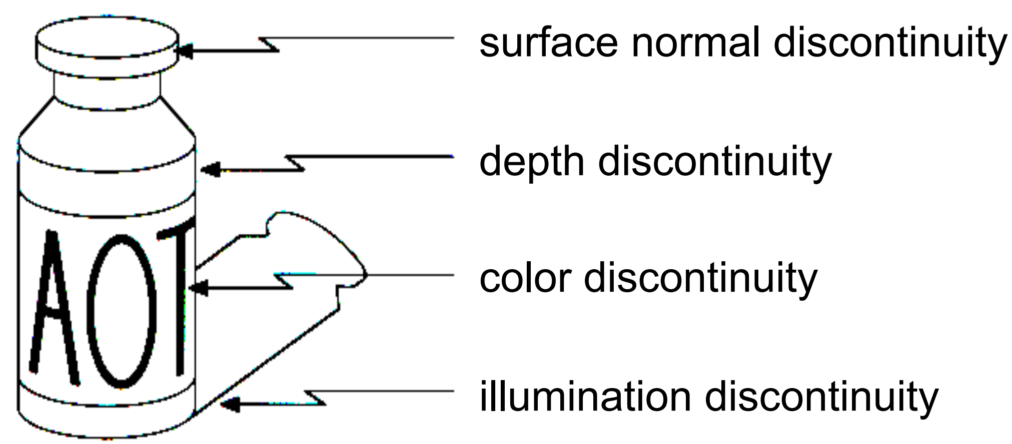
Computational Vision

Lecture 5.1: Advanced Edge Detection

Hamid Dehghani
Office: UG38

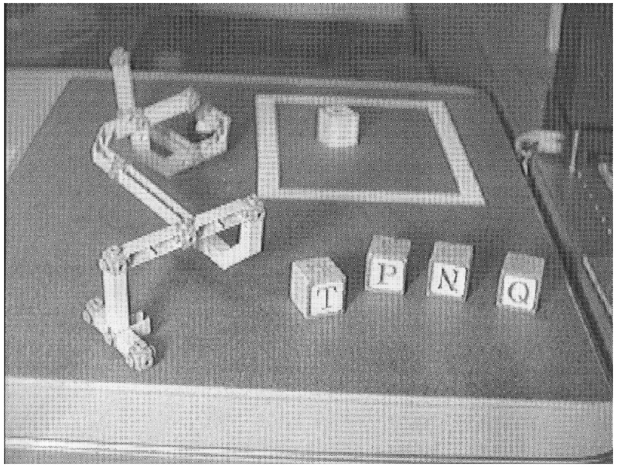
What Causes Intensity Changes?

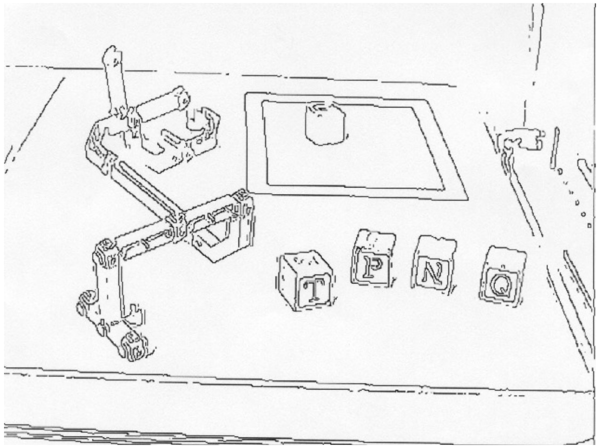
- Geometric events
  - surface orientation (boundary) discontinuities
  - depth discontinuities
  - color and texture discontinuities
- Non-geometric events
  - illumination changes
  - specularities
  - shadows
  - inter-reflections



Goal of Edge Detection

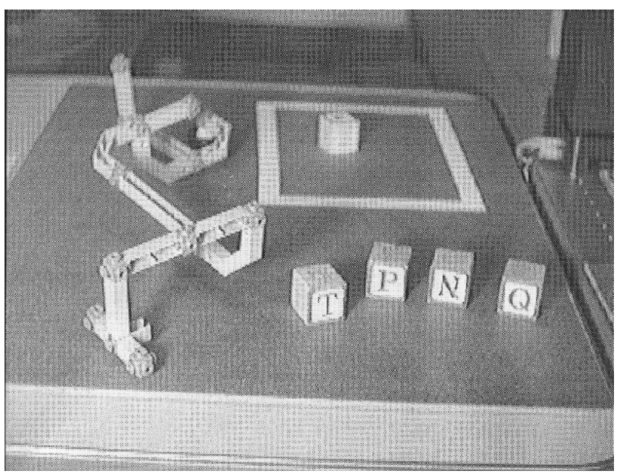
- Produce a line “drawing” of a scene from an image of that scene.

