

EECE 5639 Computer Vision I

Lecture 6

Filtering, Edges.

Project 2 is out.

First Midterm: February 12

Next Class

Modern Edge Detection, More features

Correlation Example

Step 1

| | | |
|----|----|---|
| 1 | 1 | 1 |
| -1 | 2 | 1 |
| -1 | -1 | 1 |

| | | | |
|---|---|---|---|
| 2 | 2 | 2 | 3 |
| 2 | 1 | 3 | 3 |
| 2 | 2 | 1 | 2 |
| 1 | 3 | 2 | 2 |

h

| | | | | |
|---|----|---|---|---|
| 0 | 0 | 0 | | |
| 0 | 4 | 2 | 2 | 3 |
| 0 | -2 | 1 | 3 | 3 |
| | 2 | 2 | 1 | 2 |
| | 1 | 3 | 2 | 2 |

f



| | | | |
|---|--|--|--|
| 5 | | | |
| | | | |
| | | | |
| | | | |
| | | | |

$f \cdot h$

Filtering with Correlation



Original

| | | |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 0 | 0 | 0 |

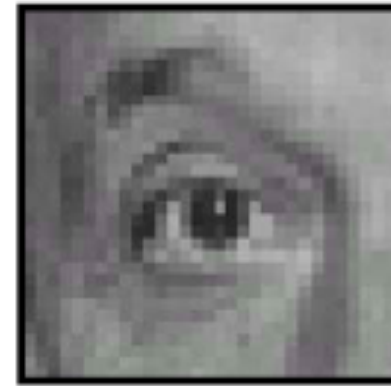
?

Filtering with Correlation



Original

| | | |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 0 | 0 | 0 |



**Filtered
(no change)**

Filtering with Correlation



Original

| | | |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 0 | 1 |
| 0 | 0 | 0 |

?

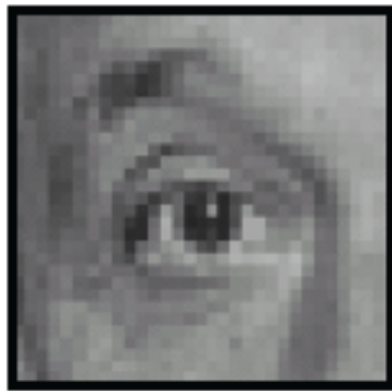
Filtering with Correlation



| | | |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 0 | 1 |
| 0 | 0 | 0 |





Filtering with Correlation



Original

$$* \left(\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \right) =$$

Filtering with Correlation


$$* \left(\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \right) =$$


Original

Sharpening filter
(accentuates edges)

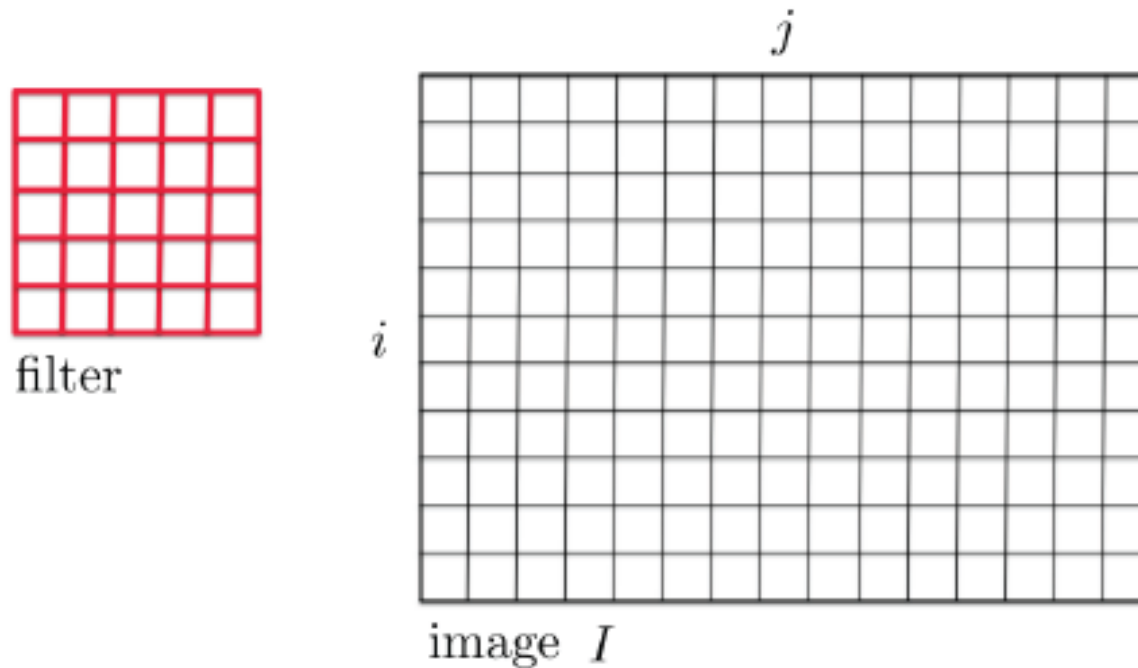
Efficient Implementation

Both, the BOX filter and the Gaussian filter are separable:

First convolve each row with a 1D filter

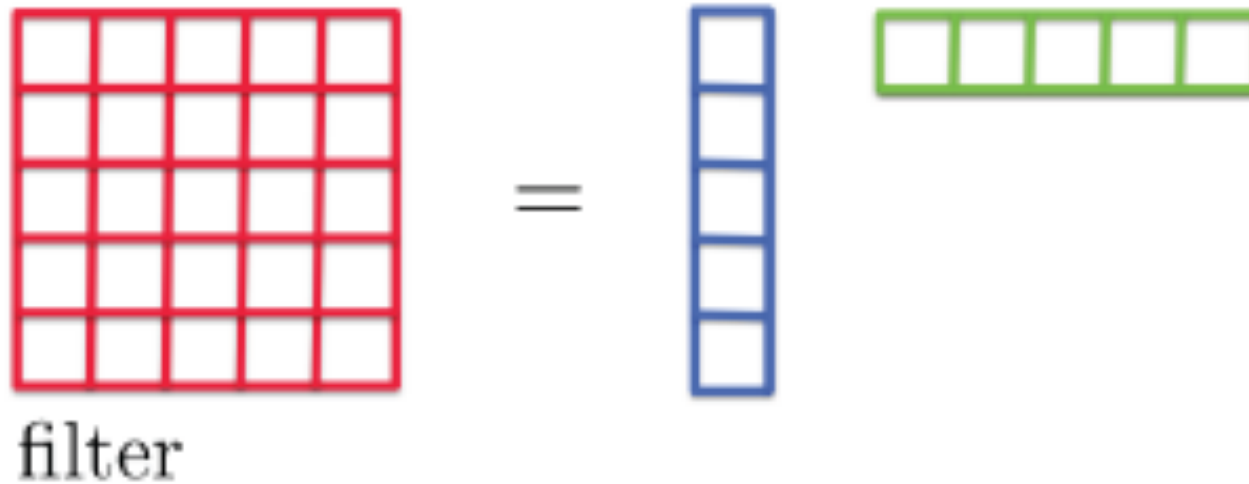
Then convolve each column with a 1D filter.

