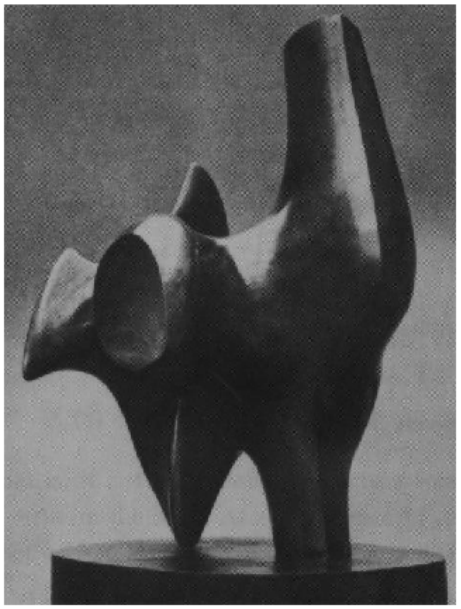
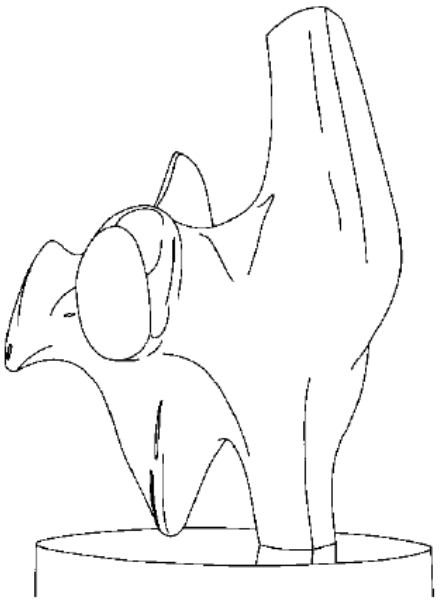
Edge Detectors

Edge Detection

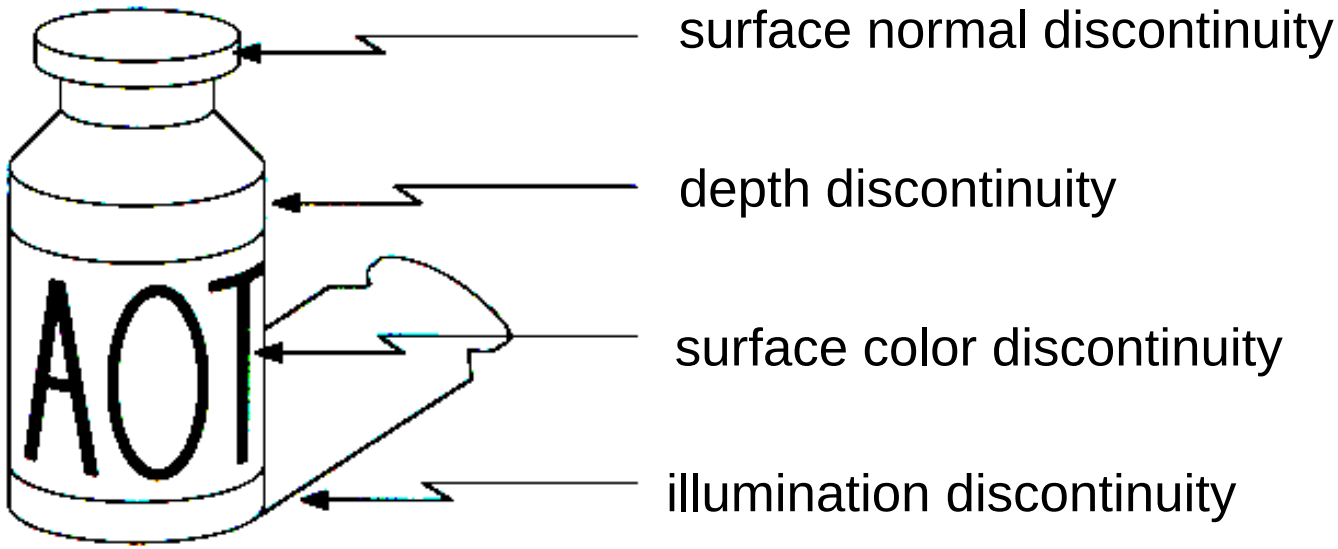
- Convert a 2D image into a set of curves
  - Extracts salient features of the scene
  - More compact than pixels





Edges

- Edges are caused by a variety of factors



Edges

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

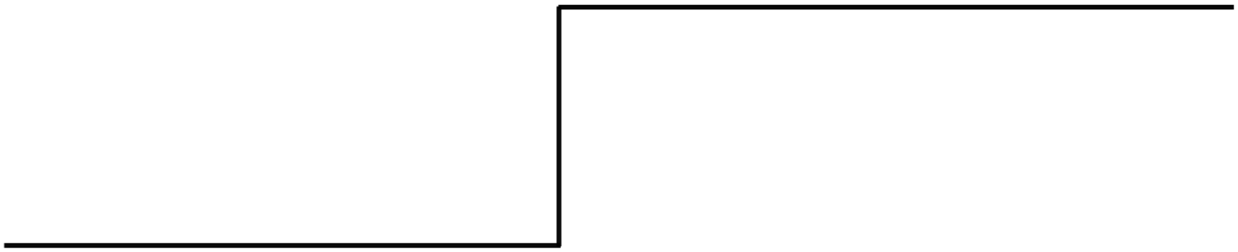
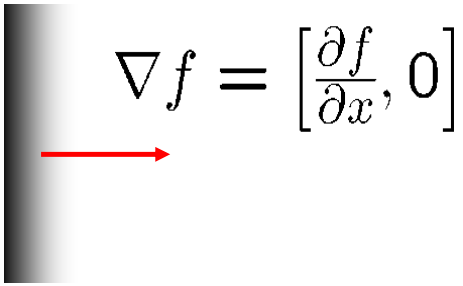
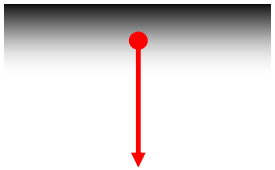


Image Gradient

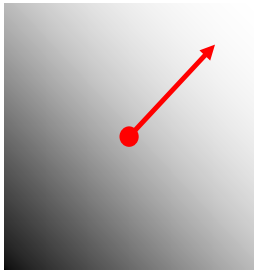
- The gradient of an image: $\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$
- The gradient points in the direction of most rapid change in intensity



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



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



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

Image Gradient

- gradient direction

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

- The edge strength

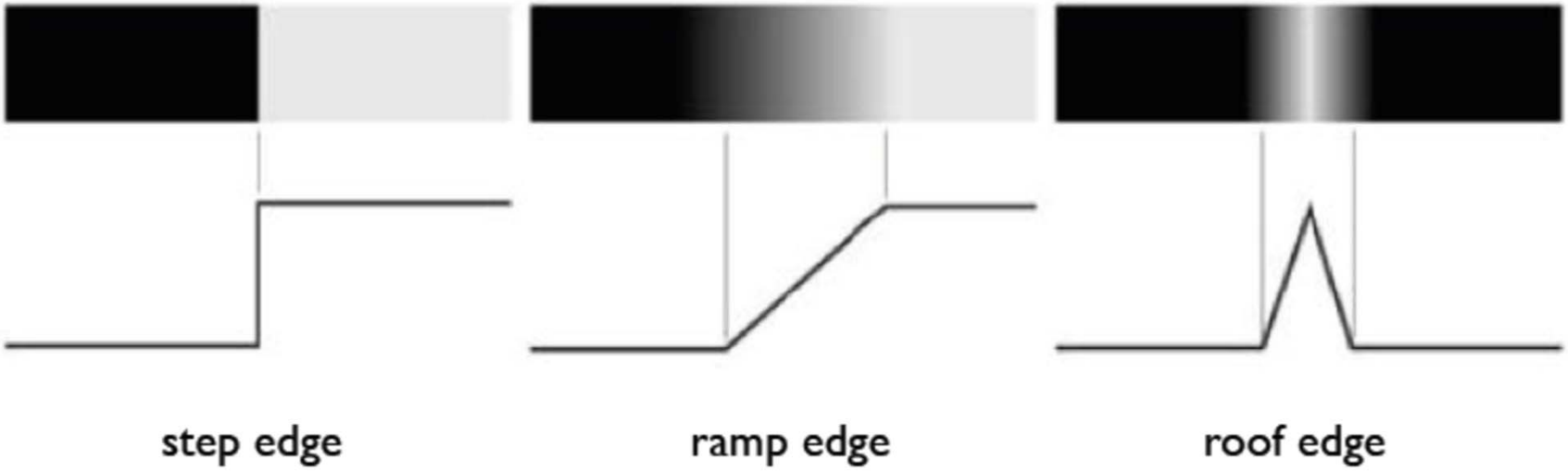
$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