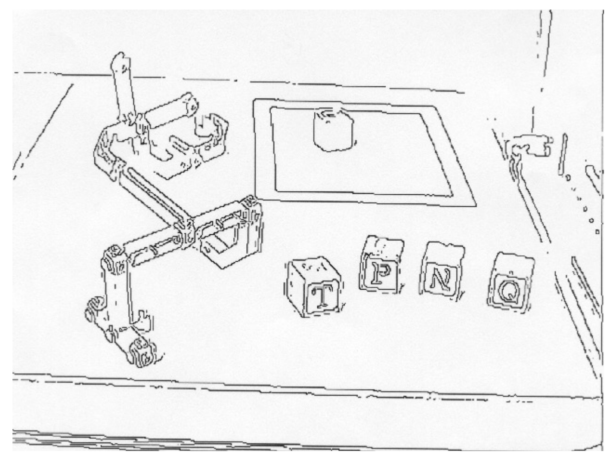




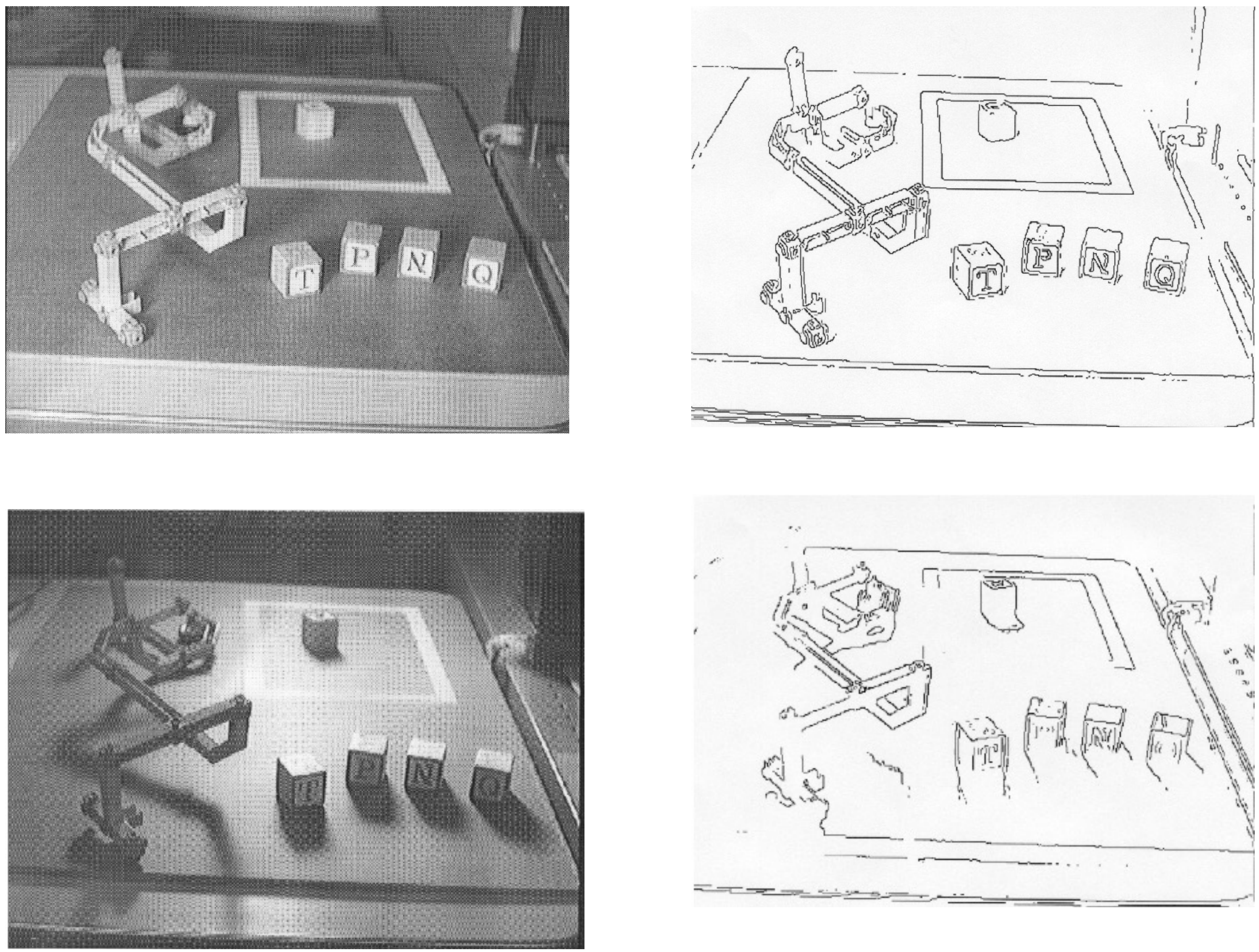
Why is Edge Detection Useful?

