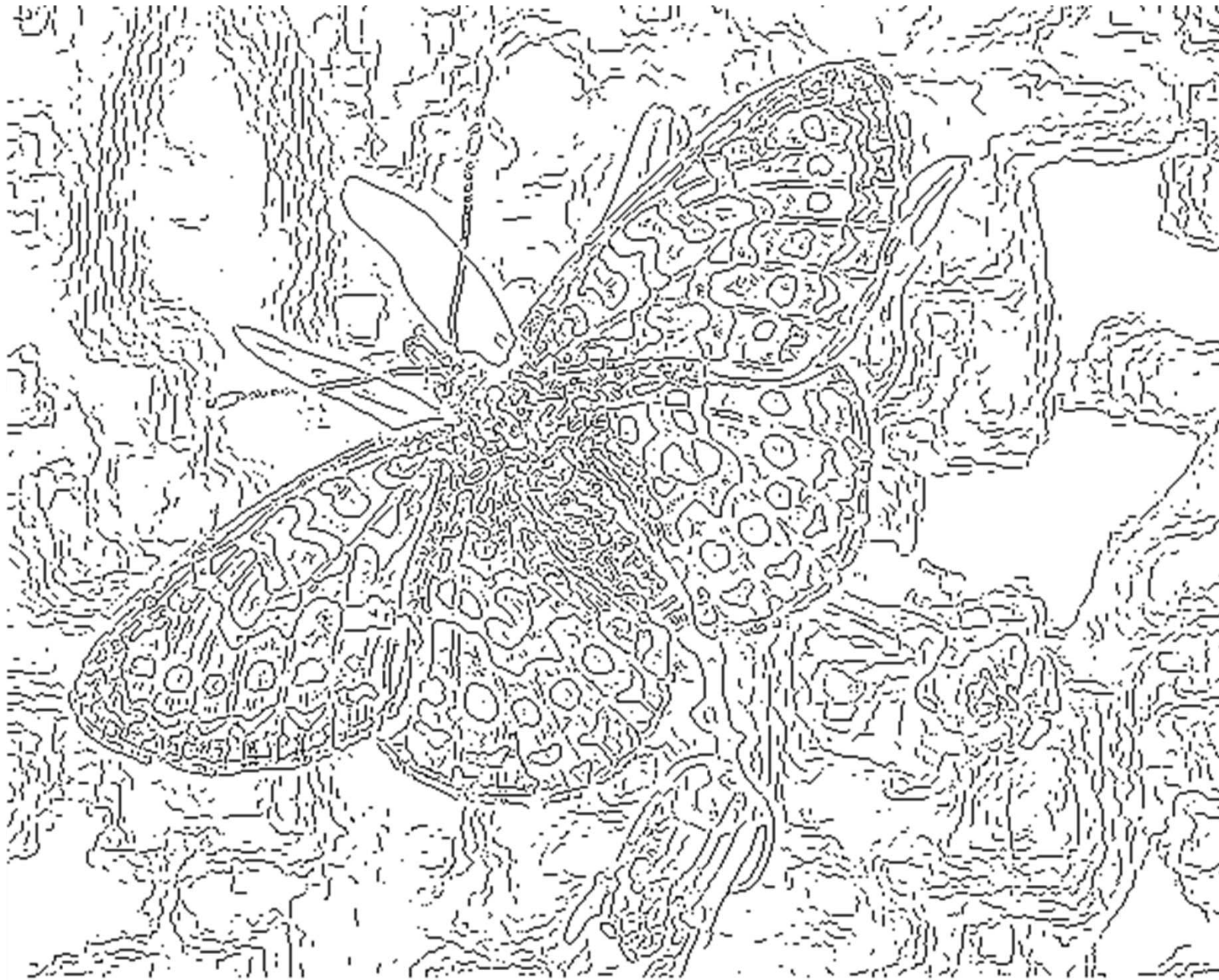
## Multi-resolution Edge Detection

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

(Forsyth & Ponce)