- **Discrete Gradient**: by finite differences

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

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

Types of Edges



Sobel Operator

- Better approximations of the derivatives
  - *Sobel* operators

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

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

- The standard def. 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

Sobel Operator - Result

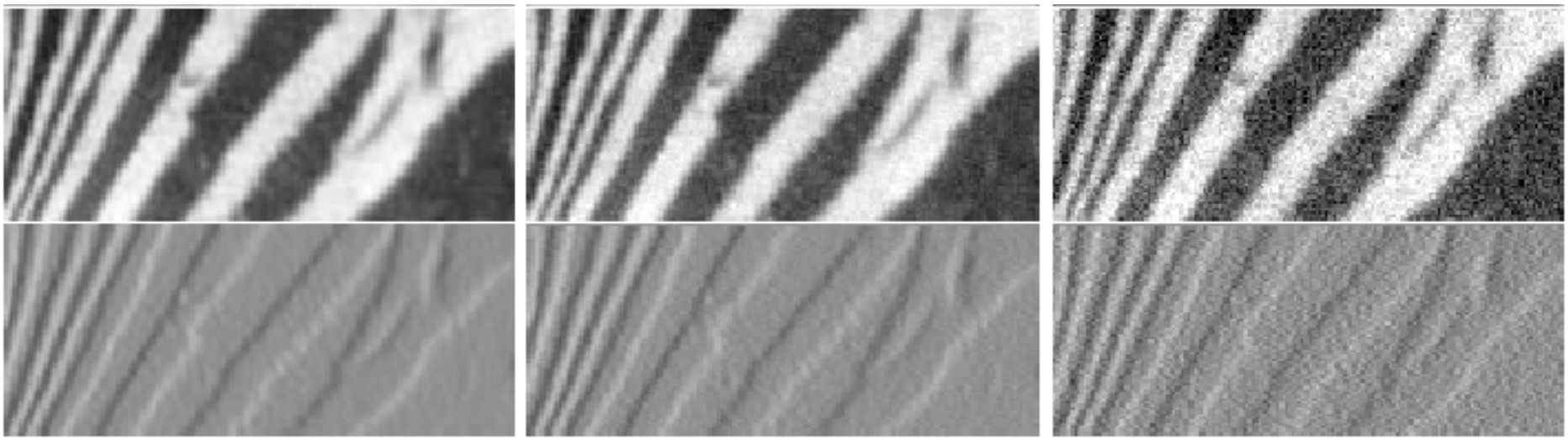
Original

Convolution with Sobel

Thresholding (Value = 64)

Thresholding (Value = 96)