Separable Filters

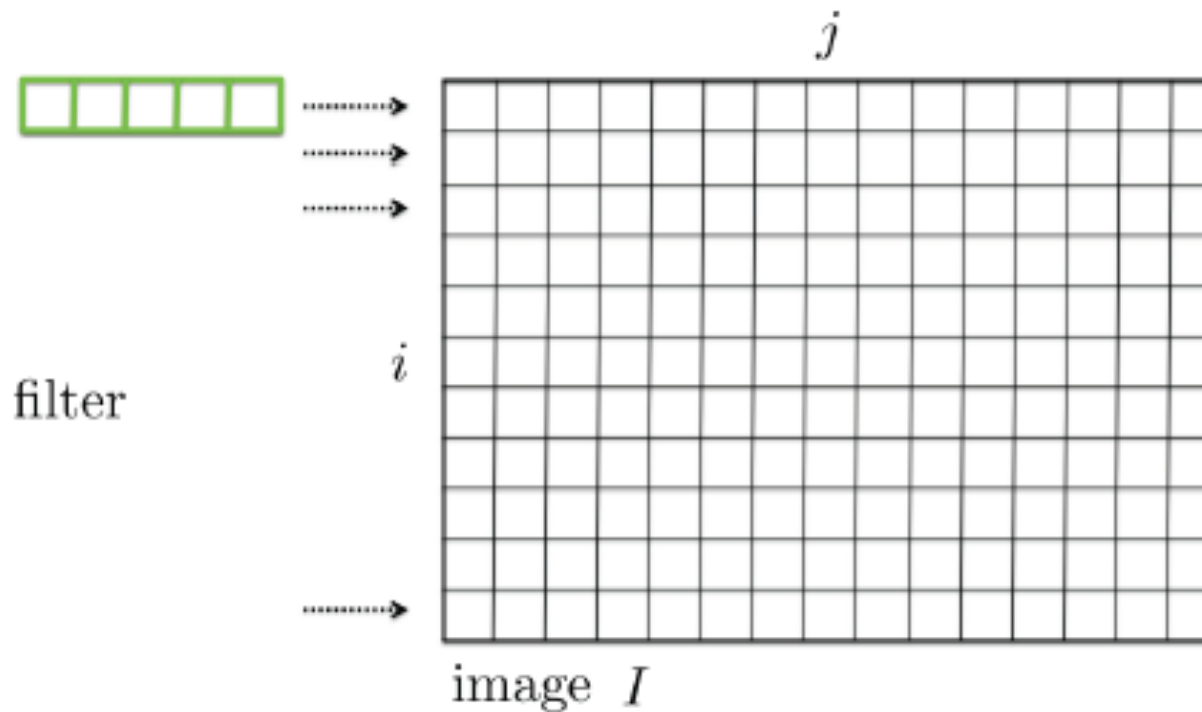


a $K \times K$ filter: K^2 operations/pixel

Separable Filters

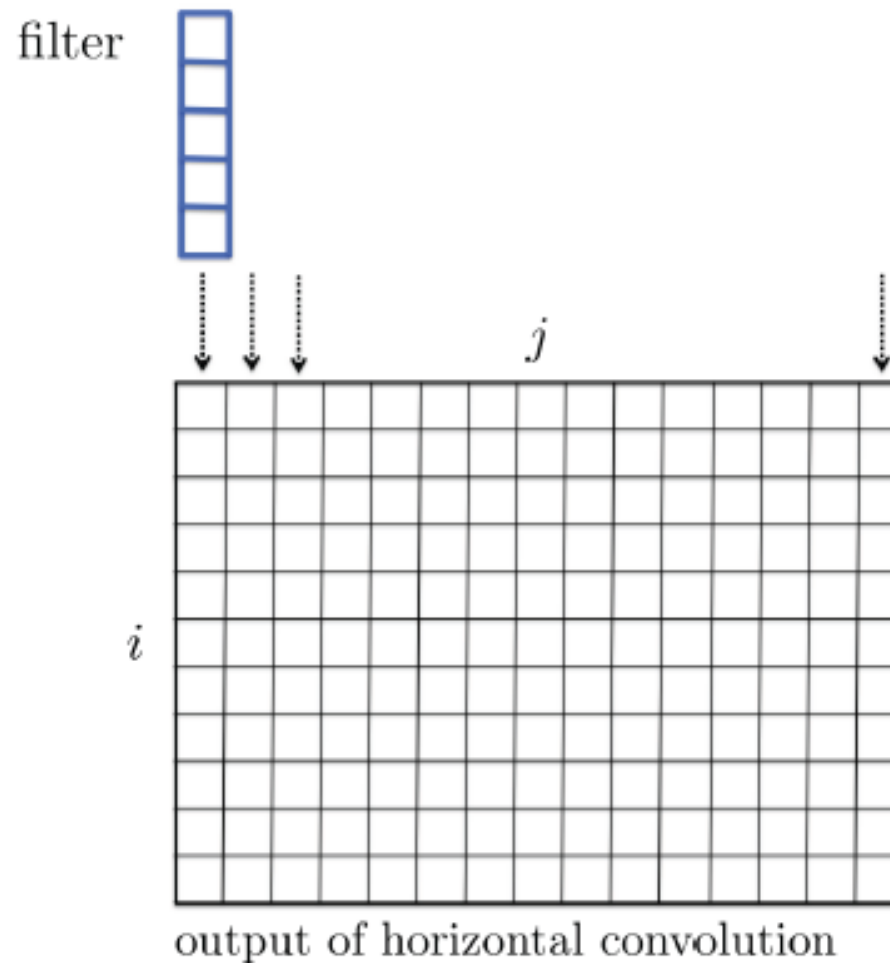


Separable Filters



a 1xK filter: K operations/pixel

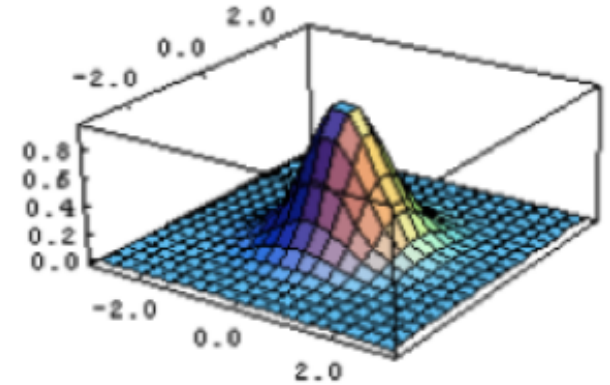
Separable Filters



for a $K \times 1$ filter: K operations/pixel

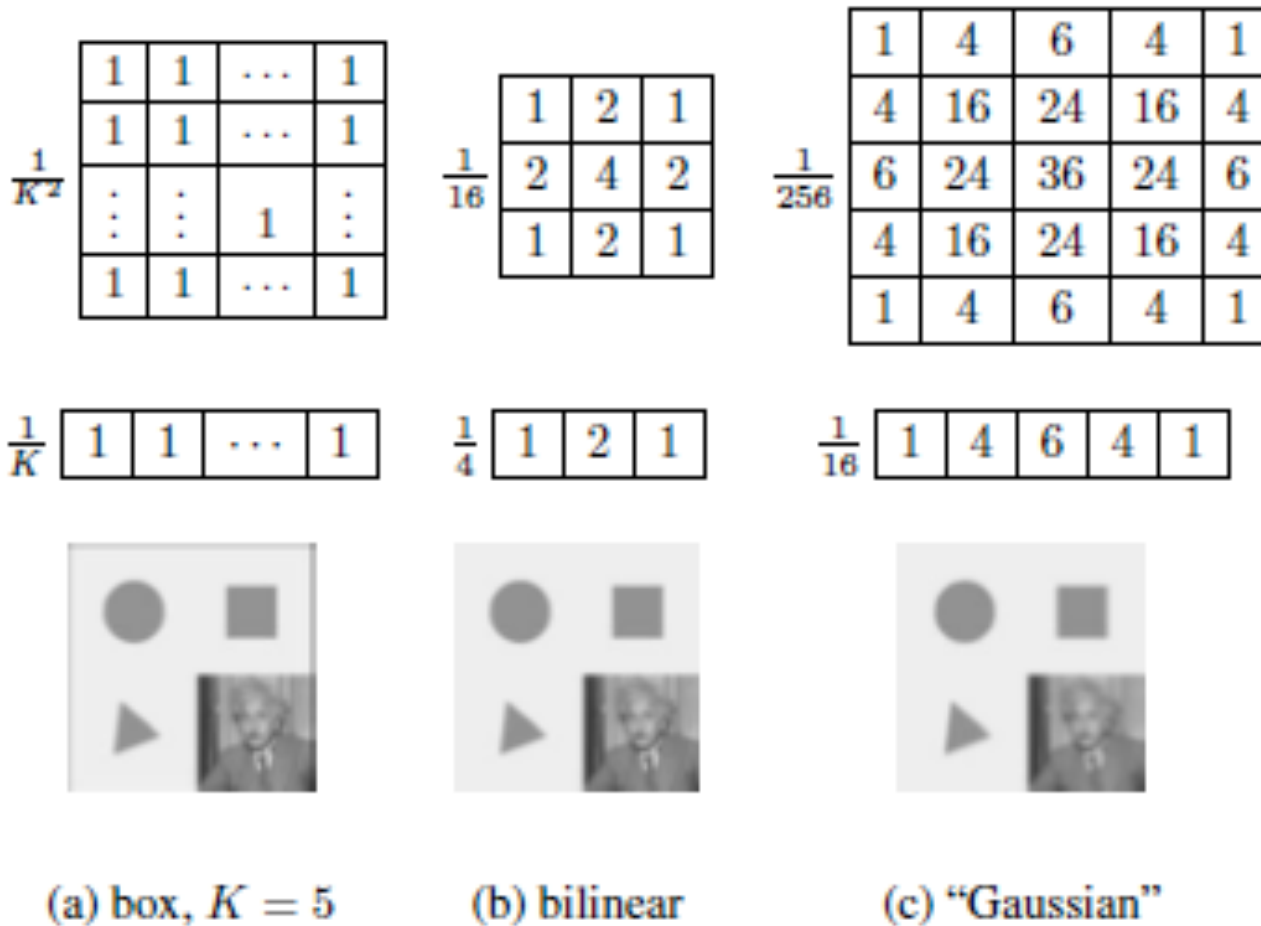
Gaussian Filter is Separable

$$f(x, y) = \frac{1}{K} e^{-\frac{x^2 + y^2}{2\sigma^2}}$$



$$f(x, y) = \left(\frac{1}{\sqrt{K}} e^{-\frac{x^2}{2\sigma^2}} \right) \left(\frac{1}{\sqrt{K}} e^{-\frac{y^2}{2\sigma^2}} \right)$$

Separable Filters



Separable Filters

How do we know a filter is separable?

By inspection

Looking at the analytic form of it

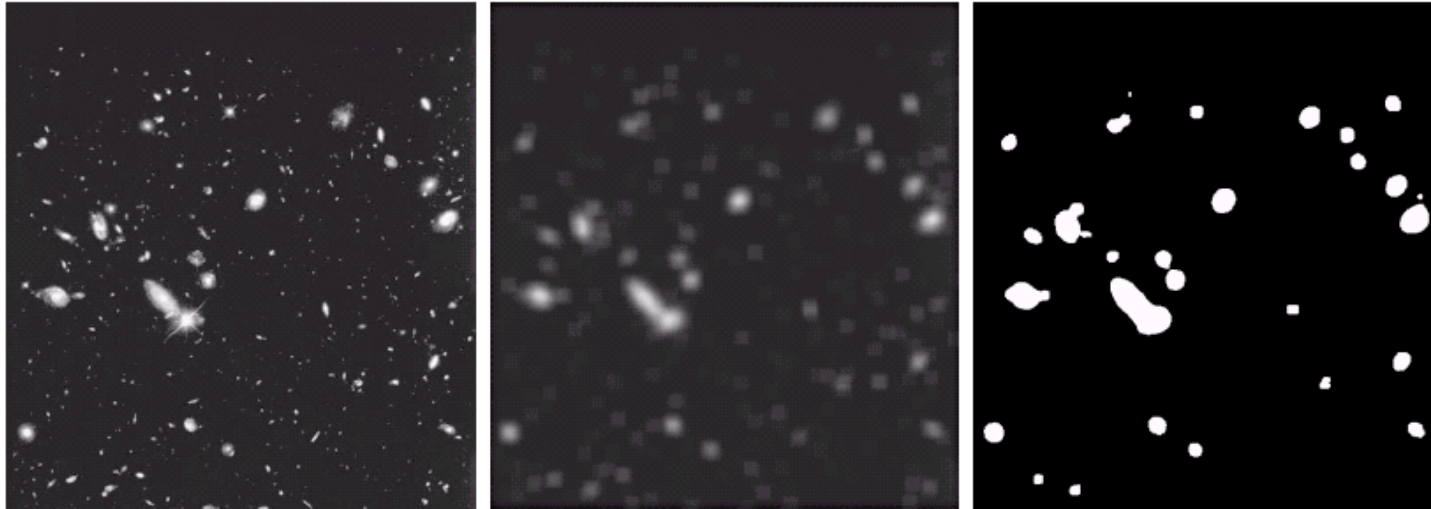
Looking at its Singular Value Decomposition (SVD)

$$F = U\Sigma V^T$$

$$\Sigma = \begin{bmatrix} \sigma_1 & 0 & 0 & \dots & 0 \\ 0 & \sigma_2 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \dots & \vdots \\ 0 & 0 & 0 & \dots & \sigma_K \end{bmatrix} \quad U = \begin{bmatrix} | & | & \dots & | \\ u_1 & u_2 & \dots & u_K \\ | & | & \dots & | \end{bmatrix} \quad V = \begin{bmatrix} | & | & \dots & | \\ v_1 & v_2 & \dots & v_K \\ | & | & \dots & | \end{bmatrix}$$

$$F = \sum_i \sigma_i u_i \cdot v_i^T$$

if one singular value is non-zero.



a b c

FIGURE 3.36 (a) Image from the Hubble Space Telescope. (b) Image processed by a 15×15 averaging mask. (c) Result of thresholding (b). (Original image courtesy of NASA.)

Effects of increasing mask size

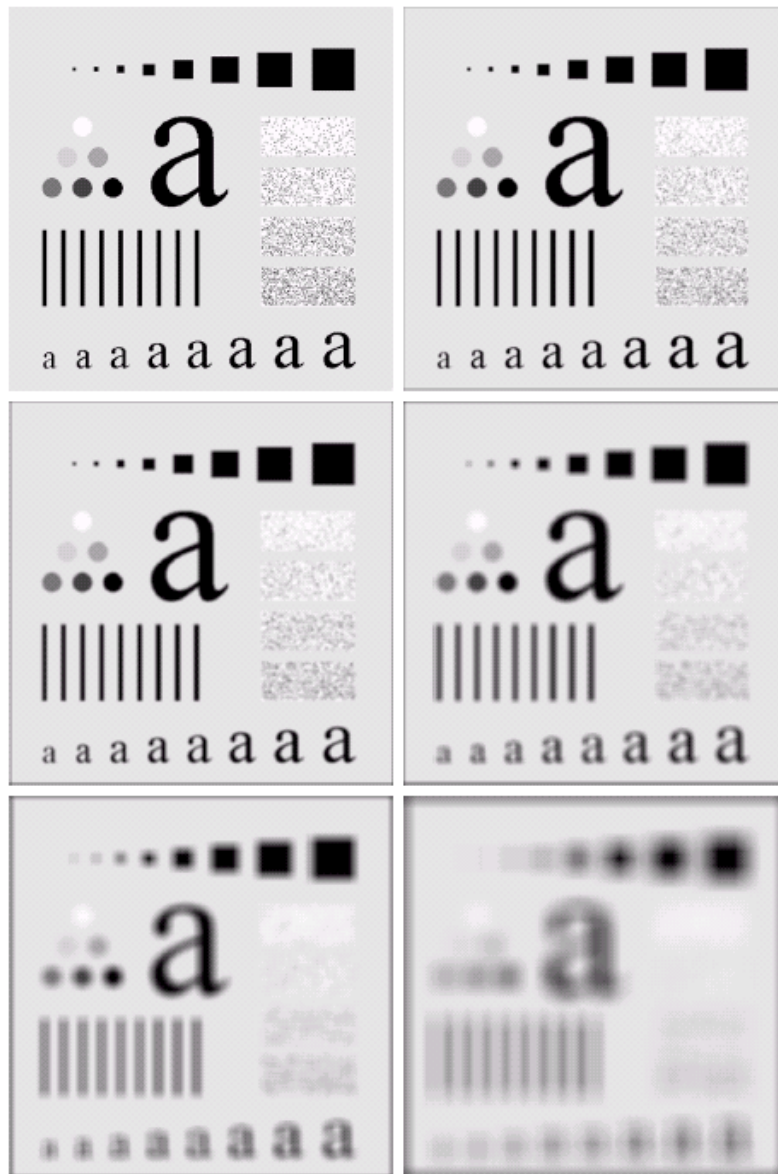
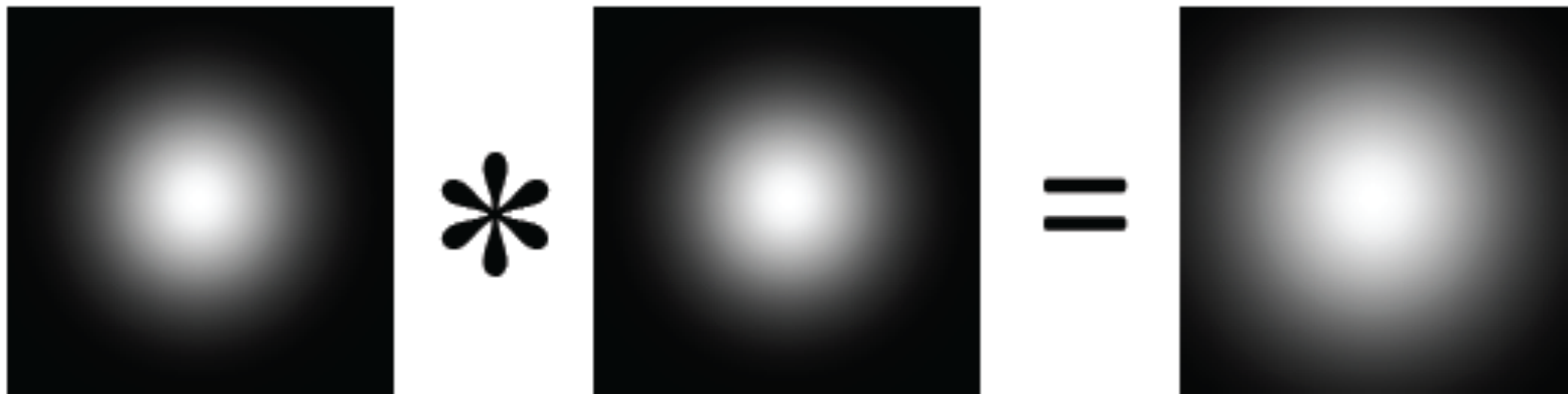


FIGURE 3.35 (a) Original image, of size 500×500 pixels. (b)–(f) Results of smoothing with square averaging filter masks of sizes $n = 3, 5, 9, 15$, and 35 , respectively. The black squares at the top are of sizes 3, 5, 9, 15, 25, 35, 45, and 55 pixels, respectively; their borders are 25 pixels apart. The letters at the bottom range in size from 10 to 24 points, in increments of 2 points; the large letter at the top is 60 points. The vertical bars are 5 pixels wide and 100 pixels high; their separation is 20 pixels. The diameter of the circles is 25 pixels, and their borders are 15 pixels apart; their gray levels range from 0% to 100% black in increments of 20%. The background of the image is 10% black. The noisy rectangles are of size 50×120 pixels.