fine scale  
high  
threshold





coarse  
scale,  
high  
threshold





coarse  
scale  
low  
threshold

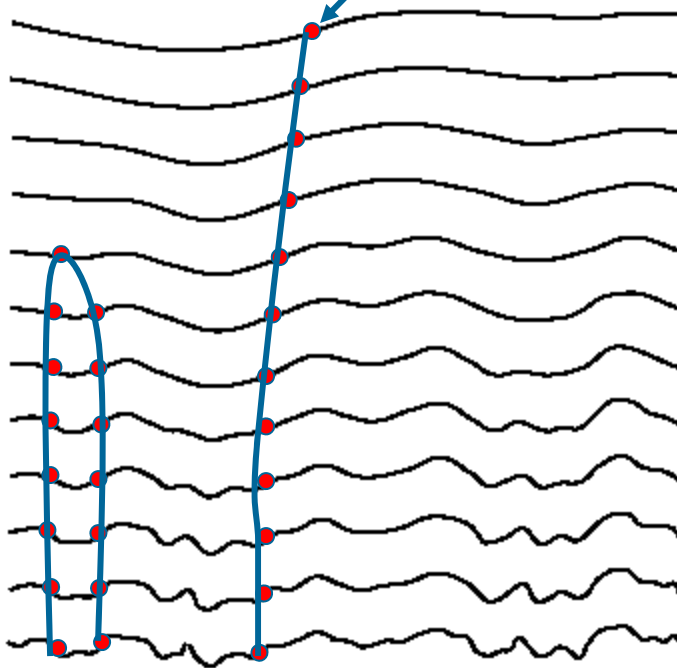




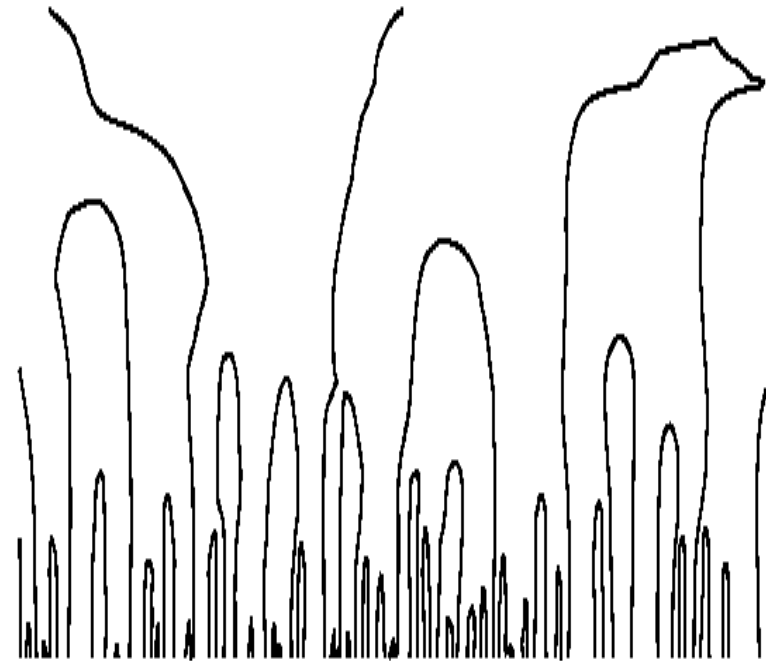
# Scale space (Witkin 83)