Noise - Effect



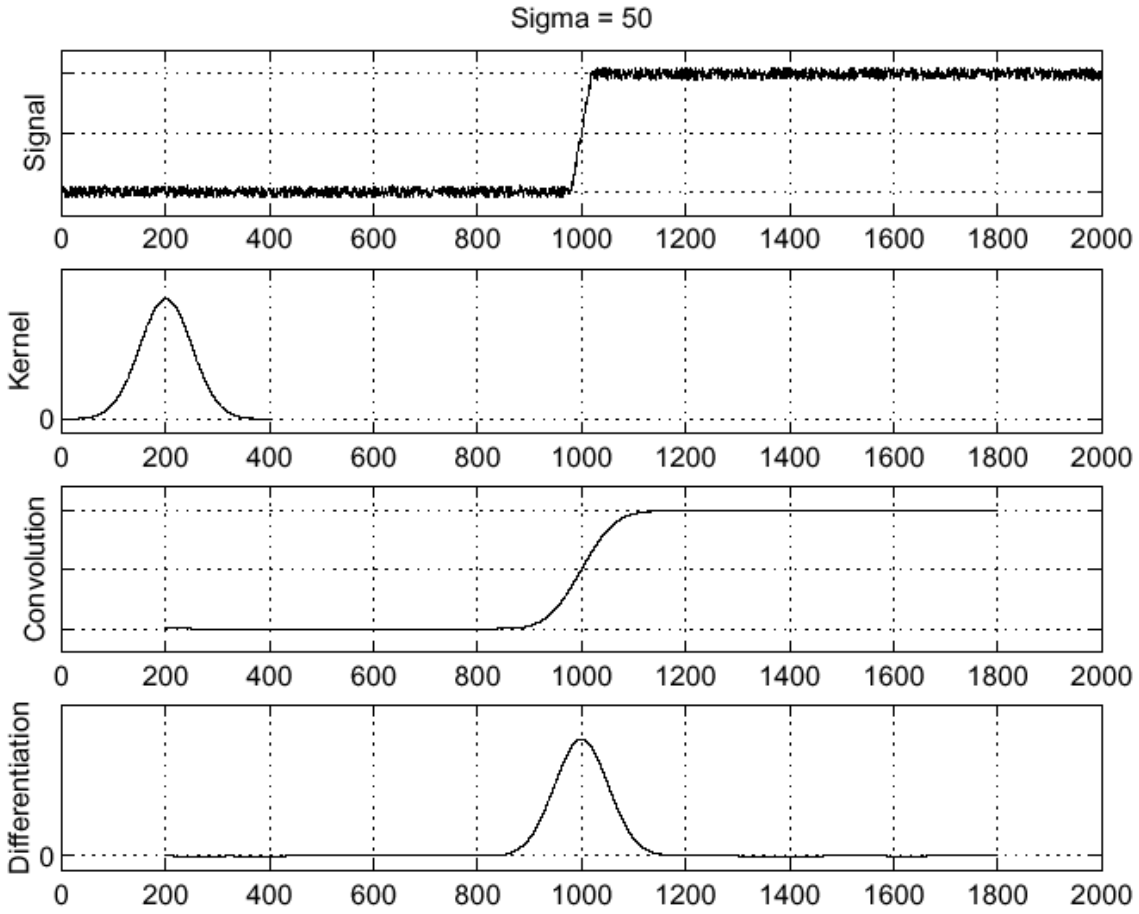
Solution: Smooth First

f

h

$h * f$

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



Derivative Theorem

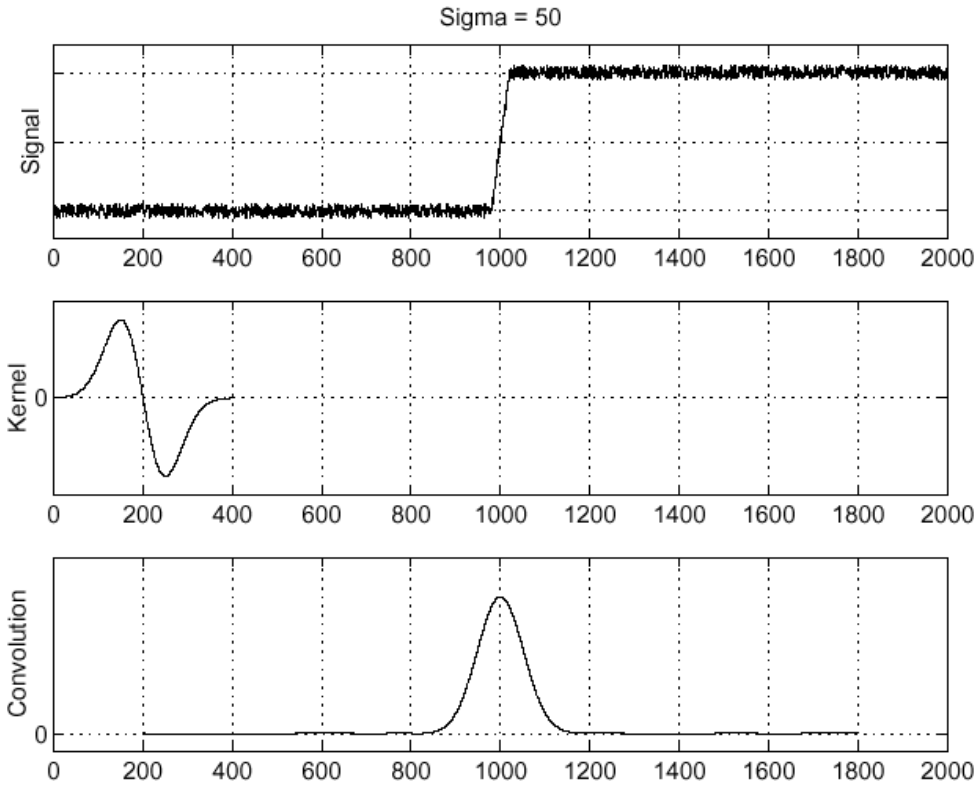
$$\frac{\partial}{\partial x}(h \star f) = (\frac{\partial}{\partial x}h) \star f$$

- This saves us one operation:

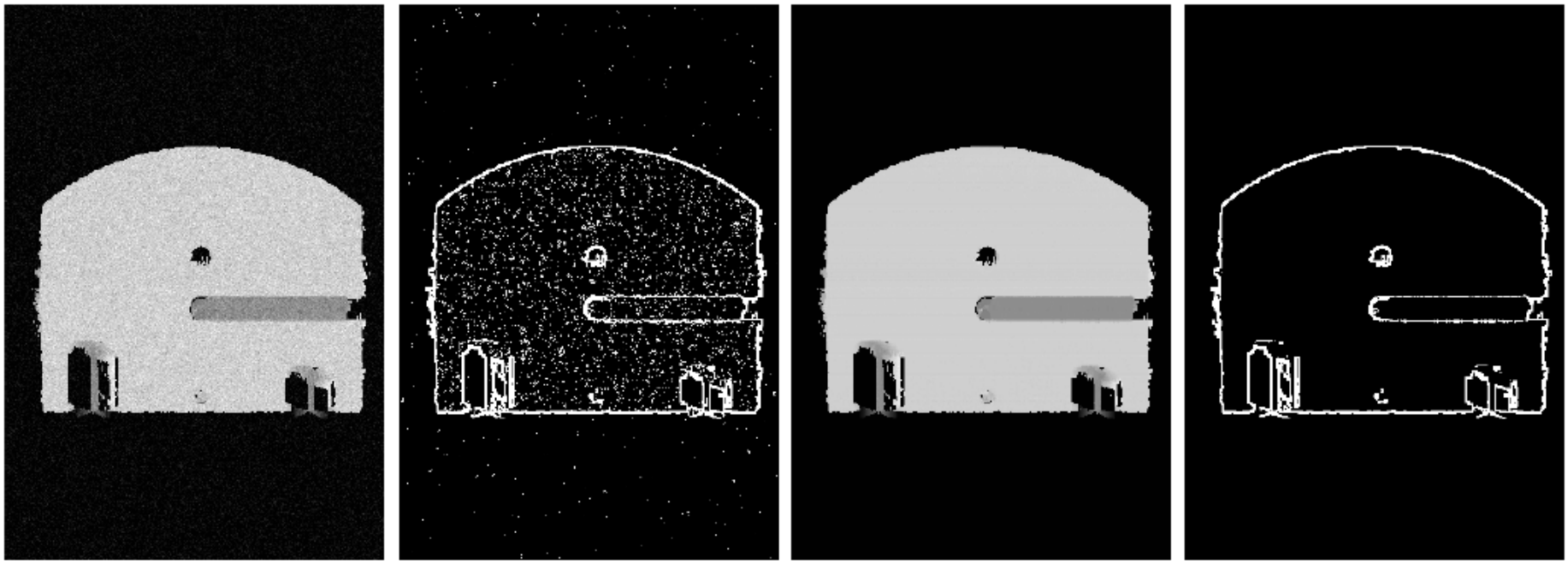
f

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

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



Result



Without Gaussian

With Gaussian

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

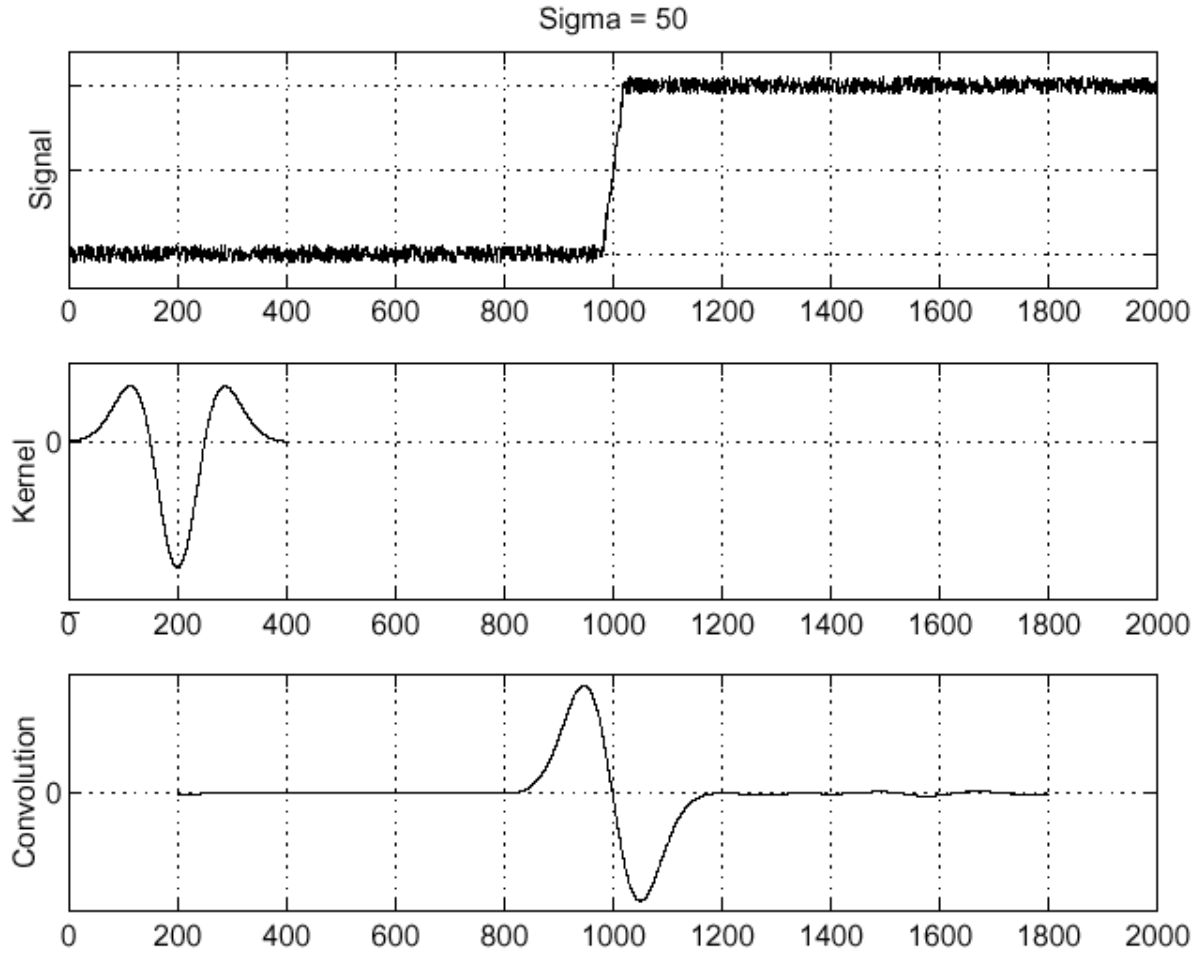
−1	−1	−1
−1	8	−1
−1	−1	−1

Laplacian of Gaussian

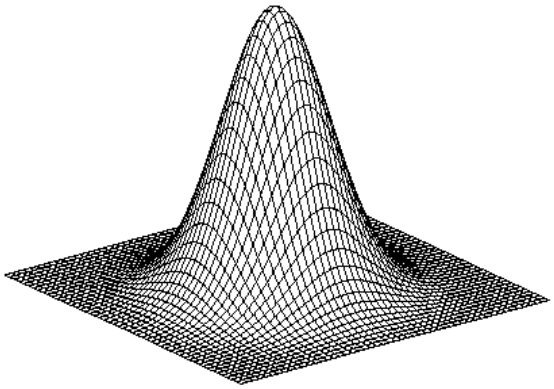
f

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

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

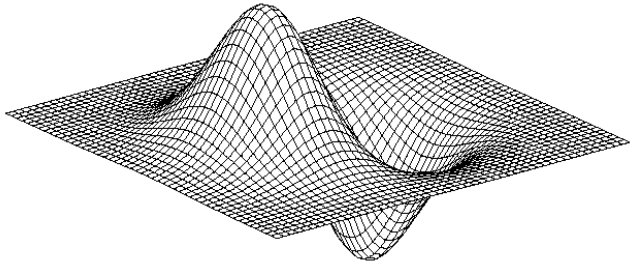


2D edge detection filters



Gaussian

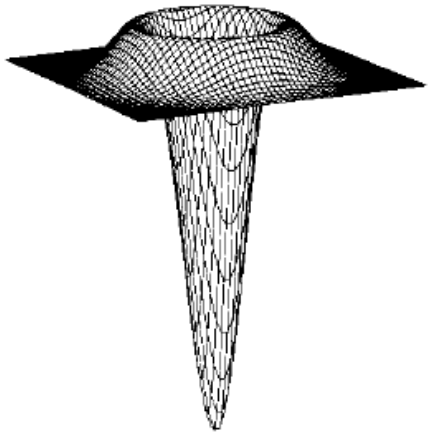
$$h_\sigma(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$