Cascading

Convolving twice with a Gaussian Kernel (σ), is the same as convolving once with a Gaussian Kernel ($\sqrt{2}\sigma$)



Efficient Implementation: Integral Image

If an image is going to be repeatedly convolved with different box filters a pre-computed summed area table can save computations for future use.

$$s(i, j) = \sum_{k=0}^i \sum_{l=0}^j f(k, l)$$

| | | | | |
|---|---|---|---|---|
| 3 | 2 | 7 | 2 | 3 |
| 1 | 5 | 1 | 3 | 4 |
| 5 | 1 | 3 | 5 | 1 |
| 4 | 3 | 2 | 1 | 6 |
| 2 | 4 | 1 | 4 | 8 |

Efficient Implementation: Integral Image

If an image is going to be repeatedly convolved with different box filters a pre-computed summed area table can save computations for future use.

$$s(i, j) = \sum_{k=0}^i \sum_{l=0}^j f(k, l)$$

| | | | | |
|---|---|---|---|---|
| 3 | 2 | 7 | 2 | 3 |
| 1 | 5 | 1 | 3 | 4 |
| 5 | 1 | 3 | 5 | 1 |
| 4 | 3 | 2 | 1 | 6 |
| 2 | 4 | 1 | 4 | 8 |

| | | | | |
|----|----|----|----|----|
| 3 | 5 | 12 | 14 | 17 |
| 4 | 11 | 19 | 24 | 31 |
| 9 | 17 | 28 | 38 | 46 |
| 13 | 24 | 37 | 48 | 62 |
| 15 | 30 | 44 | 59 | 81 |

$$s(i, j) = s(i - 1, j) + s(i, j - 1) - s(i - 1, j - 1) + f(k, l)$$

Efficient Implementation: Integral Image

If an image is going to be repeatedly convolved with different box filters a pre-computed summed area table can save computations for future use.

$$s(i, j) = \sum_{k=0}^i \sum_{l=0}^j f(k, l)$$

| | | | | |
|---|---|---|---|---|
| 3 | 2 | 7 | 2 | 3 |
| 1 | 5 | 1 | 3 | 4 |
| 5 | 1 | 3 | 5 | 1 |
| 4 | 3 | 2 | 1 | 6 |
| 2 | 4 | 1 | 4 | 8 |

(a) $S = 24$

| | | | | |
|----|----|----|----|----|
| 3 | 5 | 12 | 14 | 17 |
| 4 | 11 | 19 | 24 | 31 |
| 9 | 17 | 28 | 38 | 46 |
| 13 | 24 | 37 | 48 | 62 |
| 15 | 30 | 44 | 59 | 81 |

(b) $s = 28$

| | | | | |
|----|----|----|----|----|
| 3 | 5 | 12 | 14 | 17 |
| 4 | 11 | 19 | 24 | 31 |
| 9 | 17 | 28 | 38 | 46 |
| 13 | 24 | 37 | 48 | 62 |
| 15 | 30 | 44 | 59 | 81 |

(c) $S = 24$

$$S = 48 - 14 - 13 + 3$$

Limitations of averaging

Signal frequencies shared with noise are lost, resulting in blurring.

Impulsive noise is diffused but not removed.

It spreads pixel values, resulting in blurring.

Non-linear Filtering

Replace each pixel with the **MEDIAN** value of all the pixels in the neighborhood.

Example:

I

| | | | | | |
|----|----|----|----|----|----|
| 10 | 11 | 10 | 0 | 0 | 1 |
| 9 | 10 | 11 | 1 | 0 | 1 |
| 10 | 9 | 10 | 0 | 2 | 1 |
| 11 | 10 | 9 | 10 | 9 | 11 |
| 9 | 10 | 11 | 9 | 99 | 11 |
| 10 | 9 | 9 | 11 | 10 | 10 |

O

| | | | | | |
|---|----|---|---|---|---|
| X | X | X | X | X | X |
| X | 10 | | | | X |
| X | | | | | X |
| X | | | | | X |
| X | | | | | X |
| X | X | X | X | X | X |

10, 11, 10, 9, 10, 11, 10, 9,
10

sort
→

9, 9, 10, 10, 10, 10, 10, 11, 11

median

Example:

I

| | | | | | |
|----|----|----|----|----|----|
| 10 | 11 | 10 | 0 | 0 | 1 |
| 9 | 10 | 11 | 1 | 0 | 1 |
| 10 | 9 | 10 | 0 | 2 | 1 |
| 11 | 10 | 9 | 10 | 9 | 11 |
| 9 | 10 | 11 | 9 | 99 | 11 |
| 10 | 9 | 9 | 11 | 10 | 10 |

O

| | | | | | |
|---|----|----|---|---|---|
| X | X | X | X | X | X |
| X | 10 | 10 | 1 | 1 | X |
| X | | | | | X |
| X | | | | | X |
| X | | | | | X |
| X | X | X | X | X | X |

11, 10, 0, 10, 11, 1, 9, 10
, 0

sort

0, 0, 1, 9, 10, 10, 10, 11, 11

median

Example:

I

| | | | | | |
|----|----|----|----|----|----|
| 10 | 11 | 10 | 0 | 0 | 1 |
| 9 | 10 | 11 | 1 | 0 | 1 |
| 10 | 9 | 10 | 0 | 2 | 1 |
| 11 | 10 | 9 | 10 | 9 | 11 |
| 9 | 10 | 11 | 9 | 99 | 11 |
| 10 | 9 | 9 | 11 | 10 | 10 |

O

| | | | | | |
|---|----|---|---|----|---|
| X | X | X | X | X | X |
| X | 10 | | | | X |
| X | | | | | X |
| X | | | 9 | | X |
| X | | | | 10 | X |
| X | X | X | X | X | X |

10, 9, 11, 9, 99, 11, 11, 10, 10

sort

9, 9, 10, 10, 10, 11, 11, 11, 99

median

Median Filter Properties

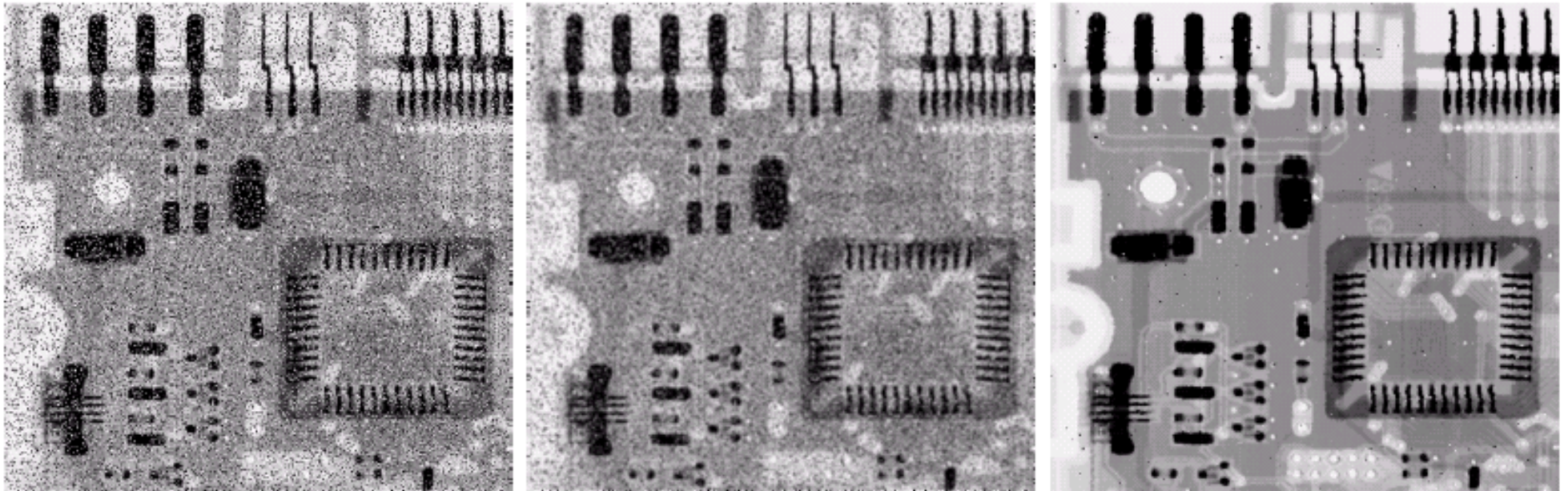
Non-linear

Does not spread the noise

Can remove spike noise

Expensive to run

Median Filter



a b c

FIGURE 3.37 (a) X-ray image of circuit board corrupted by salt-and-pepper noise. (b) Noise reduction with a 3×3 averaging mask. (c) Noise reduction with a 3×3 median filter. (Original image courtesy of Mr. Joseph E. Pascente, Lixi, Inc.)

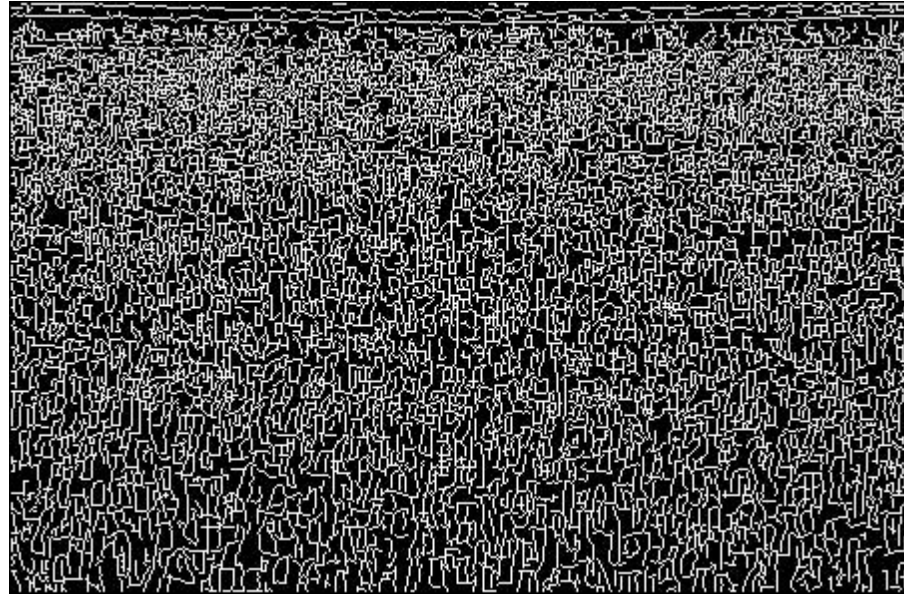
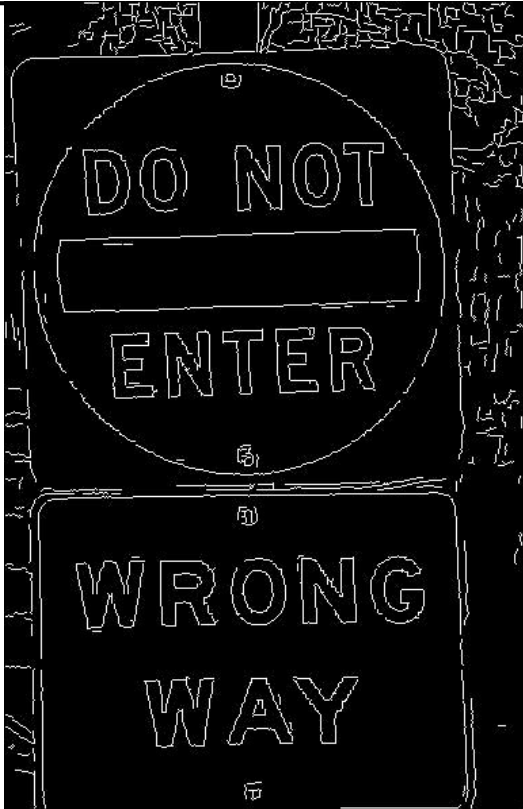
Image Features

Edges

What is an Image Feature?



What is an Image Feature?



What is an Image Feature?