first derivative peaks

larger  $\sigma$



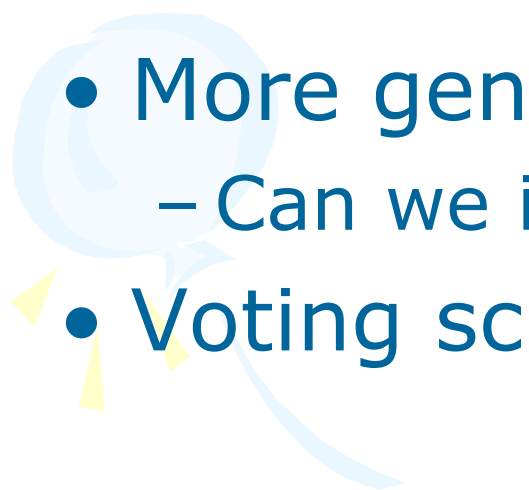

Gaussian filtered signal



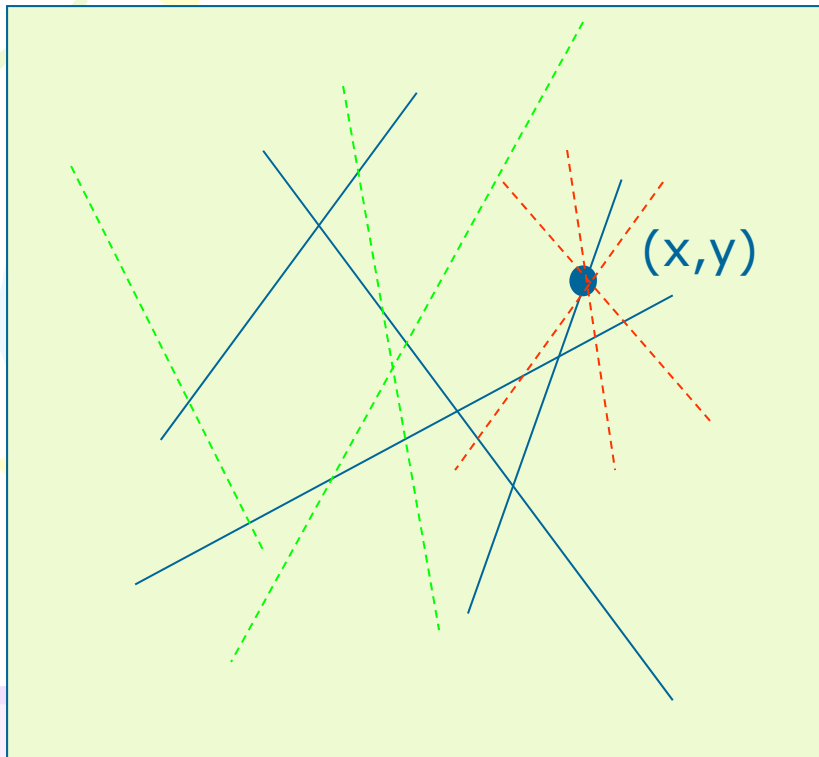
- Properties of scale space (with smoothing)
  - edge position may shift with increasing scale ( $\sigma$ )
  - two edges may merge with increasing scale
  - an edge may **not** split into two with increasing scale