- Important features can be extracted from the edges of an image (e.g., corners, lines, curves).
- These features are used by higher-level computer vision algorithms (e.g., recognition).



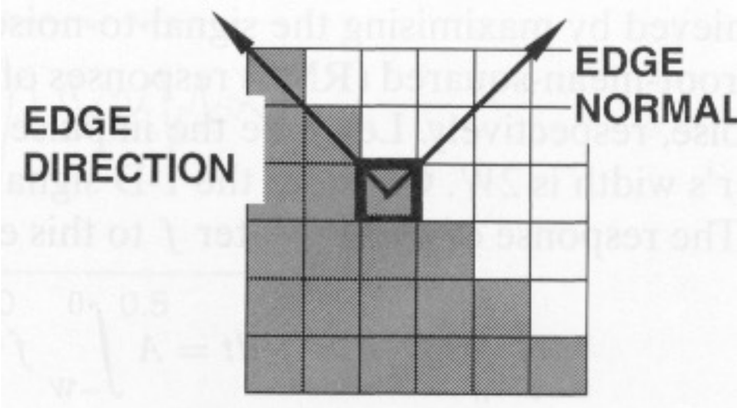


Effect of Illumination



Edge Descriptors

- **Edge direction:** perpendicular to the direction of maximum intensity change (i.e., edge normal)
- **Edge strength:** related to the local image contrast along the normal.
- **Edge position:** the image position at which the edge is located.



Main Steps in Edge Detection

(1) Smoothing: suppress as much noise as possible, without destroying true edges.

(2) Enhancement: apply differentiation to enhance the quality of edges (i.e., sharpening).

Main Steps in Edge Detection (cont'd)

(3) Thresholding: determine which edge pixels should be discarded as noise and which should be retained (i.e., threshold edge magnitude).

(4) Localization: determine the exact edge location.

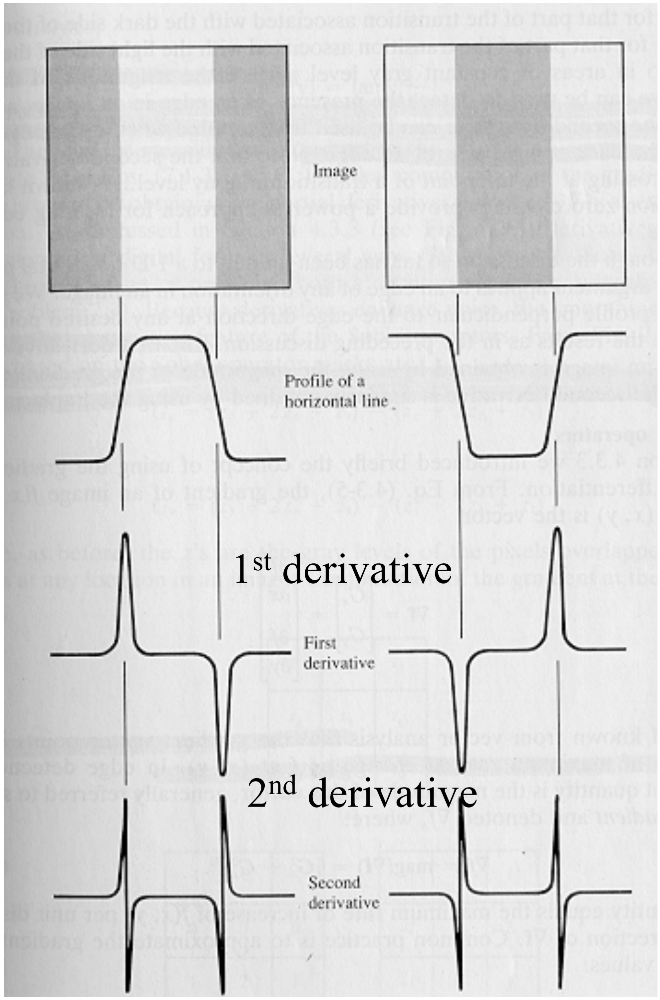
sub-pixel resolution might be required for some applications to estimate the location of an edge to better than the spacing between pixels.

Edge Detection Using Derivatives

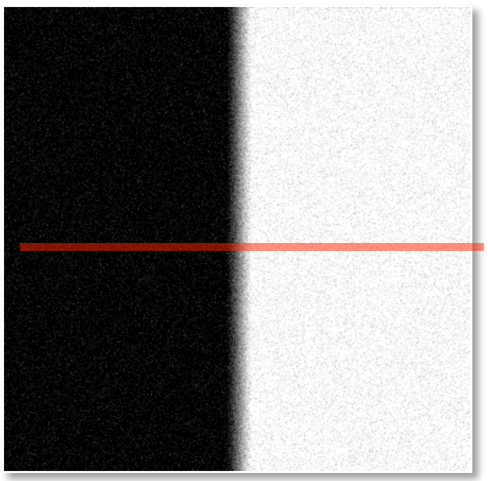
- Often, points that lie on an edge are detected by:

  (1) Detecting the local **maxima** or **minima** of the first derivative.