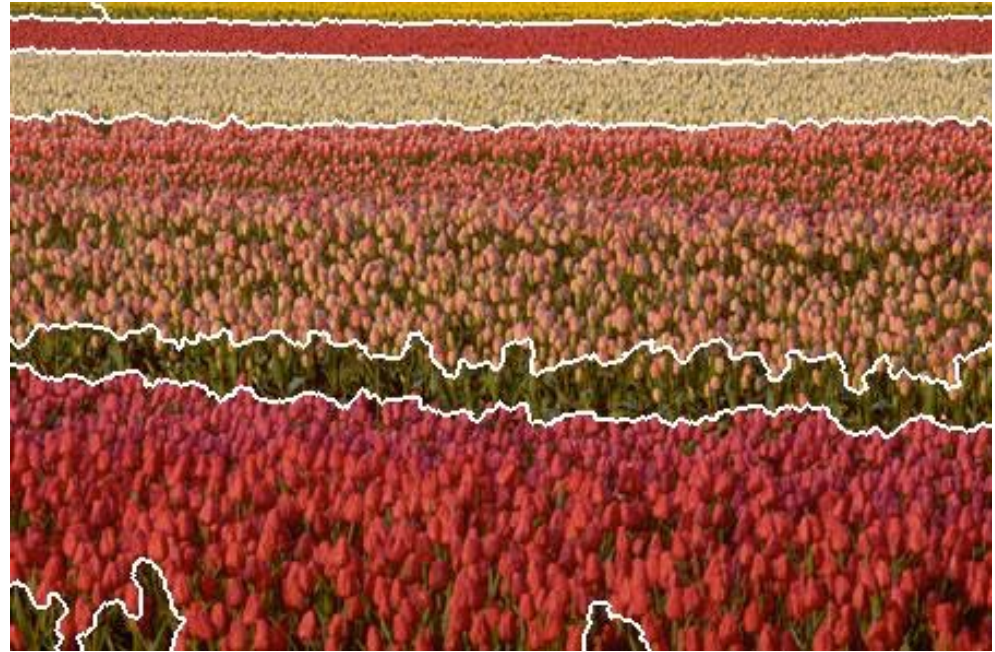
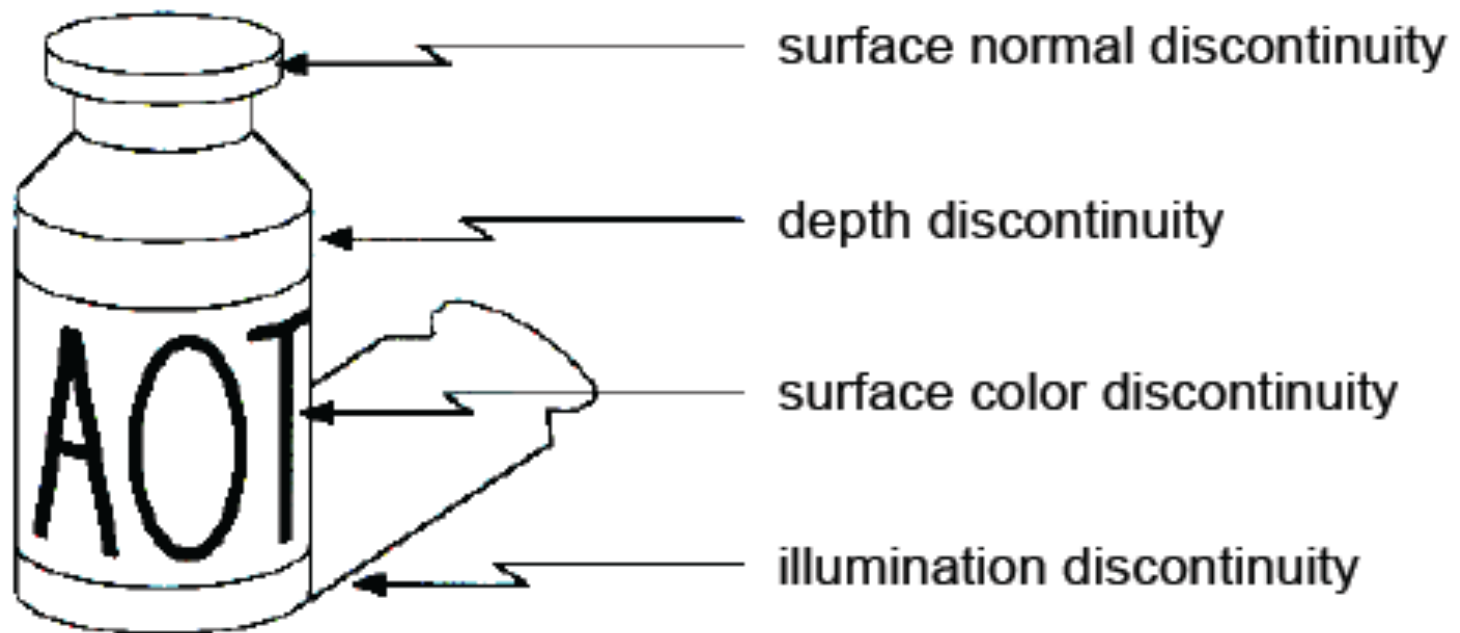


Image Features

Local, meaningful, detectable parts of the image.

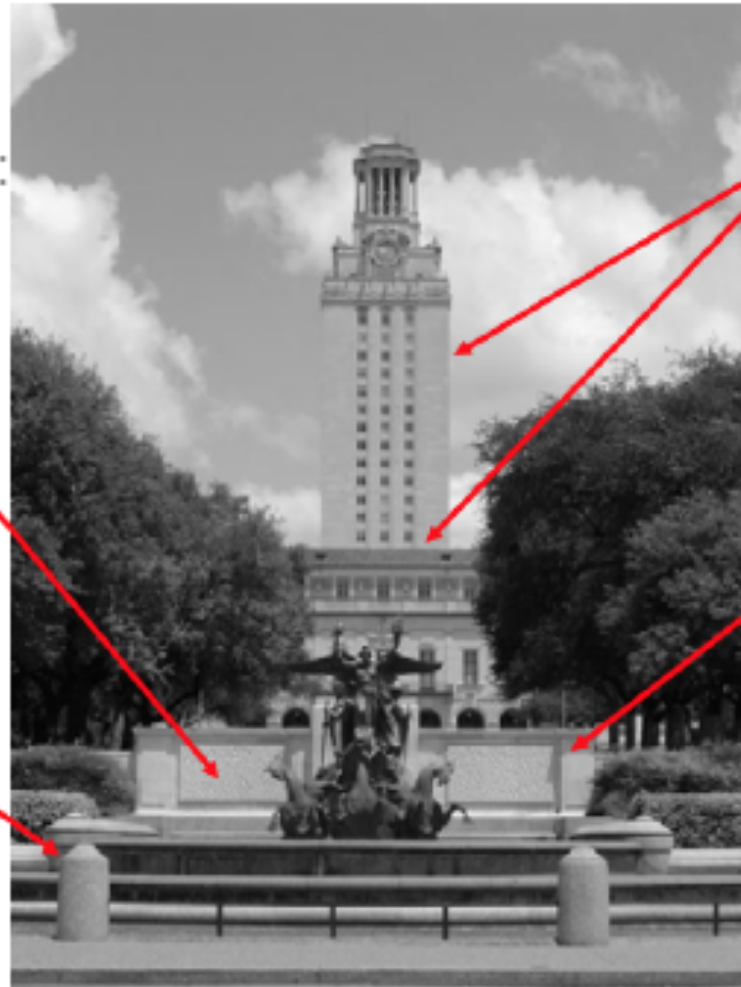
Origin of Edges



Origin of Edges

Reflectance change:
appearance
information, texture

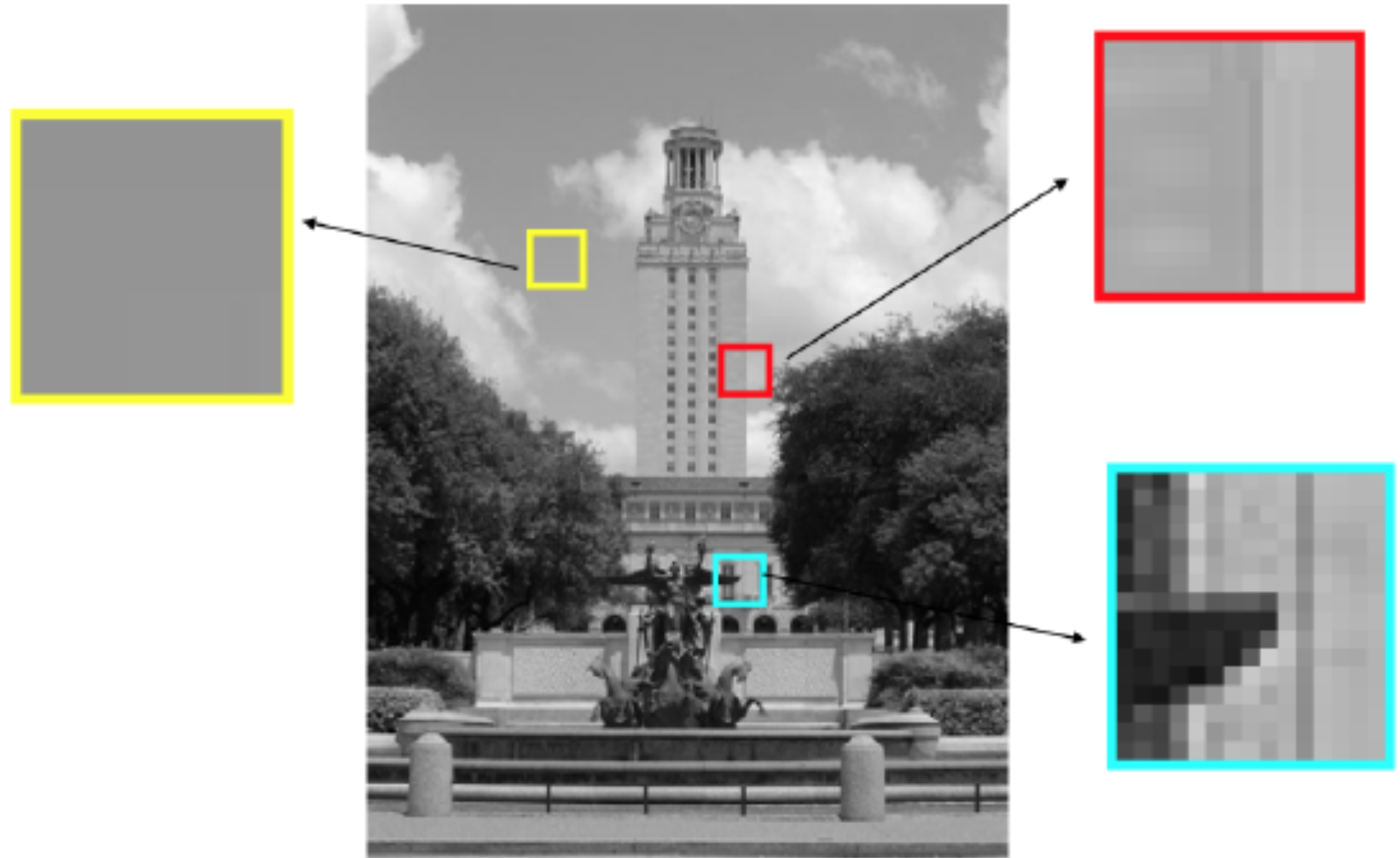
Change in surface
orientation: shape



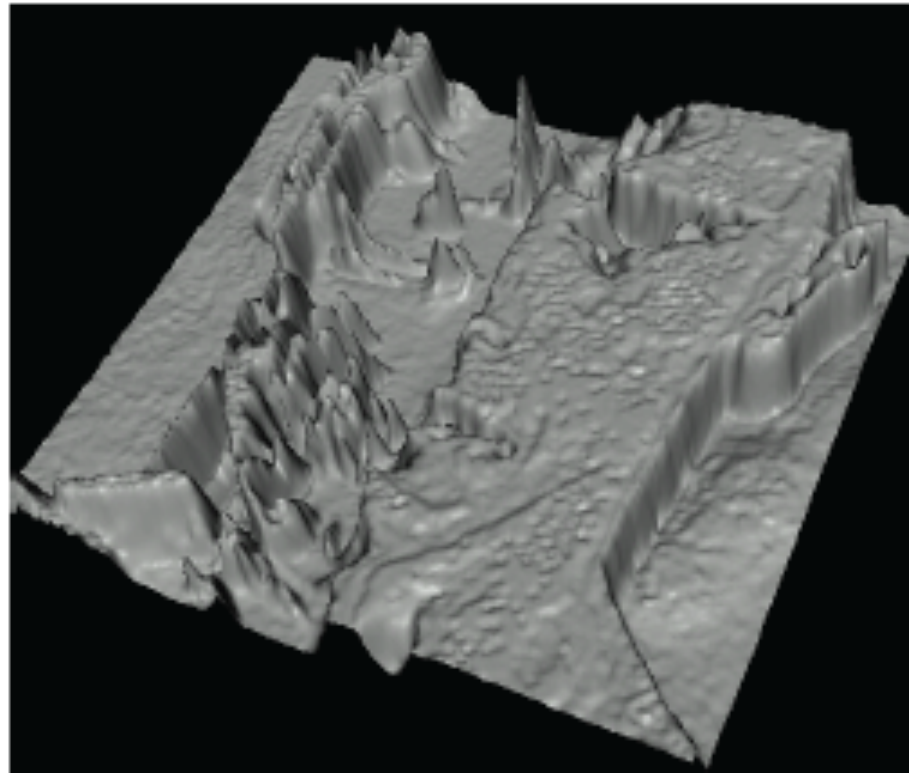
Depth discontinuity:
object boundary

Cast shadows

Looking locally:



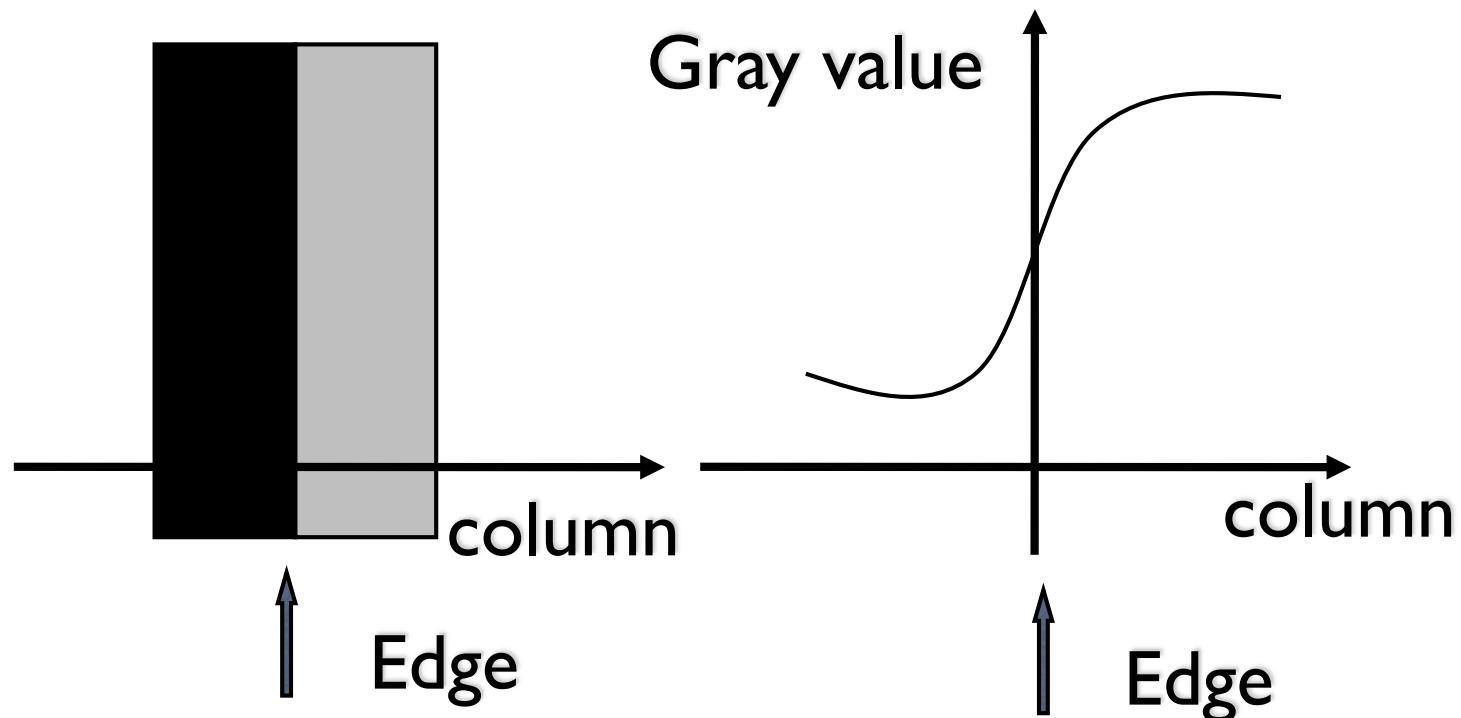
Images as Functions



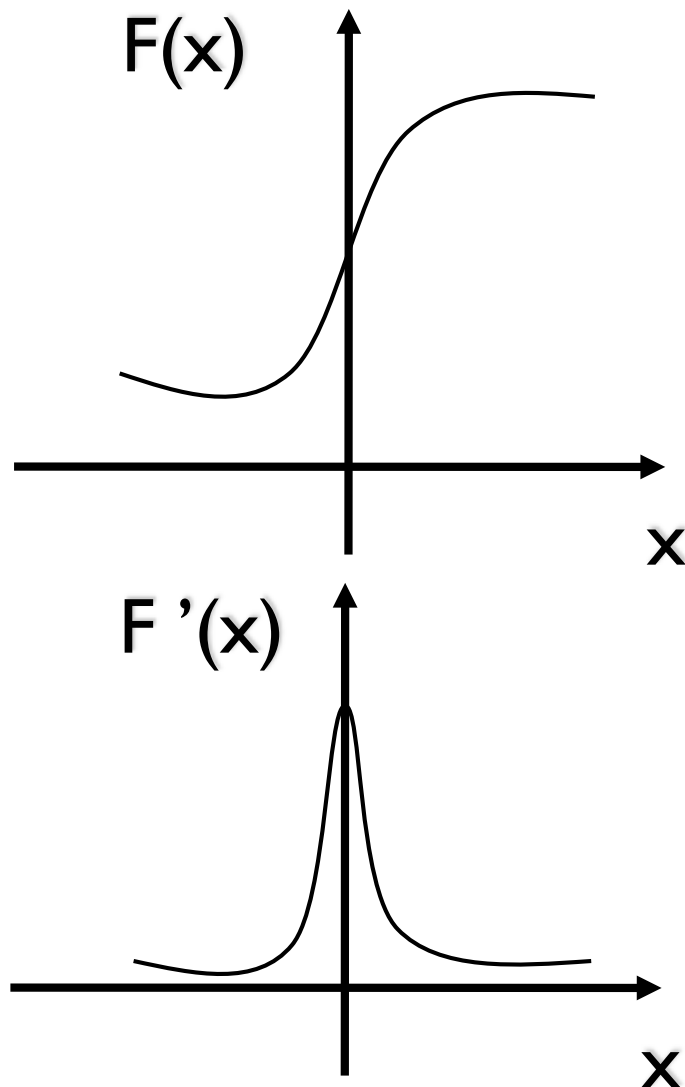
Edges look like steep cliffs

Edges

They happen at places where the image values exhibit sharp variation



Edge detection (1D)



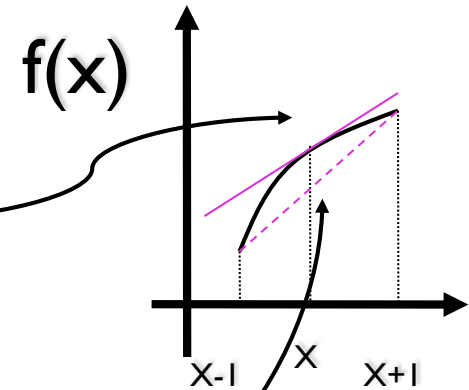
Edge= sharp variation



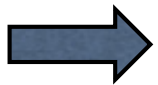
Large first derivative

Digital Approximation

$$\frac{df(x)}{dx} = \lim_{\Delta x \rightarrow 0} \frac{f(x + \Delta x) - f(x)}{\Delta x}$$



$$\frac{df(x)}{dx} \cong \frac{f(x+1) - f(x-1)}{2}$$

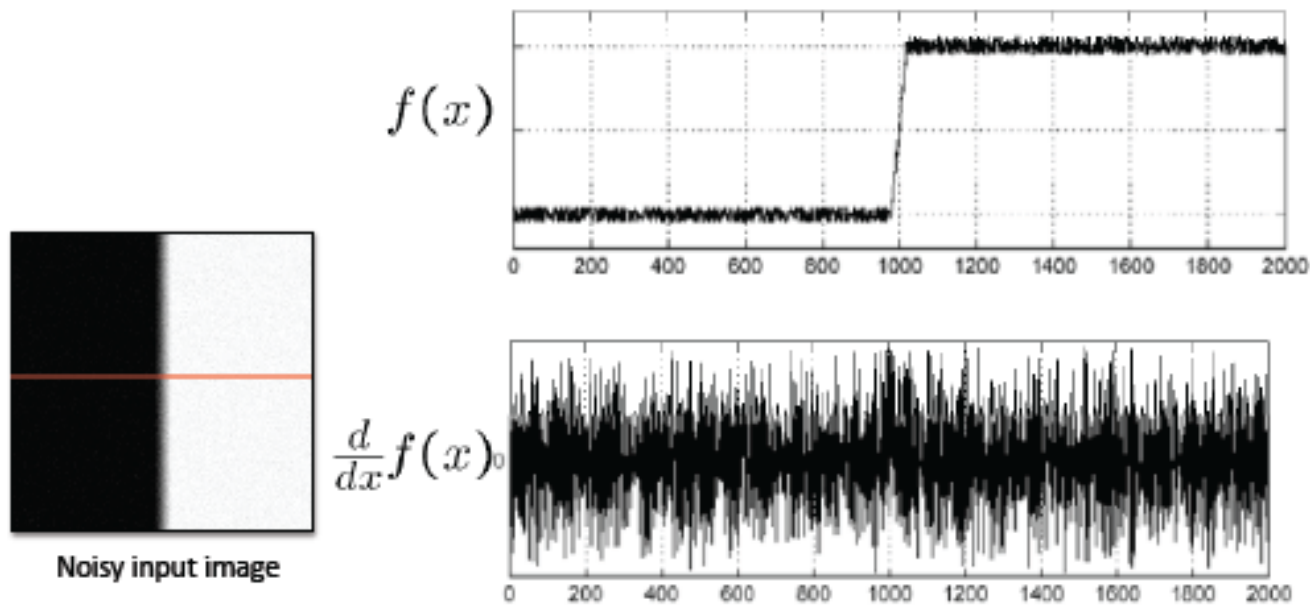


Use mask:

| | | |
|----|---|---|
| -1 | 0 | 1 |
|----|---|---|

Effect of Noise

Consider a single row or column of the image:



Edge detection & Image Noise

$$I(x) = \hat{I}(x) + N(x) \quad N(x) \sim N(0, \sigma) \text{ i.i.d}$$

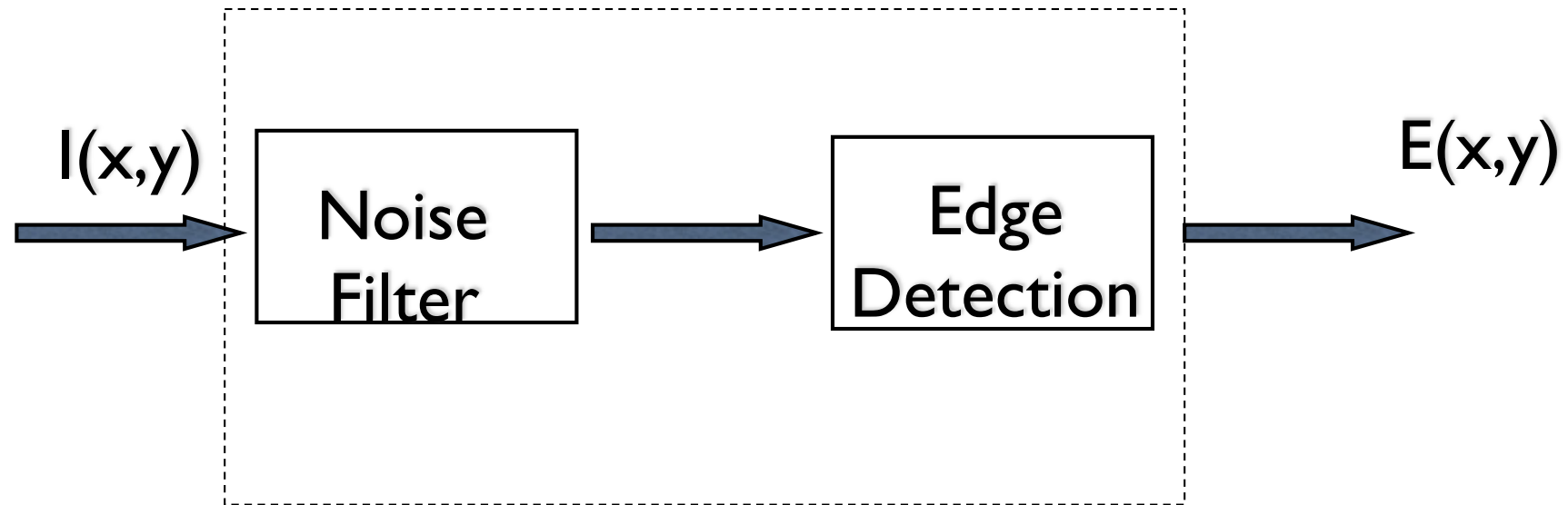
Taking differences:

$$\begin{aligned} I'(x) &\cong \hat{I}(x+1) + N(x+1) - (\hat{I}(x-1) + N(x-1)) = \\ &= \underbrace{(\hat{I}(x+1) - \hat{I}(x-1))}_{\hat{I}'(x)} + \underbrace{(N(x+1) - N(x-1))}_{N_d(x)} \end{aligned}$$