derivative of Gaussian

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

Laplacian of Gaussian



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

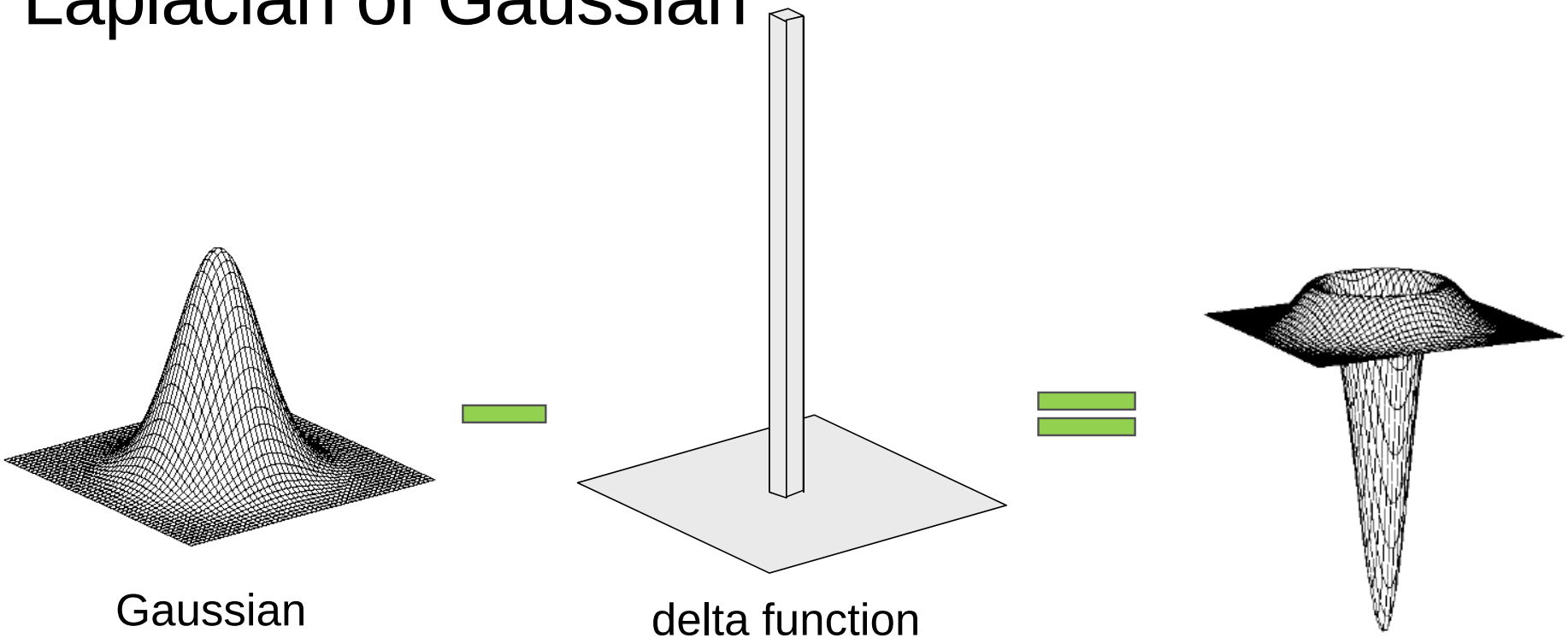
Edge detection by subtraction

original

smoothed

smoothed – original

Laplacian of Gaussian



Gaussian

delta function

Laplacian of Gaussian

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 Suppression

- 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

Steps of Canny Edge Detector

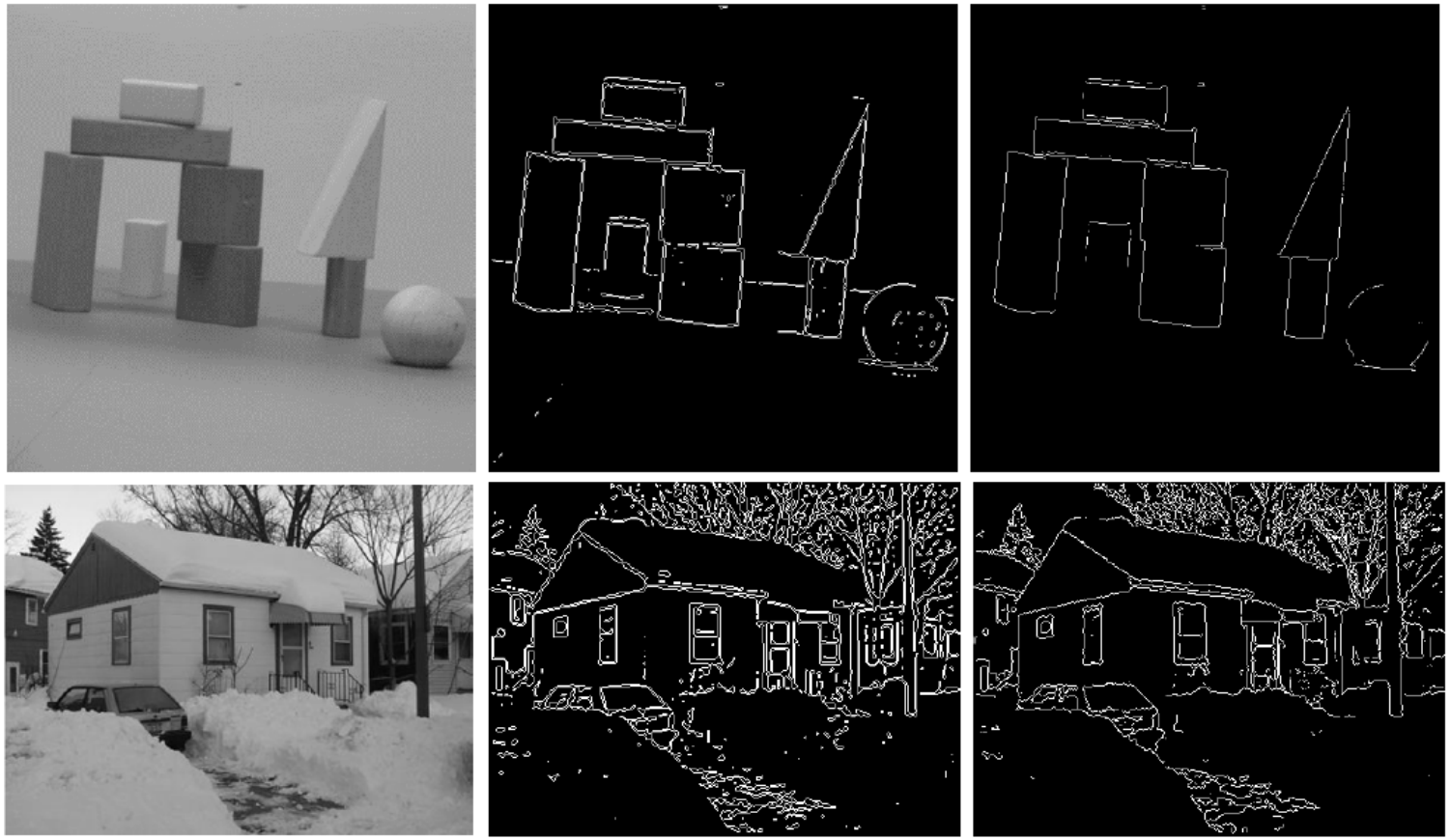
Original

Gradient Magnitude

Non-Maximum Suppression

After Hysteresis

Comparison with Laplacian Based



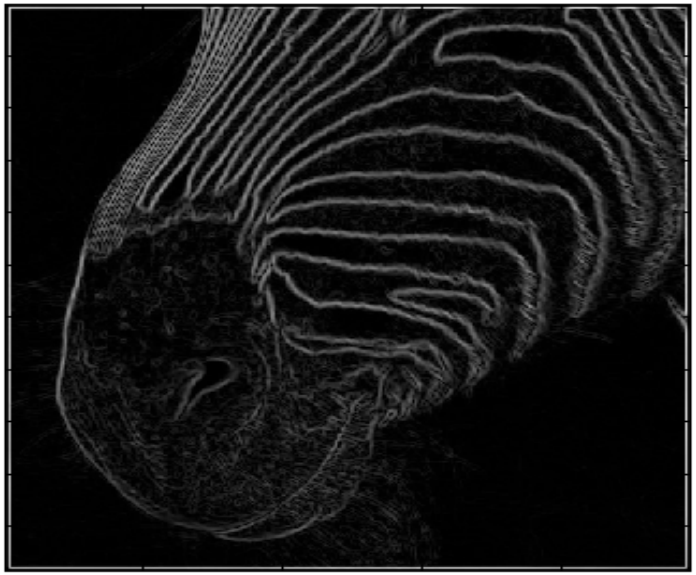
Original | **Curvature Based** | **Canny**

Effect of Smoothing (kernel size)

original

Canny with $\sigma = 1$

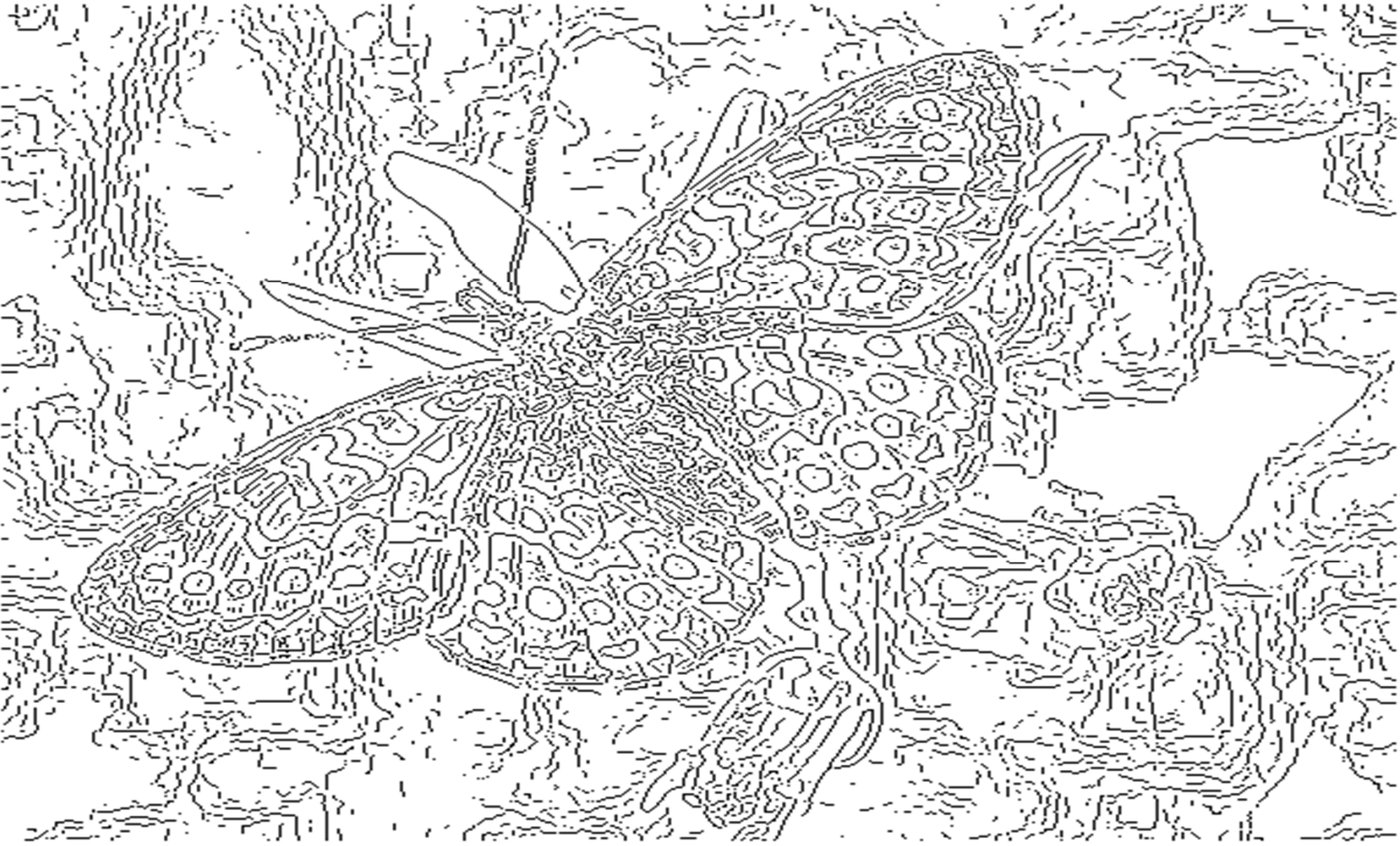
Canny with


Multi-resolution Edge Detection

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

(Forsyth & Ponce)