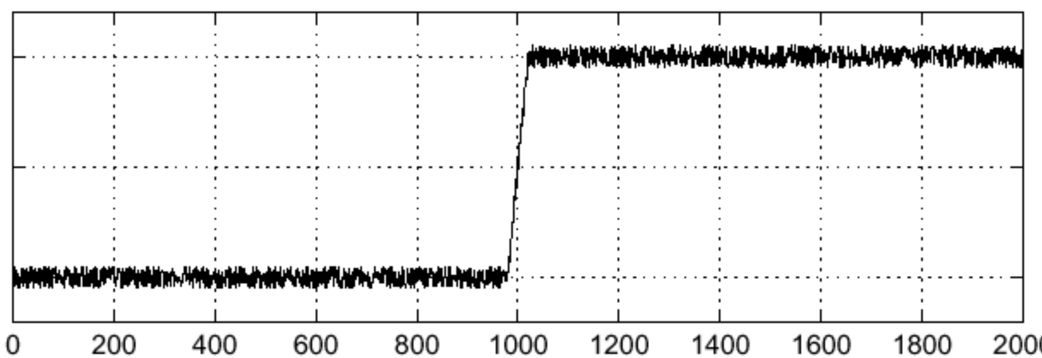
  (2) Detecting the **zero-crossings** of the second derivative.