Output noise: $E(N_d(x)) = 0$

$$\begin{aligned} E(N_d^2(x)) &= E(N^2(x+1) + N^2(x-1) + 2N(x+1)N(x-1)) = \\ &= \sigma^2 + \sigma^2 + 0 = 2\sigma^2 \end{aligned} \quad \longrightarrow \quad \text{Increases noise !!}$$

Noise cleaning and Edge Detection

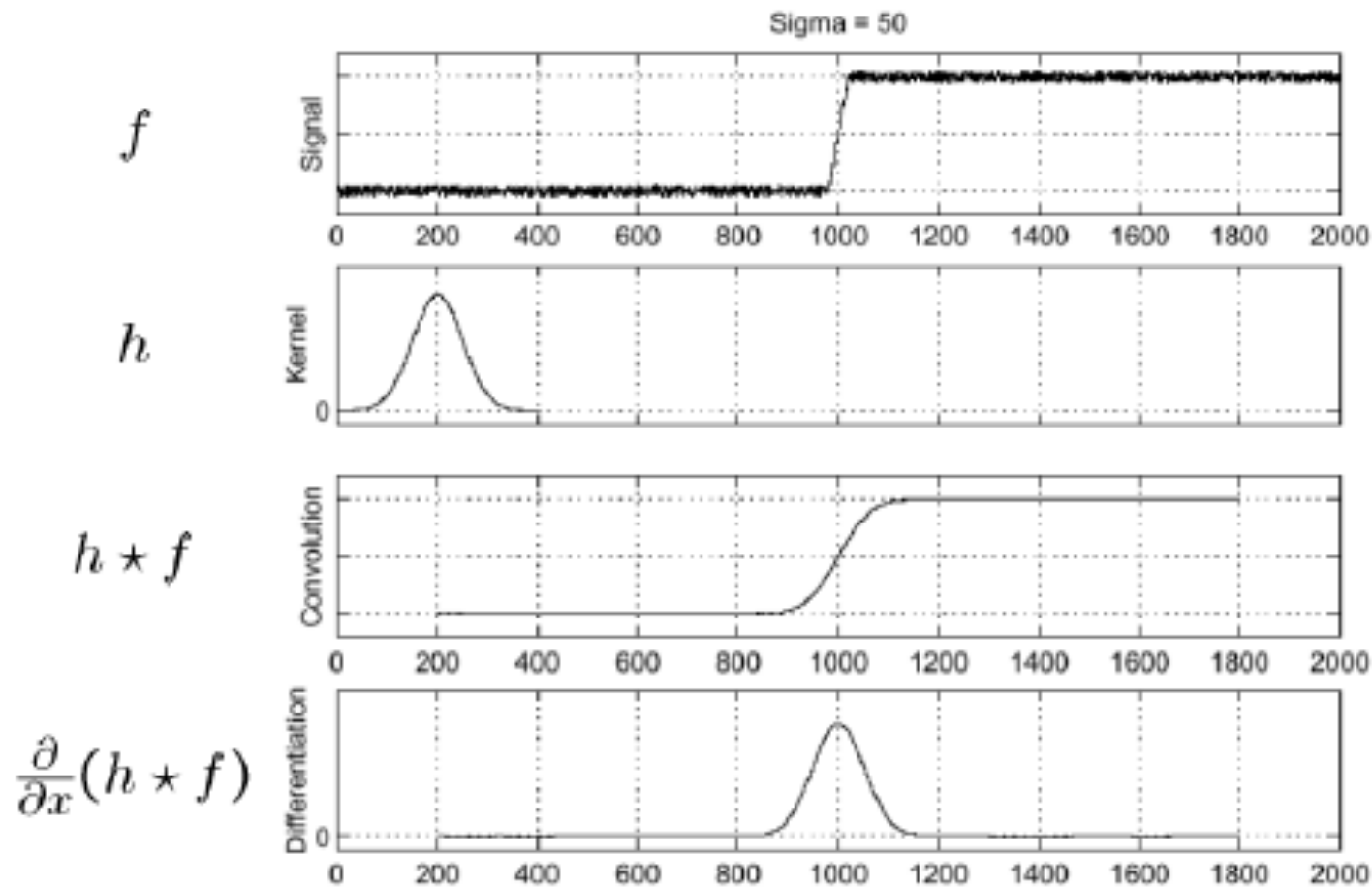


Combine Linear Filters

Dealing with Noise

Smooth first with a Gaussian, then take derivative:

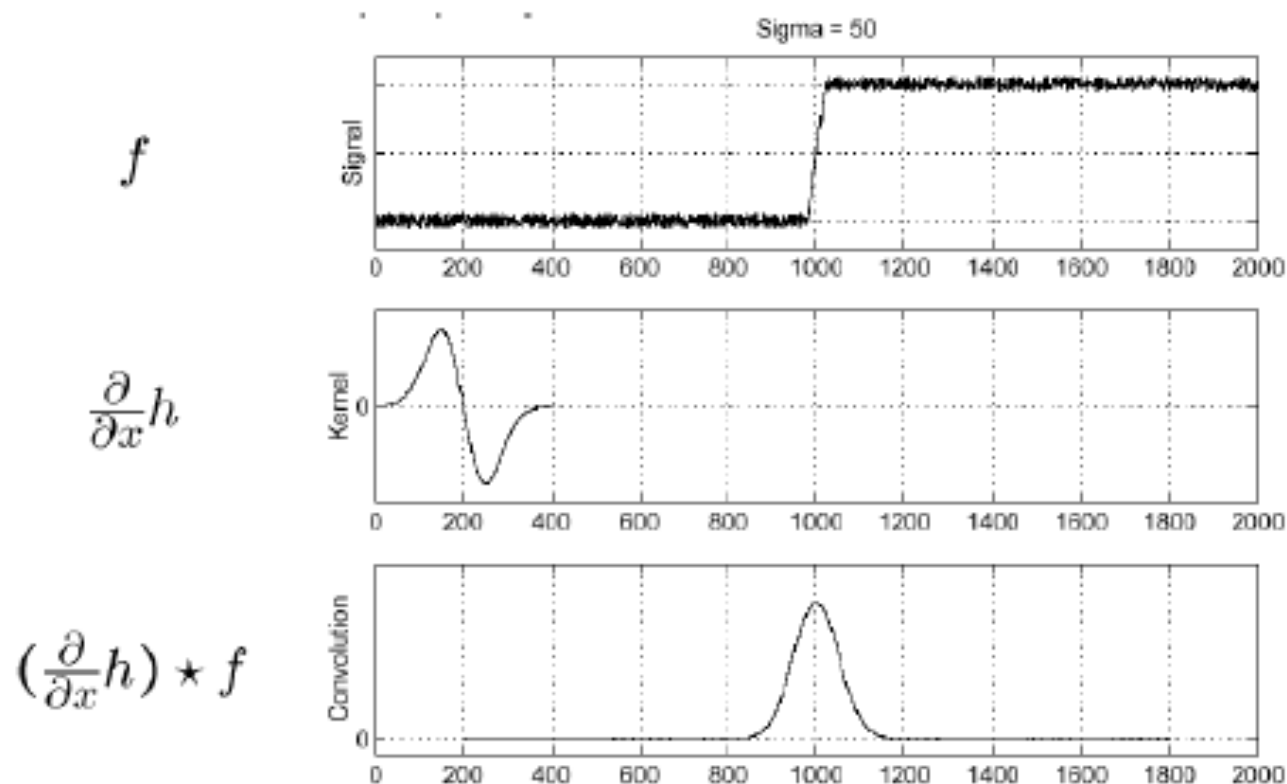
$$\frac{\partial}{\partial x}(h * f)$$



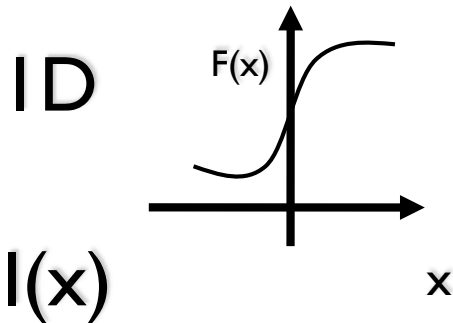
Derivative Theorem of Convolution

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

It saves one operation!



Edge Detection (2D)

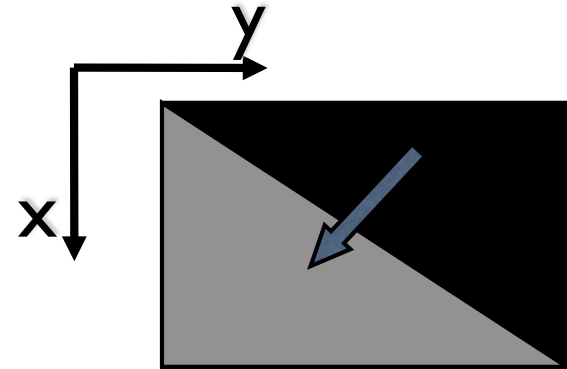


$$\frac{dl(x)}{dx}$$

$$\frac{dl(x)}{dx} > Th$$

2D

$l(x,y)$



$$\nabla l(x,y) = [\partial l(x,y)/\partial x \quad \partial l(x,y)/\partial y]^t = [l_x(x,y) \quad l_y(x,y)]^t$$

$$|\nabla l(x,y)| = (l_x^2(x,y) + l_y^2(x,y))^{1/2} > Th$$

$$\tan \theta = l_x(x,y) / l_y(x,y)$$

Edge Detection (2D)

Vertical Edges:

Convolve with:

| | | |
|----|---|---|
| -1 | 0 | 1 |
|----|---|---|

Horizontal Edges:

Convolve with:

| |
|----|
| -1 |
| 0 |
| 1 |

Noise Smoothing & Edge Detection

Convolve with:

| | | |
|----|---|---|
| -1 | 0 | 1 |
| -1 | 0 | 1 |
| -1 | 0 | 1 |

→
Vertical Edge Detection

↓
Noise Smoothing

This mask is called the (vertical) Prewitt Edge Detector

Noise Smoothing & Edge Detection

Convolve with:

| | | |
|----|----|----|
| -1 | -1 | -1 |
| 0 | 0 | 0 |
| 1 | 1 | 1 |

→
Noise Smoothing

↓
Horizontal Edge Detection

This mask is called the (horizontal) Prewitt Edge Detector

Sobel Edge Detector

Convolve with:

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

and

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

Gives more weight
to the 4-neighbors

Example

| | | | | |
|---|---|----|----|----|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 50 | 50 | 50 |
| 0 | 0 | 50 | 50 | 50 |
| 0 | 0 | 50 | 50 | 50 |

$I_x =$

| | | | | |
|---|----|-----|-----|-----|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 50 | 100 | 150 | 150 |
| 0 | 50 | 100 | 150 | 150 |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 |

$I_y =$

| | | | | |
|---|-----|-----|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 50 | 50 | 0 | 0 |
| 0 | 100 | 100 | 0 | 0 |
| 0 | 150 | 150 | 0 | 0 |
| 0 | 150 | 150 | 0 | 0 |

| | | |
|----|---|---|
| -1 | 0 | 1 |
| -1 | 0 | 1 |
| -1 | 0 | 1 |

| | | |
|----|----|----|
| -1 | -1 | -1 |
| 0 | 0 | 0 |
| 1 | 1 | 1 |

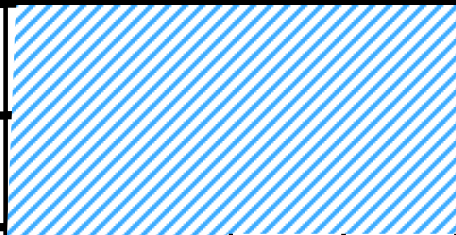

Example

$\nabla I =$

| | | | | |
|---|----|-----|-----|-----|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 50 | 100 | 150 | 150 |
| 0 | 50 | 100 | 150 | 150 |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 |

| | | | | |
|---|-----|-----|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 50 | 50 | 0 | 0 |
| 0 | 100 | 100 | 0 | 0 |
| 0 | 150 | 150 | 0 | 0 |
| 0 | 150 | 150 | 0 | 0 |

$|\nabla I| =$

| | | | | |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 |  | | | |
| 0 | | | | |
| 0 | | | | |
| 0 | | | | |
| 0 |  | | 0 | 0 |
| 0 | | | 0 | 0 |

$\tan(\theta) =$

| | | | | |
|---|----------|----------|---|---|
| X | X | X | X | X |
| X | 1 | 1/2 | 0 | 0 |
| X | 2 | 1 | 0 | 0 |
| X | ∞ | ∞ | X | X |
| X | ∞ | ∞ | X | X |