# 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  $y = mx + b$

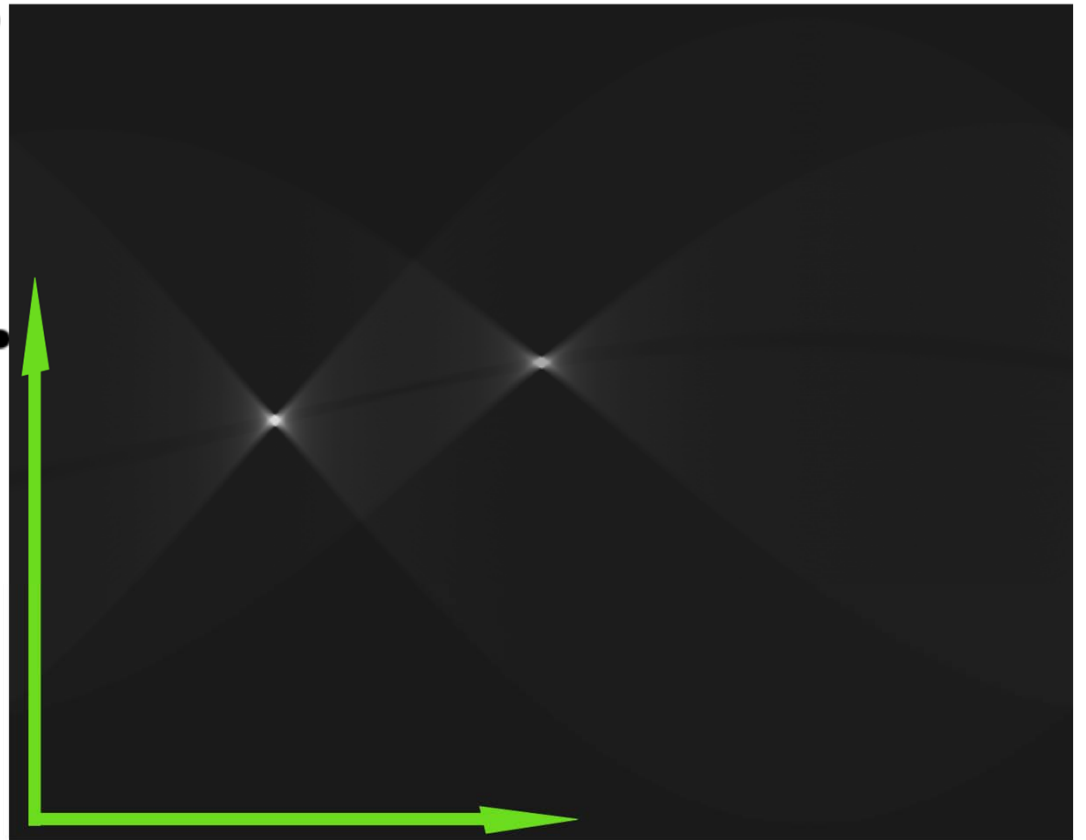
# Example

Input Image

Rendering of Transform Results

Distance from Centre

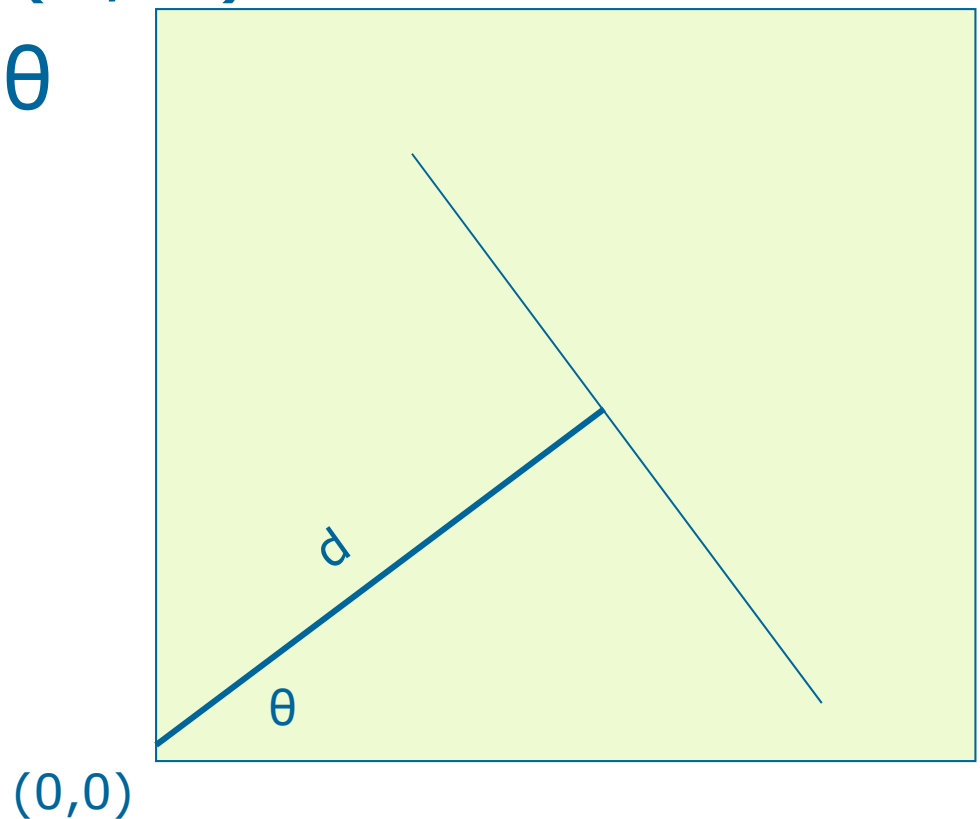
Angle





# Problem with (m,b) space

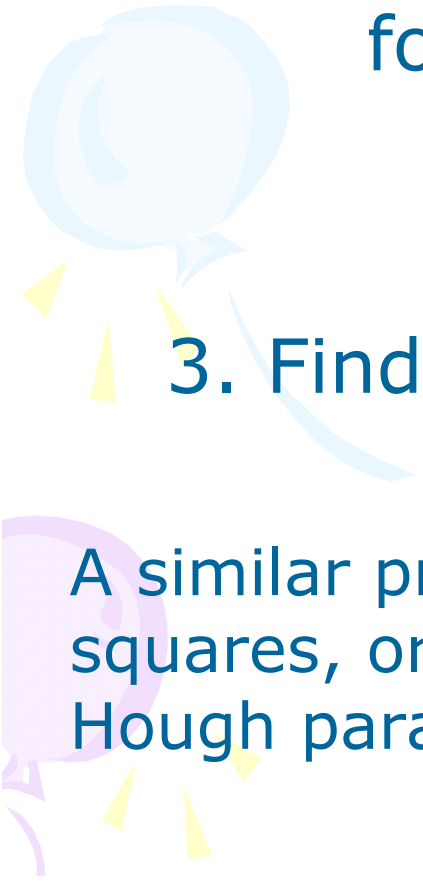
- Vertical lines have infinite m
- Polar notation of  $(d, \theta)$
- $d = x \cos \theta + y \sin \theta$



A green balloon with yellow streamers is positioned to the left of the title.