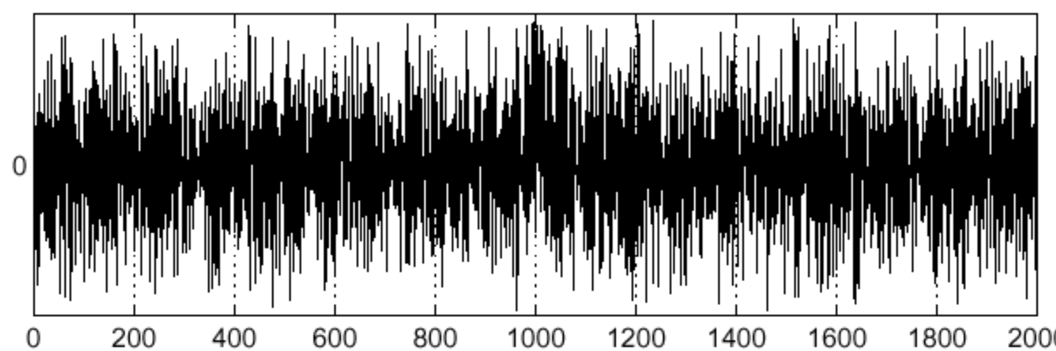
Effect of Smoothing on Derivates



$$f(x)$$

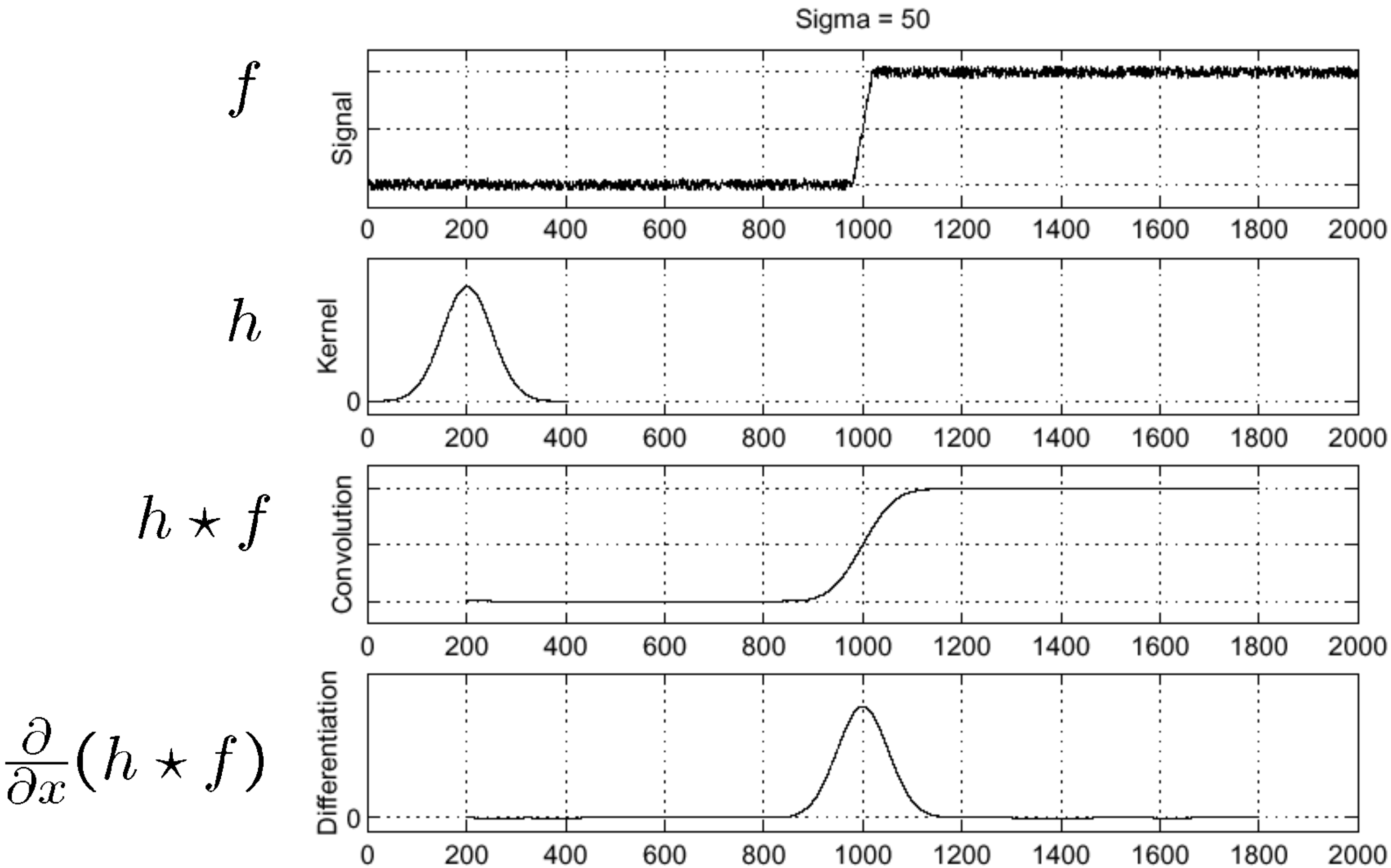


$$\frac{d}{dx}f(x)$$



Where is the edge??

Effect of Smoothing on Derivatives (cont' d)



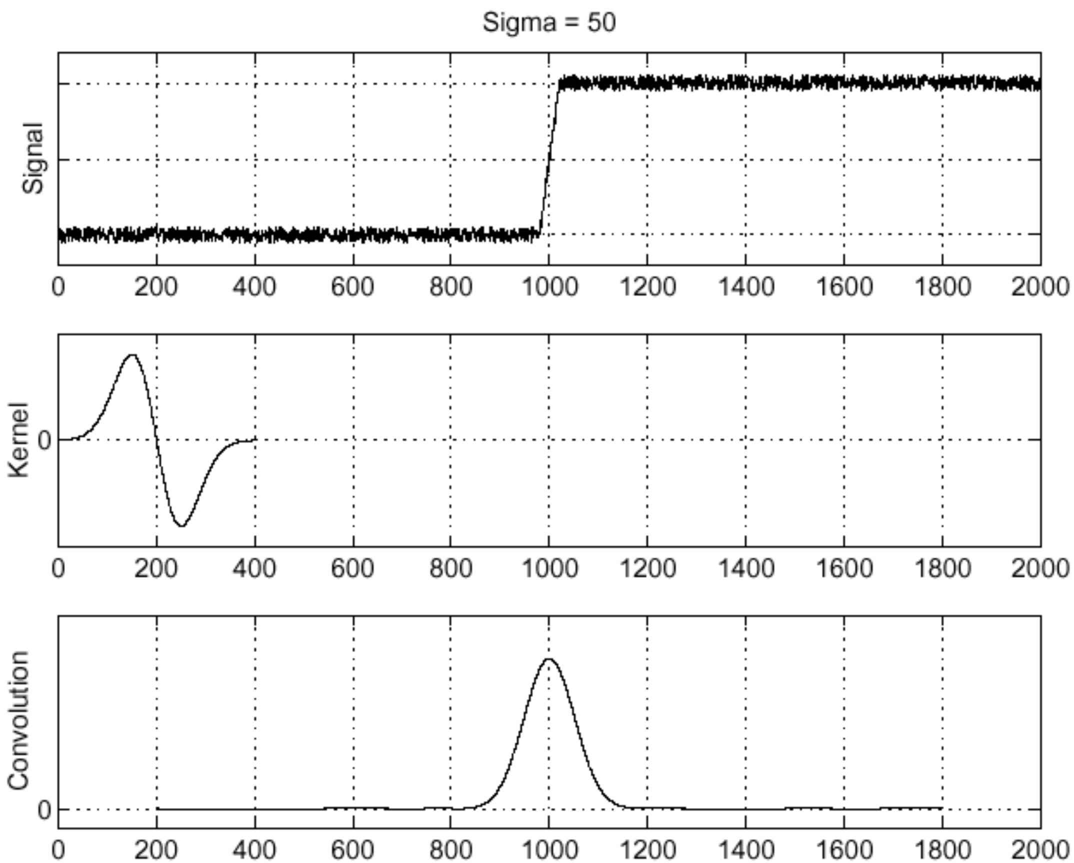
Combine Smoothing with Differentiation

$$\frac{\partial}{\partial x}(h \star f) = (\frac{\partial}{\partial x}h) \star f$$ (i.e., saves one operation)

f

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

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



Prewitt Operator

$$M_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \qquad M_y = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