fine scale, high threshold

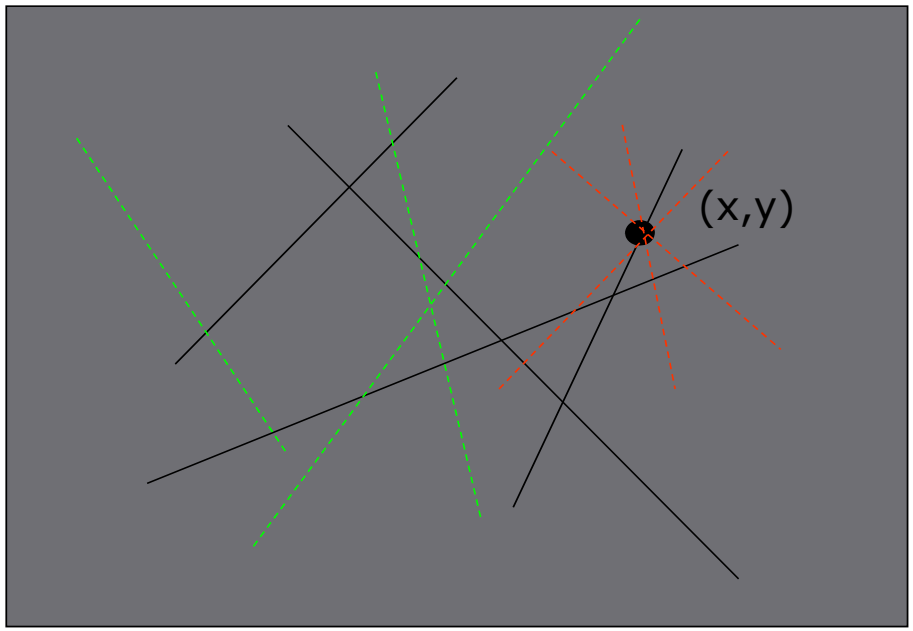
coarse scale, high threshold

Coarse scale, low threshold

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

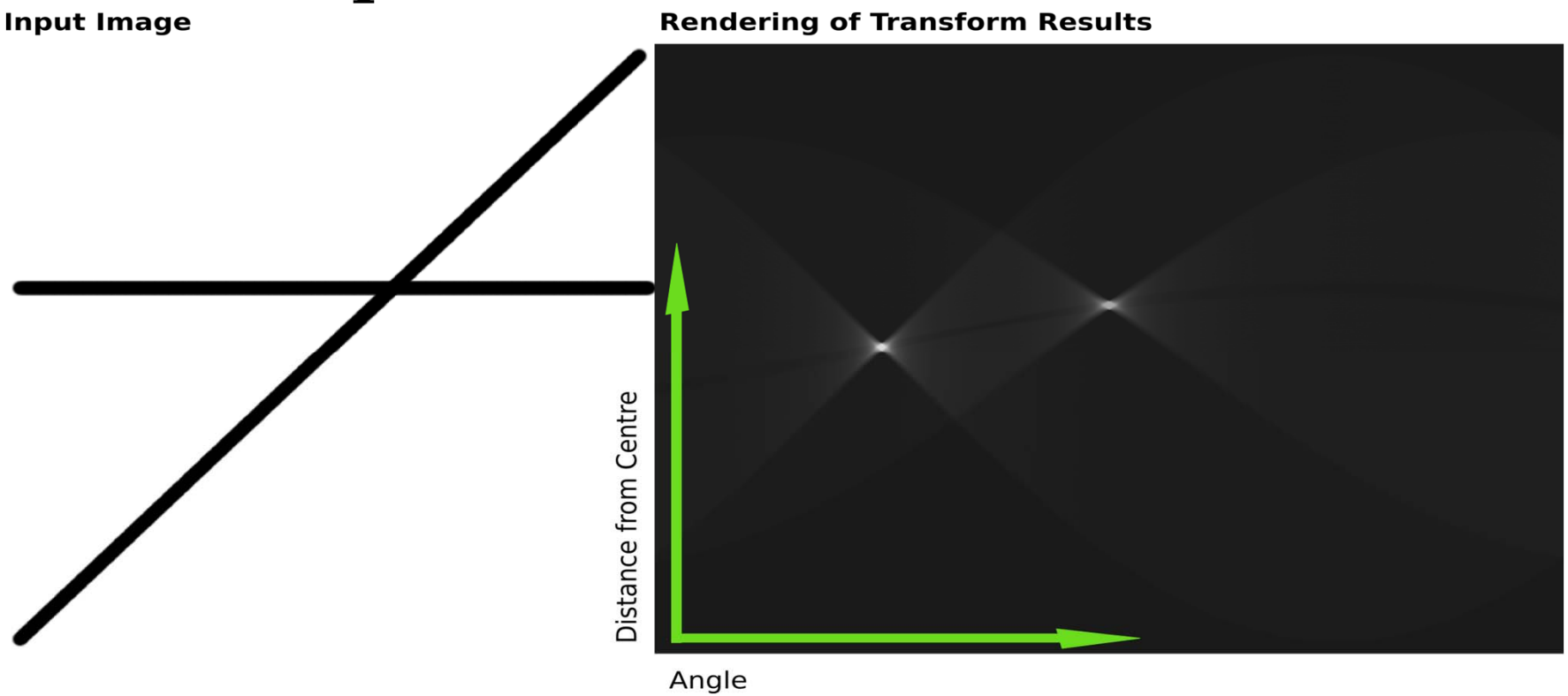


- Only a few lines can pass through (x,y)
  - $mx+b$
- Consider (m,b) space
- Red lines are given by a line in that space
  - $b = y - mx$
- Each point defines a line in the Hough space
- Each line defines a point (since same m,b)

How to identify lines?

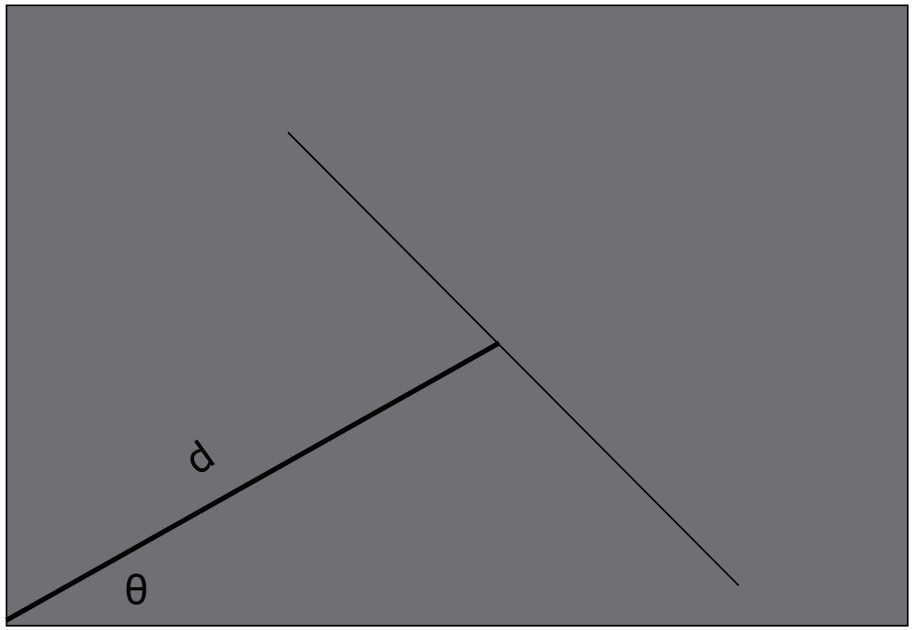
- For each edge point
  - Add intensity to the corresponding line in Hough space
- Each edge point votes on the possible lines through them
- If a line exists in the image space, that point in Hough space will get many votes and hence high intensity
- Find maxima in Hough space
- Find lines by equations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

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

   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.