# Basic Hough Transform


1. Initialize  $H[d, \theta] = 0$
2. for each edge point  $I[x, y]$  in the image  
for  $\theta = 0$  to  $180$   
     $d = x \cos \theta + y \sin \theta$   
     $H[d, \theta] += 1$
3. Find the value(s) of  $(d, \theta)$  for  $\max H[d, \theta]$

A blue balloon with yellow streamers is positioned to the left of the text.

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



# Corner Detections

- Corners have more lines passing through them than pixels on edges
  - Should be easier
  - But edge detectors fail – why?
    - Right at corner, gradient is ill-defined
    - Near corner, gradient has two different values
- 

A decorative graphic on the left side of the slide featuring three balloons: a green one at the top, a blue one in the middle, and a purple one at the bottom. Each balloon has a string and several small yellow triangular flags attached to it.

# Moravec Operator

- Self-similarity
  - How similar are neighboring patches largely overlapping to me?
- Most regions - Very similar
- Edges - Not similar in one direction (perpendicular to edge)
- Corners – not similar in any direction
- Interest point detection – not only corners

# Measuring self-similarity

- SSD = Sum of squared differences
- Corner is local maxima

		B1	B2	B3	
	A1	A2 B4	A3 B5	B6	
	A4	A5 B7	A6 B8	B9	
	A7	A8	A9		

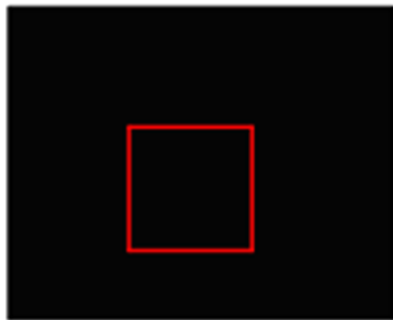
$$V = \sum_{i=1}^9 (A_i - B_i)^2 = 2 * 255^2$$

		B1	B2	B3	
	A1	A2 B4	A3 B5	B6	
	A4	A5 B7	A6 B8	B9	
	A7	A8	A9		

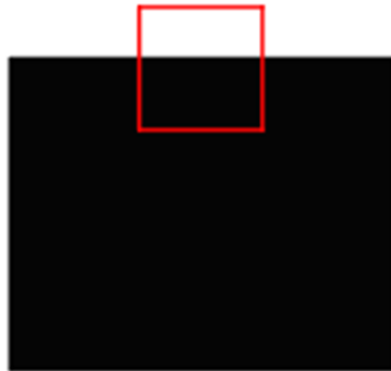
$$V = \sum_{i=1}^9 (A_i - B_i)^2 = 3 * 255^2$$

# Limitations

- Sensitive to noise
  - Responds for isolated pixel
- Larger patches for robustness