M_x and M_y are approximations at (i, j)

Sobel Operator

$$M_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad M_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

M_x and M_y are approximations at (i, j)

Edge Detection Steps Using Gradient

(1) Smooth the input image ($\hat{f}(x, y) = f(x, y) * G(x, y)$)

(2) $\hat{f}_x = \hat{f}(x, y) * M_x(x, y) \longrightarrow \frac{\partial f}{\partial x}$

(3) $\hat{f}_y = \hat{f}(x, y) * M_y(x, y) \longrightarrow \frac{\partial f}{\partial y}$

(4) $magn(x, y) = |\hat{f}_x| + |\hat{f}_y|$ (i.e., sqrt is costly!)

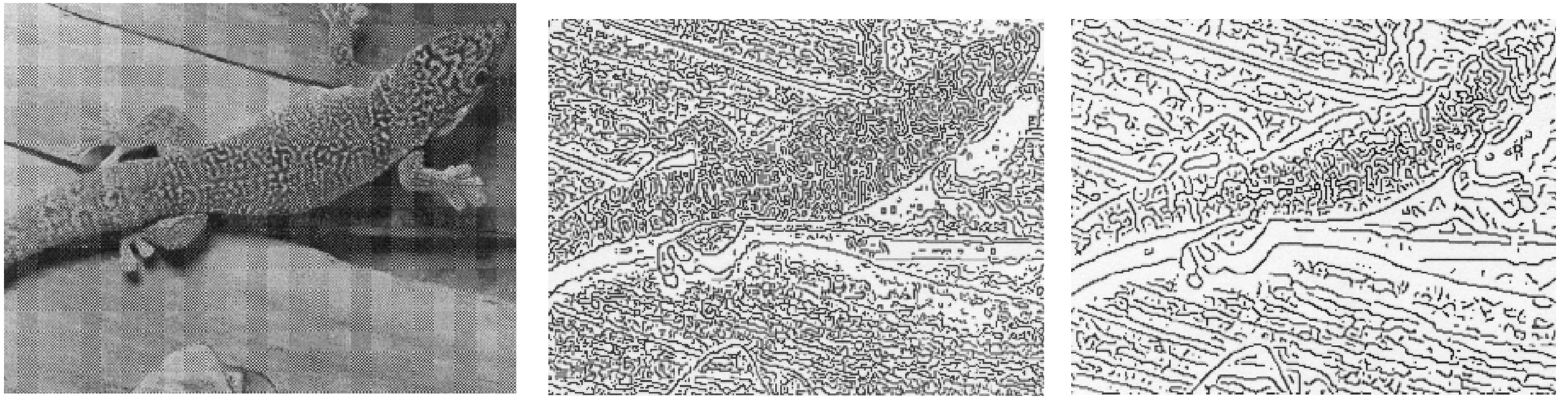
(5) $dir(x, y) = \tan^{-1}(\hat{f}_y / \hat{f}_x)$

(6) If $magn(x, y) > T$, then possible edge point

Practical Issues

- Noise suppression-localization tradeoff.
  - Smoothing depends on mask size (e.g., depends on σ for Gaussian filters).
  - Larger mask sizes reduce noise, but worsen localization (i.e., add uncertainty to the location of the edge) and vice versa.

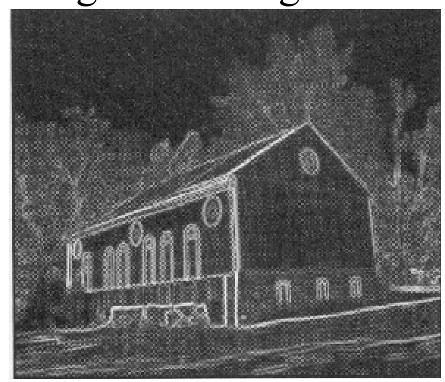
smaller mask larger mask



Practical Issues (cont’d)

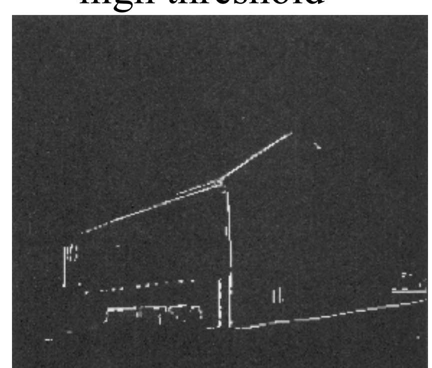
- Choice of threshold.

gradient magnitude



low threshold

high threshold



Criteria for Optimal Edge Detection

- **(1) Good detection**
  - Minimize the probability of false positives (i.e., spurious edges).
  - Minimize the probability of false negatives (i.e., missing real edges).