Non-Linear Filters

Corner Detections

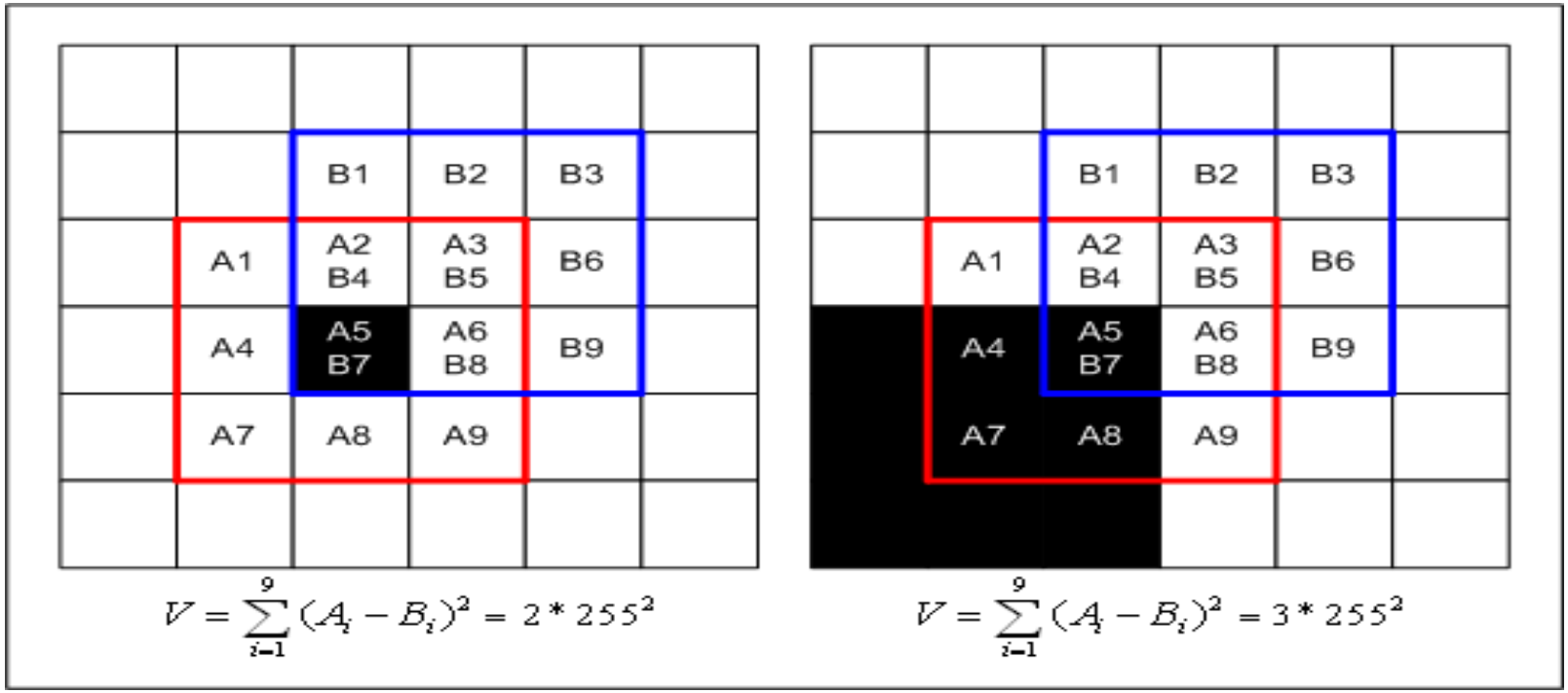
- Corners have more lines passing through them than pixels on edges
- Should be easier
- But edge detectors fail – why?
  - Right at corner, gradient is ill-defined
  - Near corner, gradient has two different values