Formal Design of an Optimal Edge Detector

Edge detection involves 3 steps:

- Noise smoothing

- Edge enhancement

- Edge localization

J. Canny formalized these steps to design an **optimal** edge detector

Canny Edge Detector

Experiments consistently show that it performs very well
Probably, the most used by C.V. practitioners

Canny Edge Detector

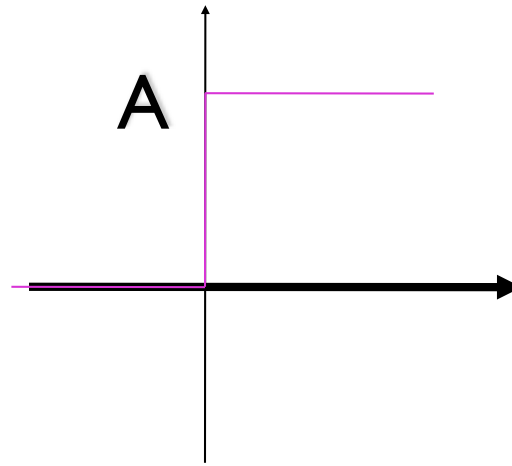
Uses a mathematical model of the edge and the noise

Formalizes a performance criteria

Synthesizes the best filter

Edge Model (1D)

An ideal edge can be modeled as a step



$$G(x) = \begin{cases} 0 & \text{if } x < 0 \\ A & \text{if } x \geq 0 \end{cases}$$

Noise Model (1D)

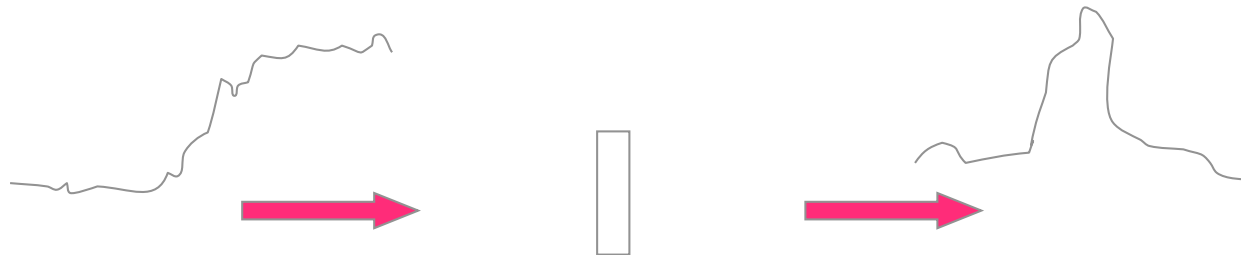
Additive, White Gaussian Noise

RMS noise amplitude/unit length n_o^2

Performance Criteria (1)

Good detection

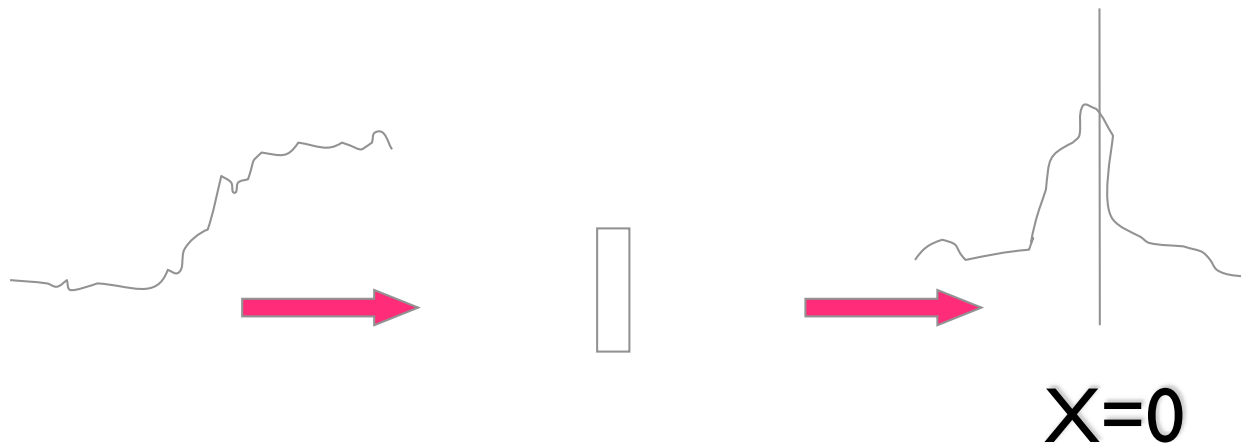
The filter must have a stronger response at the edge location ($x=0$) than to noise



Performance Criteria (2)

Good Localization

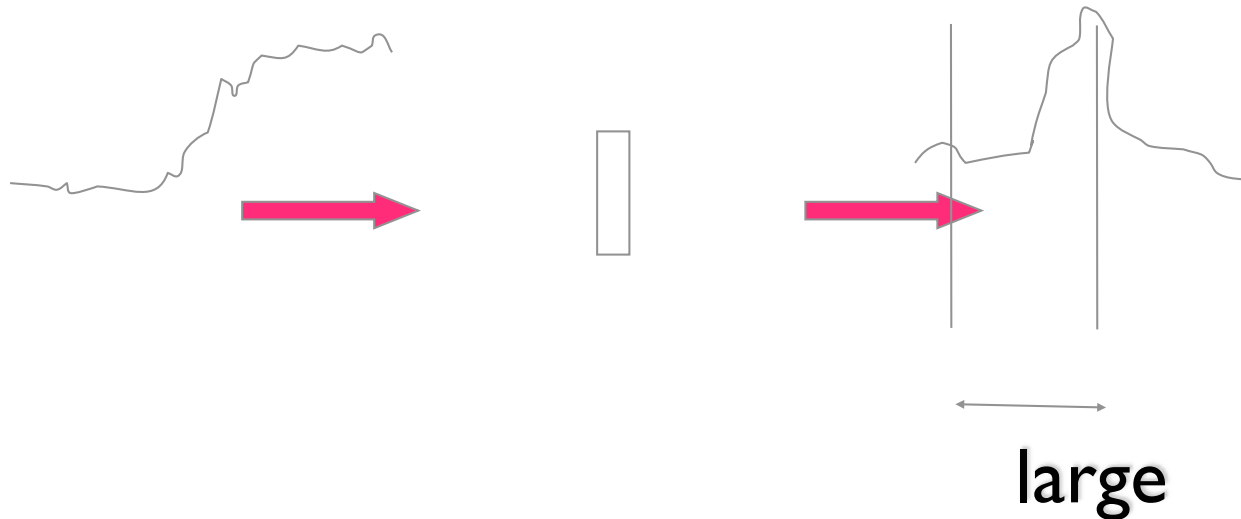
The filter response must be maximum very close to $x=0$



Performance Criteria (3)

Low False Positives

There should be only one maximum in a reasonable neighborhood of $x=0$

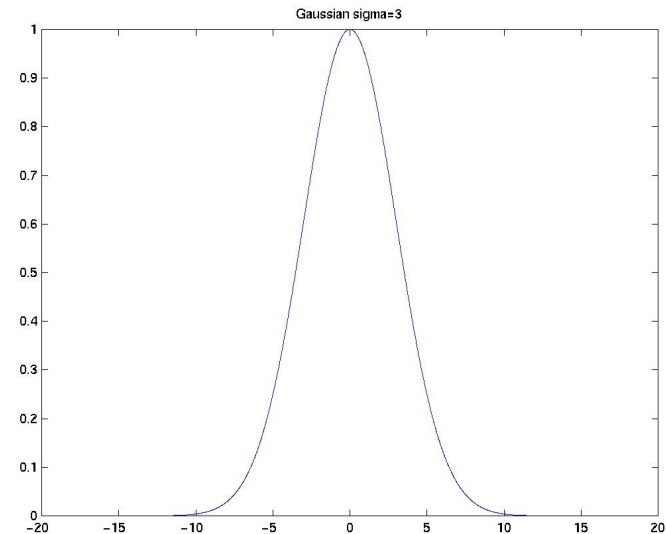


Canny Edge Detector

Canny found a linear, continuous filter that maximized the three given criteria.
There is no close-form solution for the optimal filter.
However, it looks VERY SIMILAR to the derivative of a Gaussian.

1D Gaussian

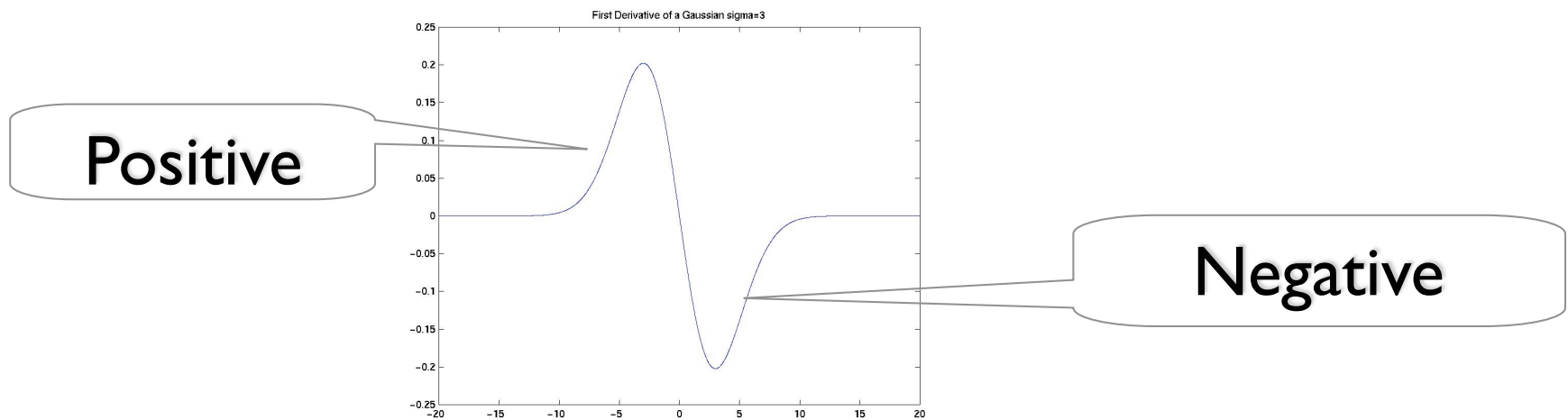
$$g(x) = e^{-\frac{x^2}{2\sigma^2}}$$



$$g'(x) = -\frac{1}{2\sigma^2} 2xe^{-\frac{x^2}{2\sigma^2}} = -\frac{x}{\sigma^2} e^{-\frac{x^2}{2\sigma^2}}$$

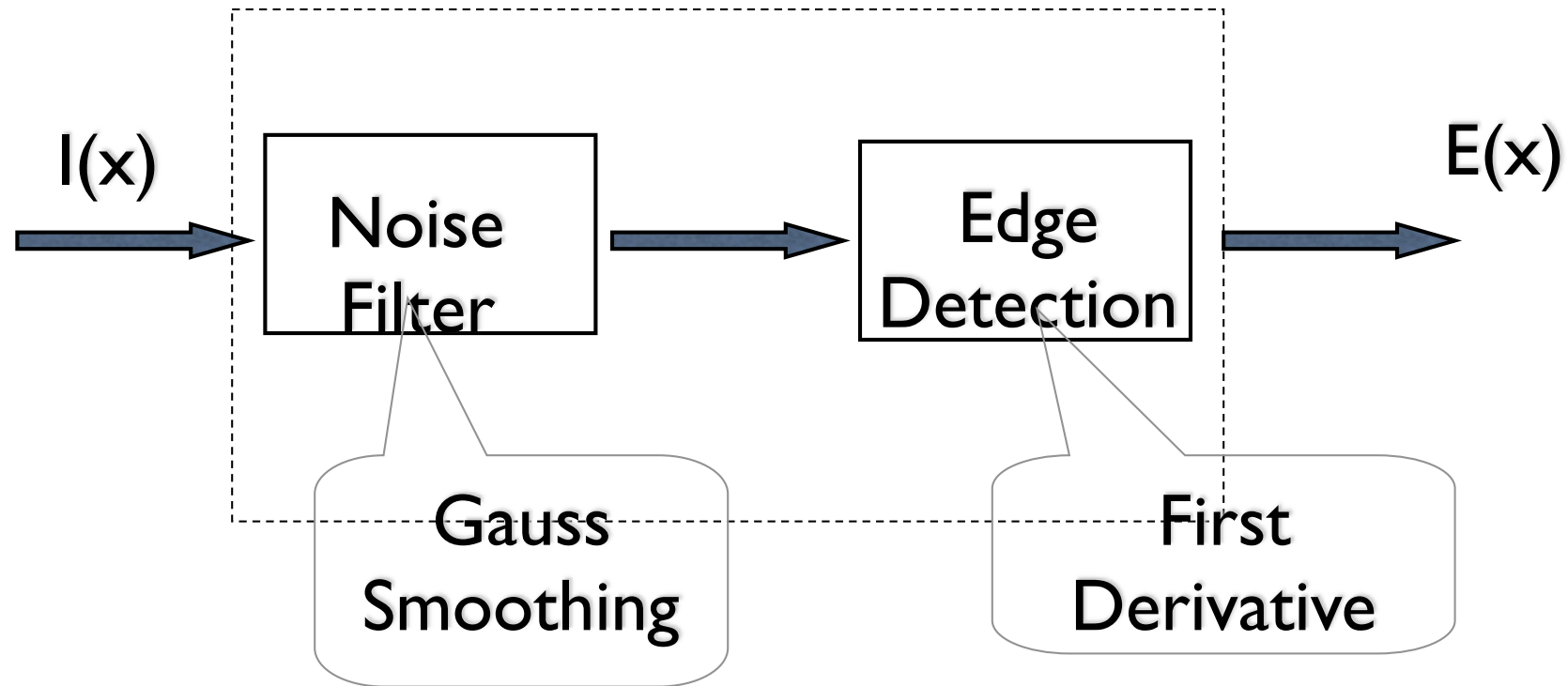
First Derivative of a Gaussian

$$g'(x) = -\frac{1}{2\sigma^2} 2xe^{-\frac{x^2}{2\sigma^2}} = -\frac{x}{\sigma^2} e^{-\frac{x^2}{2\sigma^2}}$$