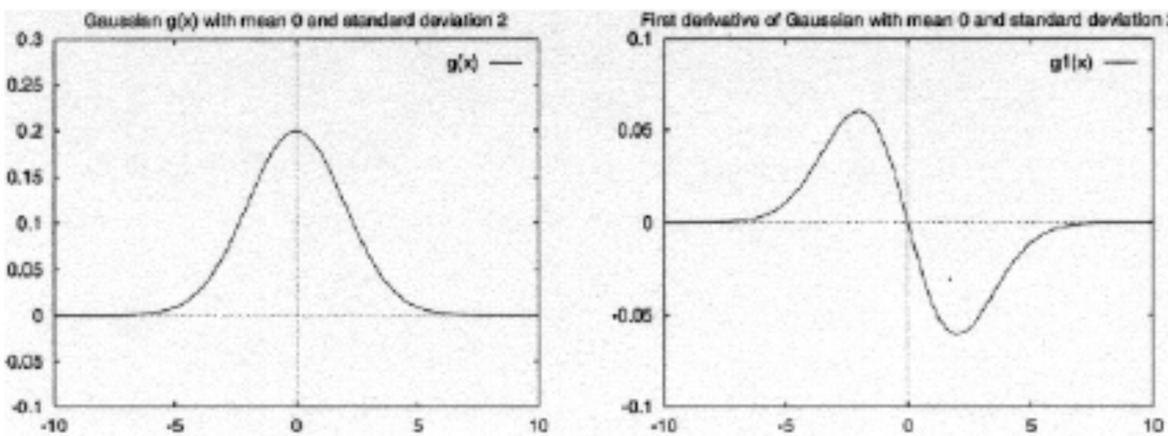
- **(2) Good localization**
  - Detected edges must be as close as possible to the true edges.

- **(3) Single response**
  - Minimize the number of local maxima around the true edge.

Canny edge detector

- Canny has shown that the **first derivative of the Gaussian** closely approximates the operator that optimizes the product of *signal-to-noise* ratio and localization.

  (i.e., analysis based on "step-edges" corrupted by "Gaussian noise")



J. Canny, ***A Computational Approach To Edge Detection***, IEEE Trans. Pattern Analysis and Machine Intelligence, 8:679-714, 1986.

Steps of Canny edge detector

Algorithm

1. Compute f_x and f_y

$$f_x = \frac{\partial}{\partial x}(f * G) = f * \frac{\partial}{\partial x} G = f * G_x$$

$$f_y = \frac{\partial}{\partial y}(f * G) = f * \frac{\partial}{\partial y} G = f * G_y$$

$G(x, y)$ is the Gaussian function

$G_x(x, y)$ is the derivate of $G(x, y)$ with respect to x: $G_x(x, y) = \frac{-x}{\sigma^2} G(x, y)$

$G_y(x, y)$ is the derivate of $G(x, y)$ with respect to y: $G_y(x, y) = \frac{-y}{\sigma^2} G(x, y)$

Steps of Canny edge detector (cont' d)

2. Compute the gradient magnitude (and direction)

$$magn(x, y) = |\hat{f}_x| + |\hat{f}_y| \quad dir(x, y) = \tan^{-1}(\hat{f}_y / \hat{f}_x)$$

3. Apply non-maxima suppression.

4. Apply hysteresis thresholding/edge linking.

Canny edge detector - example

original image

Canny edge detector – example (cont’d)

Gradient magnitude

Canny edge detector – example (cont’d)

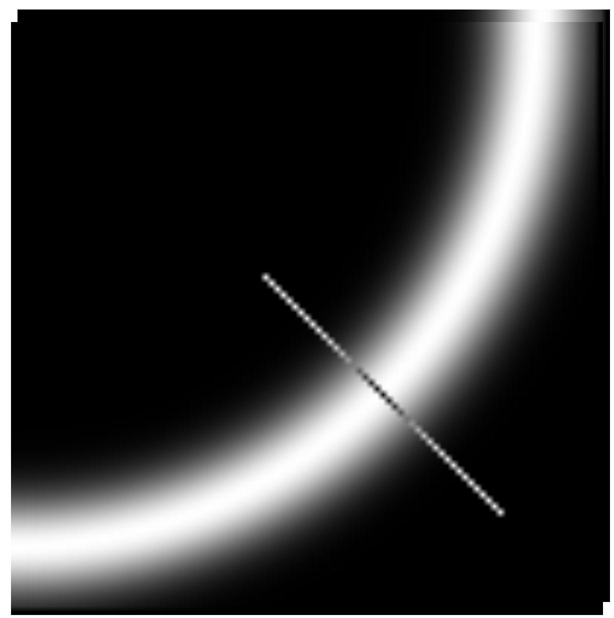
Thresholded gradient magnitude

Canny edge detector – example (cont’d)