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

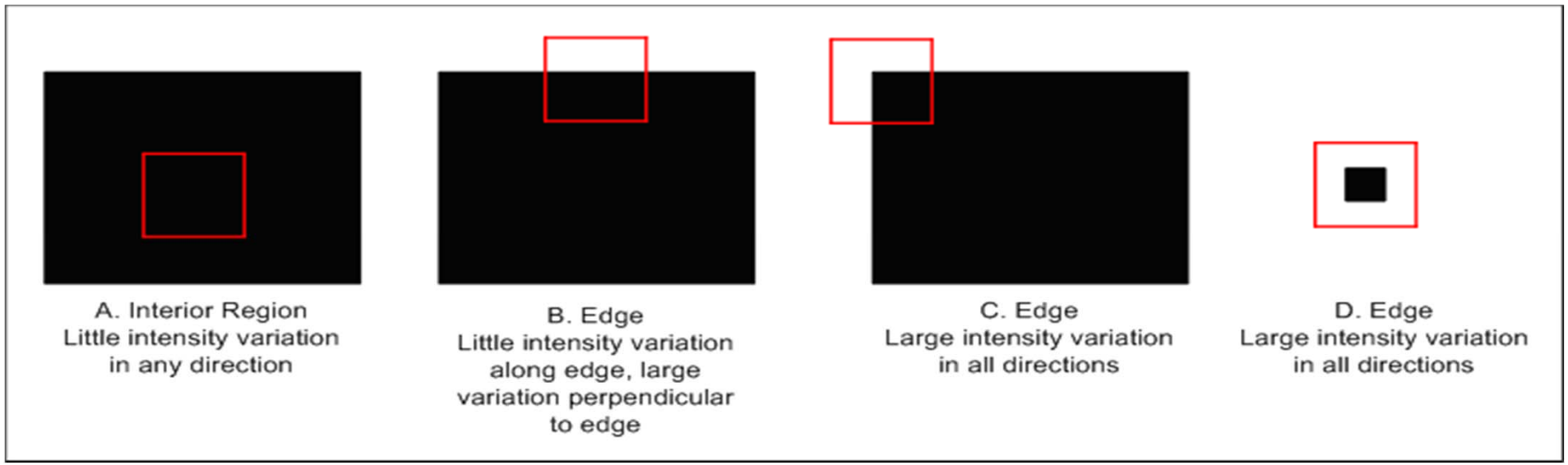
Measuring self-similarity

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



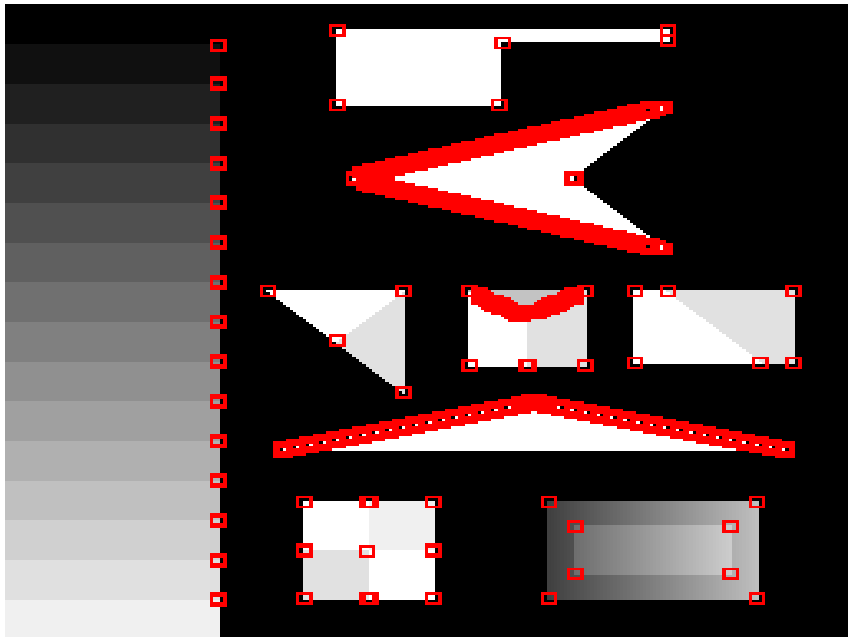
Limitations

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



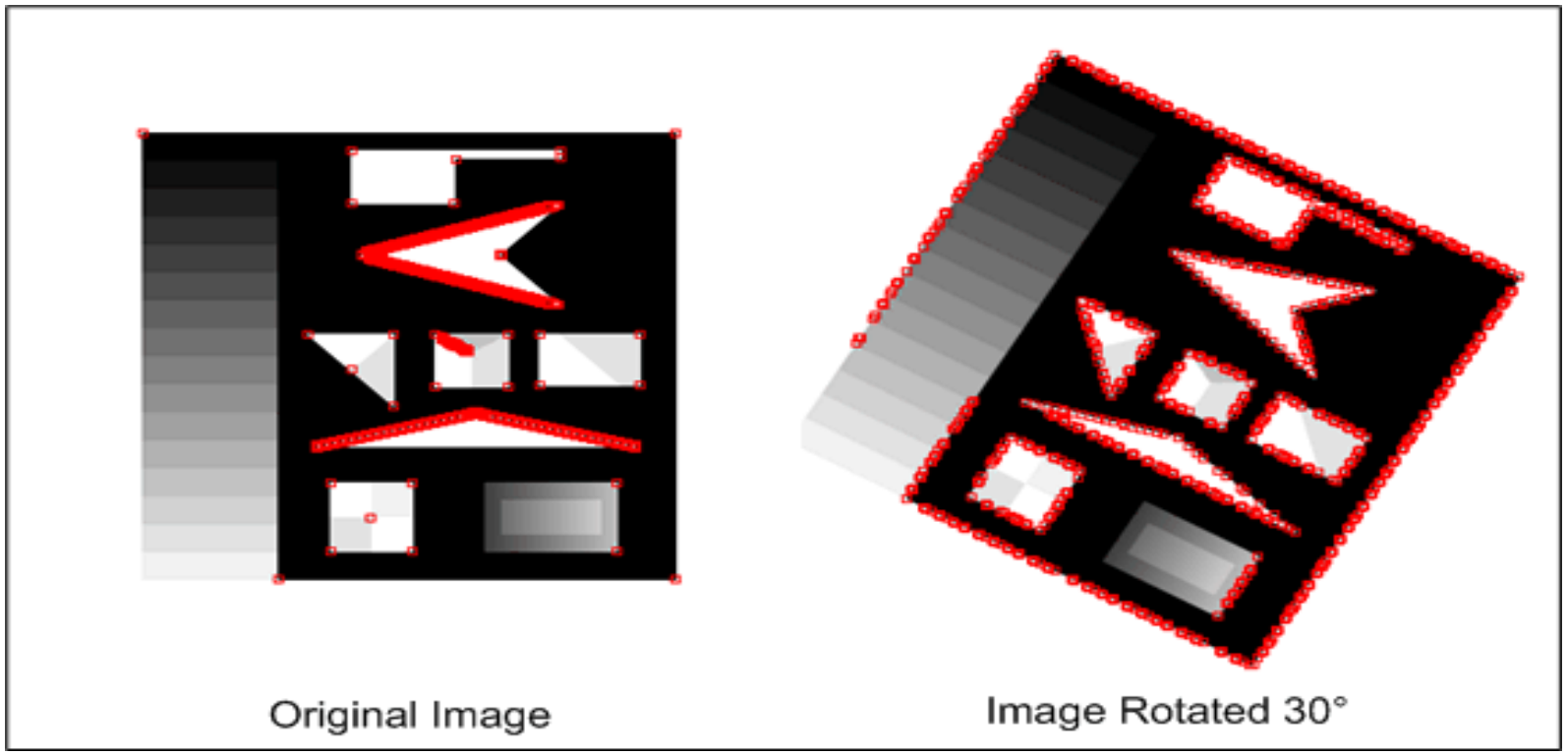
Limitations

- Responds also to diagonal edges



Limitations

- Anisotropic (Not rotationally invariant)



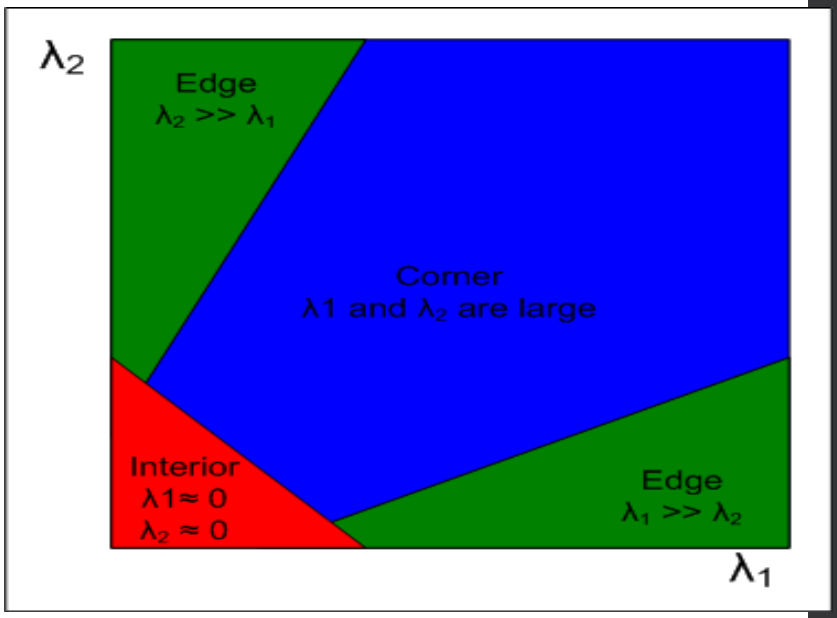
Harris & Stephens/Plessey Corner Detector

- 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?

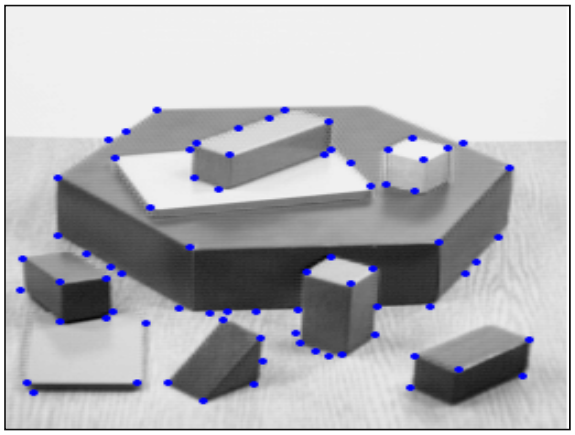
- The eigenvalues are proportional to the principal curvatures
- If both small, no edge/corner
- If one big and one small, edge
- If both big, then corner

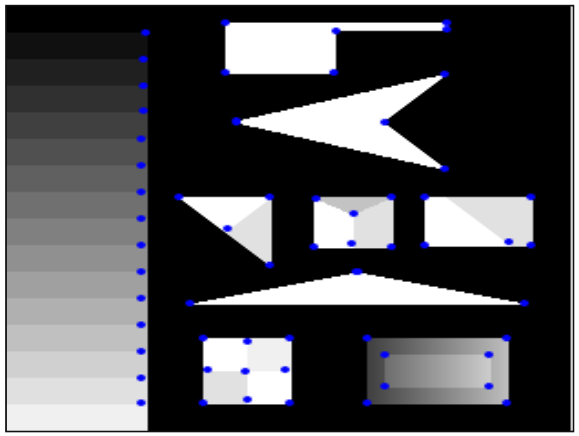


Rotationally Invariant

- 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