A. Interior Region  
Little intensity variation  
in any direction



B. Edge  
Little intensity variation  
along edge, large  
variation perpendicular  
to edge



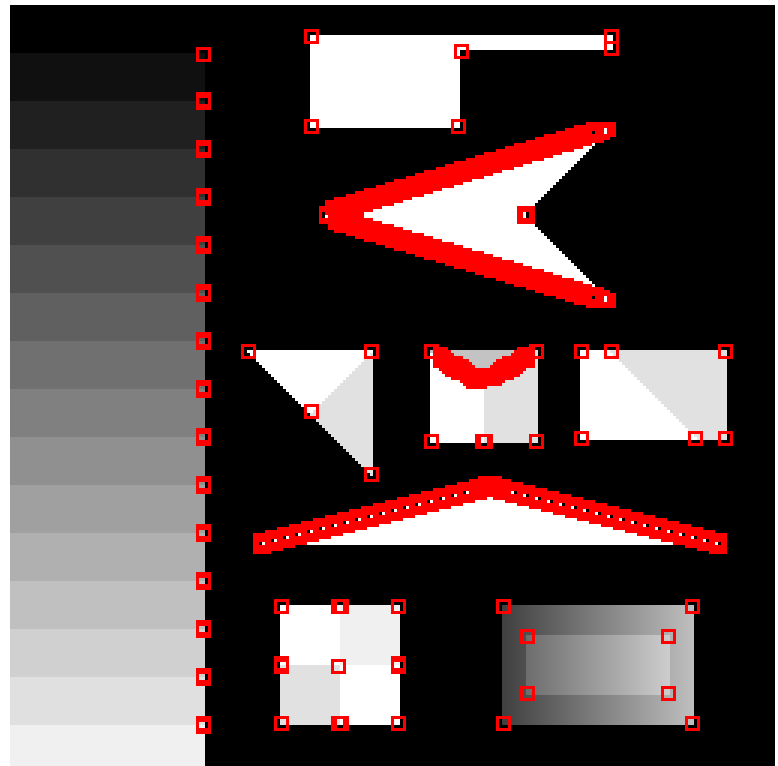
C. Edge  
Large intensity variation  
in all directions



D. Edge  
Large intensity variation  
in all directions

# Limitations

- Responds also to diagonal edges



# Limitations

- Anisotropic (Not rotationally invariant)



Original Image



Image Rotated 30°



A decorative graphic on the left side of the slide featuring three balloons: a light green one at the top, a light blue one in the middle, and a light purple one at the bottom. Each balloon has a string and small yellow triangular flags attached to it.

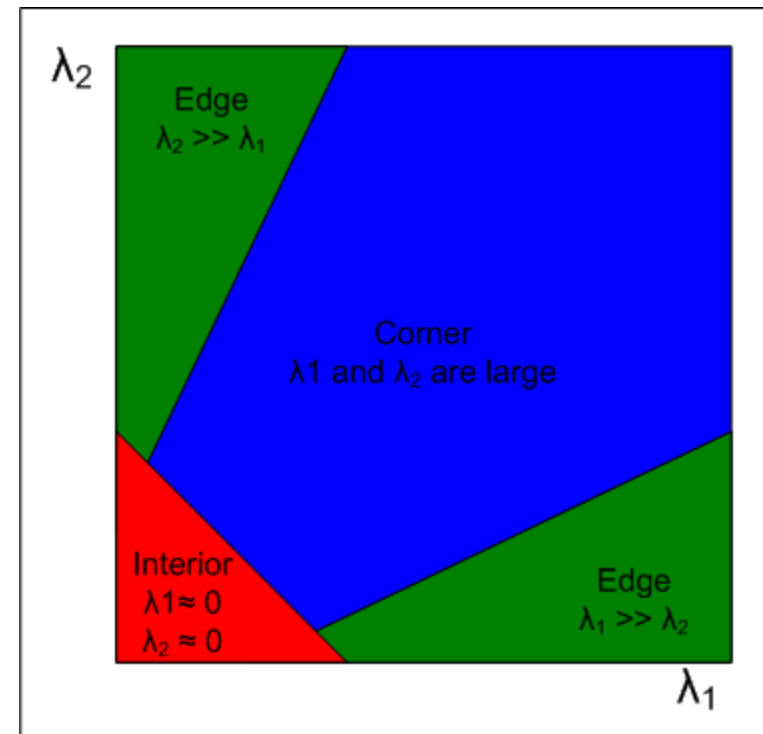
# Harris & Stephens/Plessey Corner Detector

- Consider the differential of the corner score with respect to direction
- Describes the geometry of the image surface near the point (u,v)

$$A = \sum_u \sum_v w(u, v) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} = \begin{bmatrix} \langle I_x^2 \rangle & \langle I_x I_y \rangle \\ \langle I_x I_y \rangle & \langle I_y^2 \rangle \end{bmatrix},$$