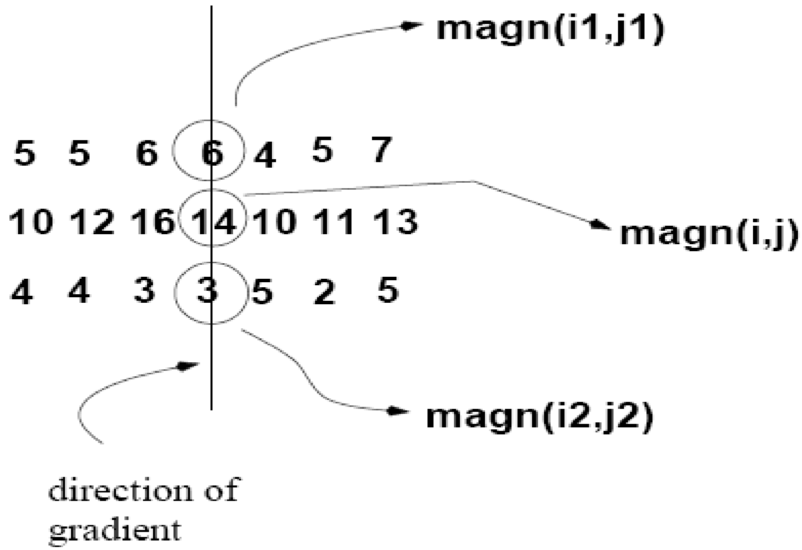
Thinning (non-maxima suppression)

Non-maxima suppression

- Check if gradient magnitude at pixel location (i,j) is local maximum along gradient direction



Non-maxima suppression (cont’d)



Algorithm

For each pixel (i,j) do:

if $magn(i, j) < magn(i_1, j_1)$ or $magn(i, j) < magn(i_2, j_2)$
then $I_N(i, j) = 0$
else $I_N(i, j) = magn(i, j)$

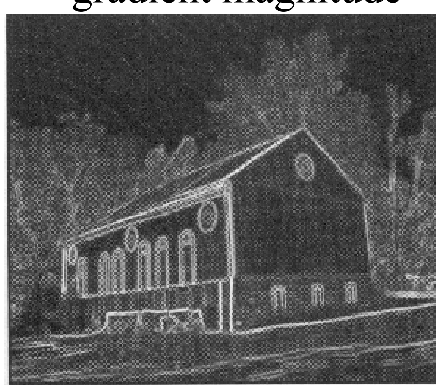
Hysteresis thresholding

- Standard thresholding:

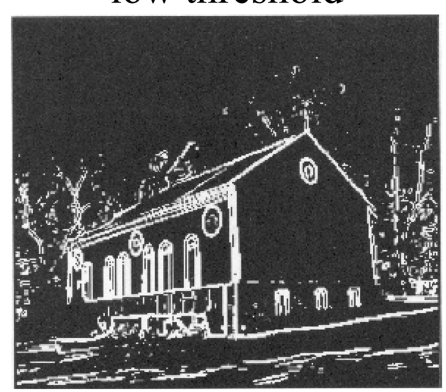
$$E(x,y) = \begin{cases} 1 & \text{if } \|\nabla f(x,y)\| > T \text{ for some threshold } T \\ 0 & \text{otherwise} \end{cases}$$

- Can only select “strong” edges.
- Does not guarantee “continuity”.

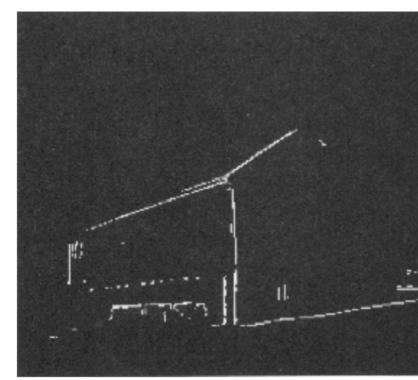
gradient magnitude



low threshold



high threshold



Hysteresis thresholding (cont'd)

- Hysteresis thresholding uses two thresholds:

  - low threshold t_l
  - high threshold t_h (usually, $t_h = 2t_l$)

- Making the assumption that important edges should be along continuous curves in the image allows us to follow a faint section of a given line and to discard a few noisy pixels that do not constitute a line but have produced large gradients.
- We begin by applying a high threshold. This marks out the edges we can be fairly sure are genuine.
- Starting from these, using the directional information derived earlier, edges can be traced through the image.
- While tracing an edge, we apply the lower threshold, allowing us to trace faint sections of edges as long as we find a starting point.