# How to find the corner?

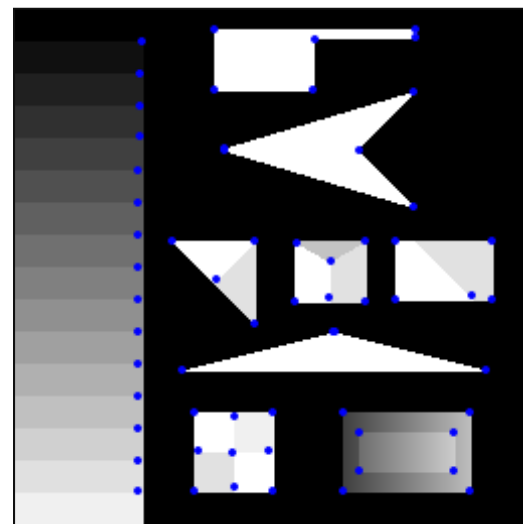
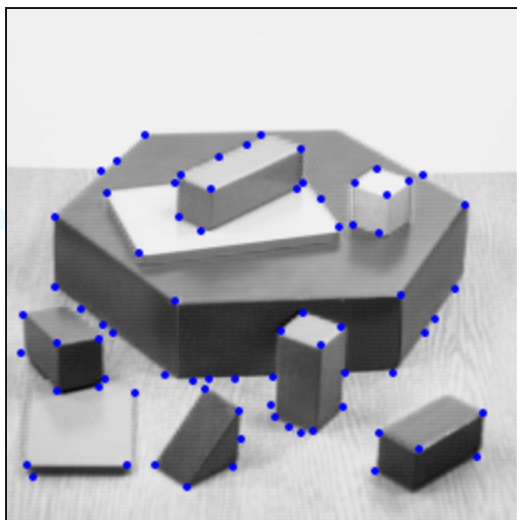
- The eigenvalues are proportional to the principal curvatures
- If both small, no edge/corner
- If one big and one small, edge
- If both big, then corner



# Rotationally Invariant

- If  $w$  is Gaussian, then this is isotropic

$$A = \sum_u \sum_v w(u, v) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} = \begin{bmatrix} \langle I_x^2 \rangle & \langle I_x I_y \rangle \\ \langle I_x I_y \rangle & \langle I_y^2 \rangle \end{bmatrix},$$

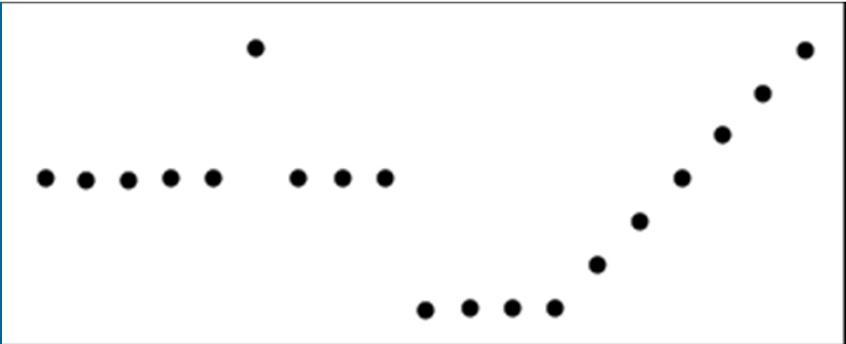
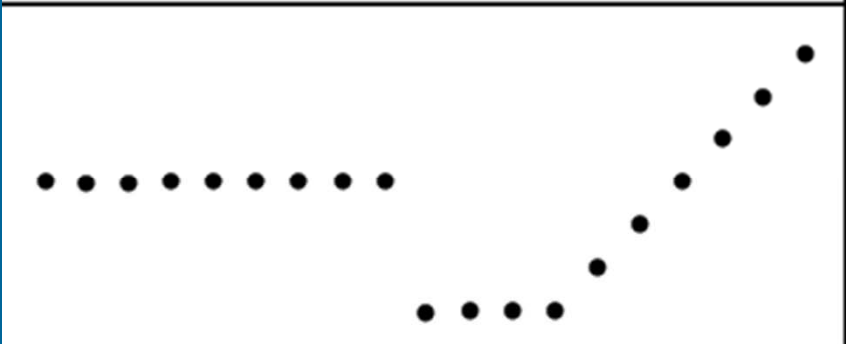





# Non-linear filters: Median filter

- Replace by median of the neighborhood
- No new gray levels
- Removes the odd man out
  - Good for outlier removal
- Retains edges

# Median filter

	INPUT
	MEDIAN
	MEAN