As a mask, it is also computing a difference (derivative)

Another interpretation:



$$E(x) = \frac{d(I(x) * G(x))}{dx} = I(x) * \frac{dG(x)}{dx}$$

Canny Edge Detector 2D

First derivative



Gradient vector

$$E(x) = \frac{d(I(x) * G(x))}{dx}$$



$$E(x, y) = \nabla(I(x, y) * G(x, y))$$

Absolute value



Magnitude

$$|E(x)| \geq Th$$



$$\|E(x, y)\| \geq Th$$

Algorithm CANNY_ENHANCER

The input is image I ; G is a zero mean Gaussian filter (std = σ)

1. $J = I * G$ (smoothing)
2. For each pixel (i,j) : (edge enhancement)

Compute the image gradient

$$\nabla J(i,j) = (J_x(i,j), J_y(i,j))'$$

1. Estimate edge strength

1. $e_s(i,j) = (J_x^2(i,j) + J_y^2(i,j))^{1/2}$

Estimate edge orientation

$$e_o(i,j) = \arctan(J_x(i,j)/J_y(i,j))$$

The output are images E_s and E_o

Efficiency Considerations

1D

$$E(x) = \frac{d(I(x) * G(x))}{dx}$$

$$= I(x) * \frac{dG(x)}{dx}$$

•

2D

$$E(x, y) = \nabla(I(x, y) * G(x, y))$$

$$= I(x, y) * \nabla G(x, y)$$

Efficiency Considerations

We can also use Gaussian separability:

$$\begin{aligned} E(x, y) &= \nabla(I(x, y) * G(x, y)) \\ &= I(x, y) * \nabla G(x, y) \\ &= \nabla G(y) * (\nabla G(x) * I(x, y)) \end{aligned}$$

CANNY_ENHANCER

The output image E_s has the magnitudes of the smoothed gradient.

σ determines the amount of smoothing.

E_s has large values at edges

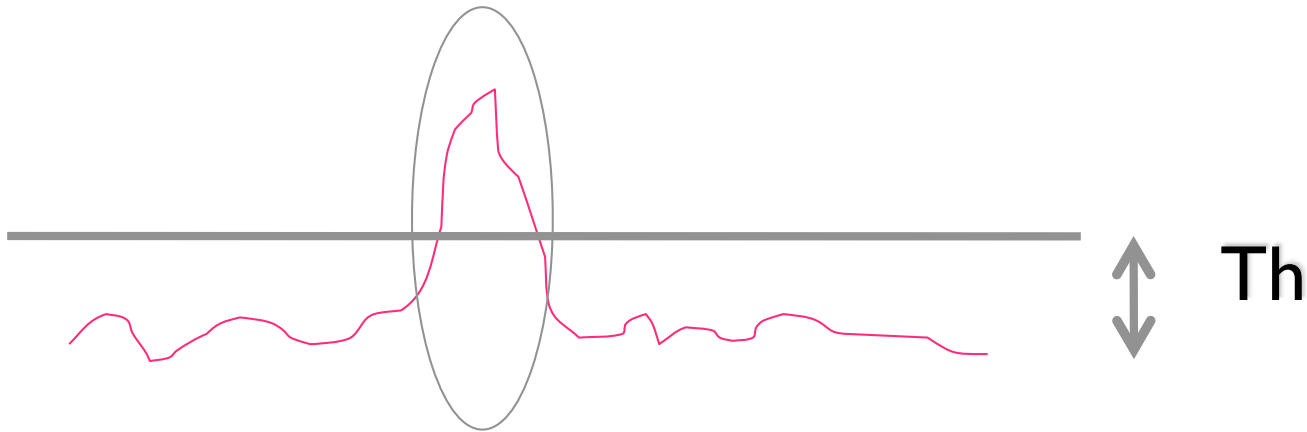


Edge ENHANCER

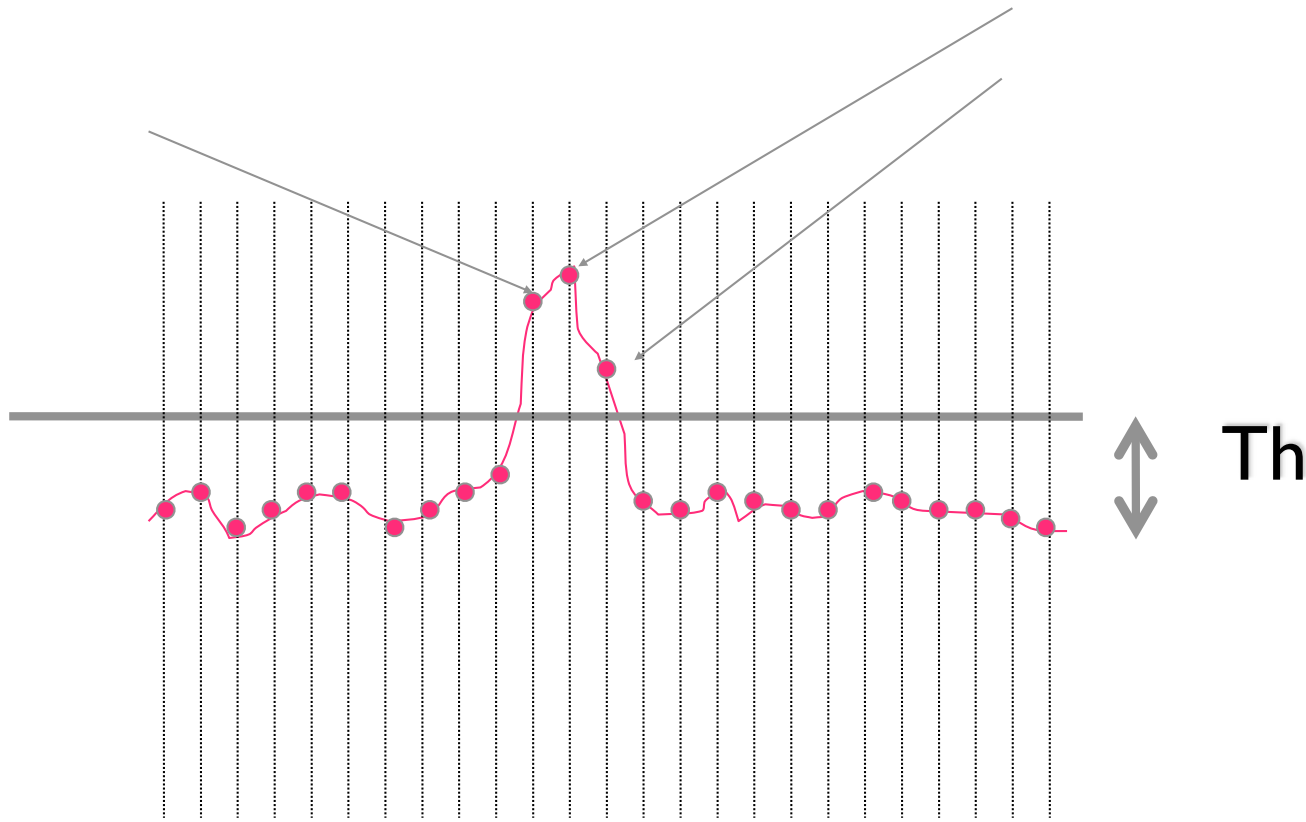
How do we “detect” edges?

E_s has large values at edges:

Find local maxima



... but it also may have wide ridges around the local maxima (large values around the edges)



NONMAX_SUPPRESSION

The inputs are E_s & E_o (outputs of CANNY_ENHANCER)

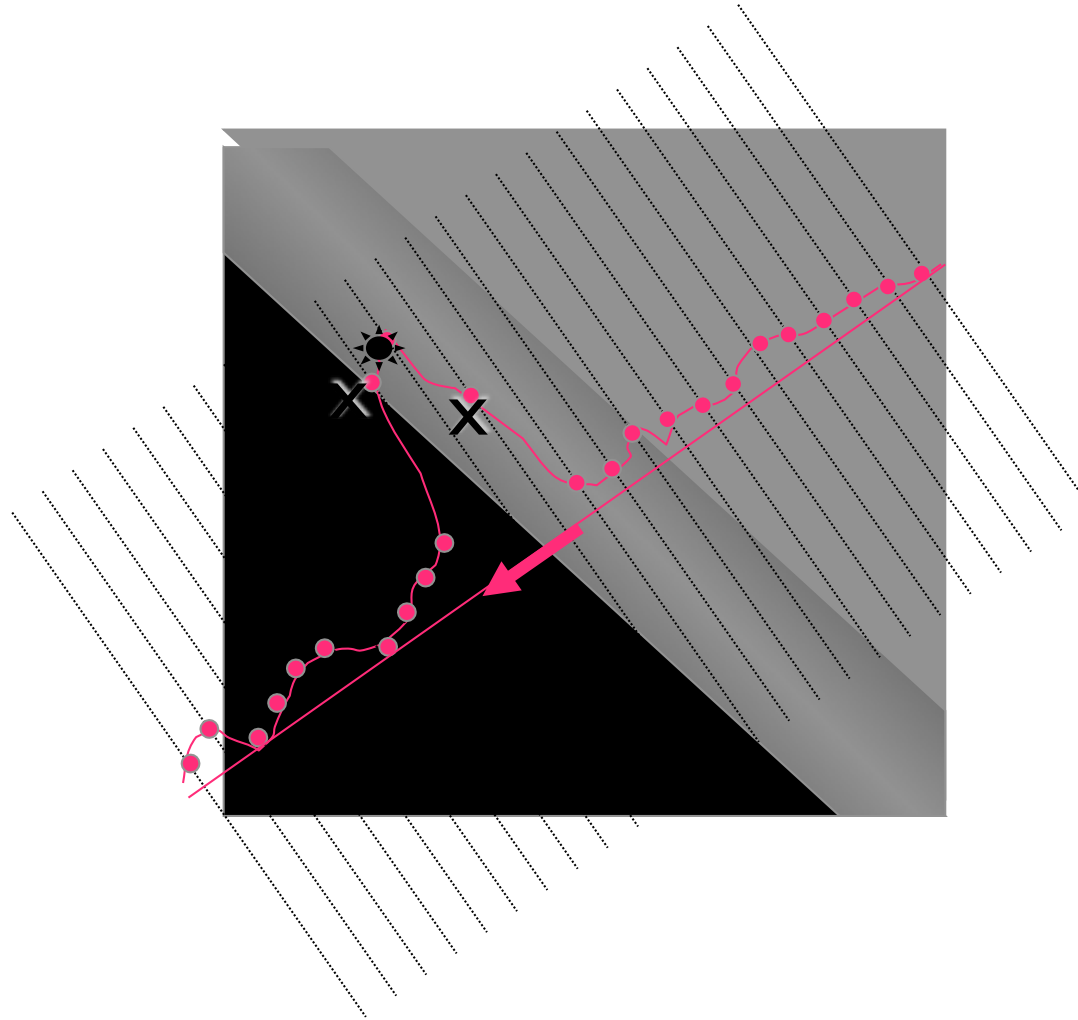
Consider 4 directions $D=\{0,45,90,135\}$ wrt x

For each pixel (i,j) do:

1. Find the direction $d \in D$ s.t. $d \cong E_o(i,j)$ (normal to the edge)
If $\{E_s(i,j)$ is smaller than at least one of its neigh. along $d\}$
 1. $I_N(i,j)=0$Otherwise, $I_N(i,j)=E_s(i,j)$

The output is the thinned edge image I_N

Graphical Interpretation



Thresholding

Edges are found by thresholding the output of NONMAX_SUPPRESSION

If the threshold is too high:

- Very few (none) edges

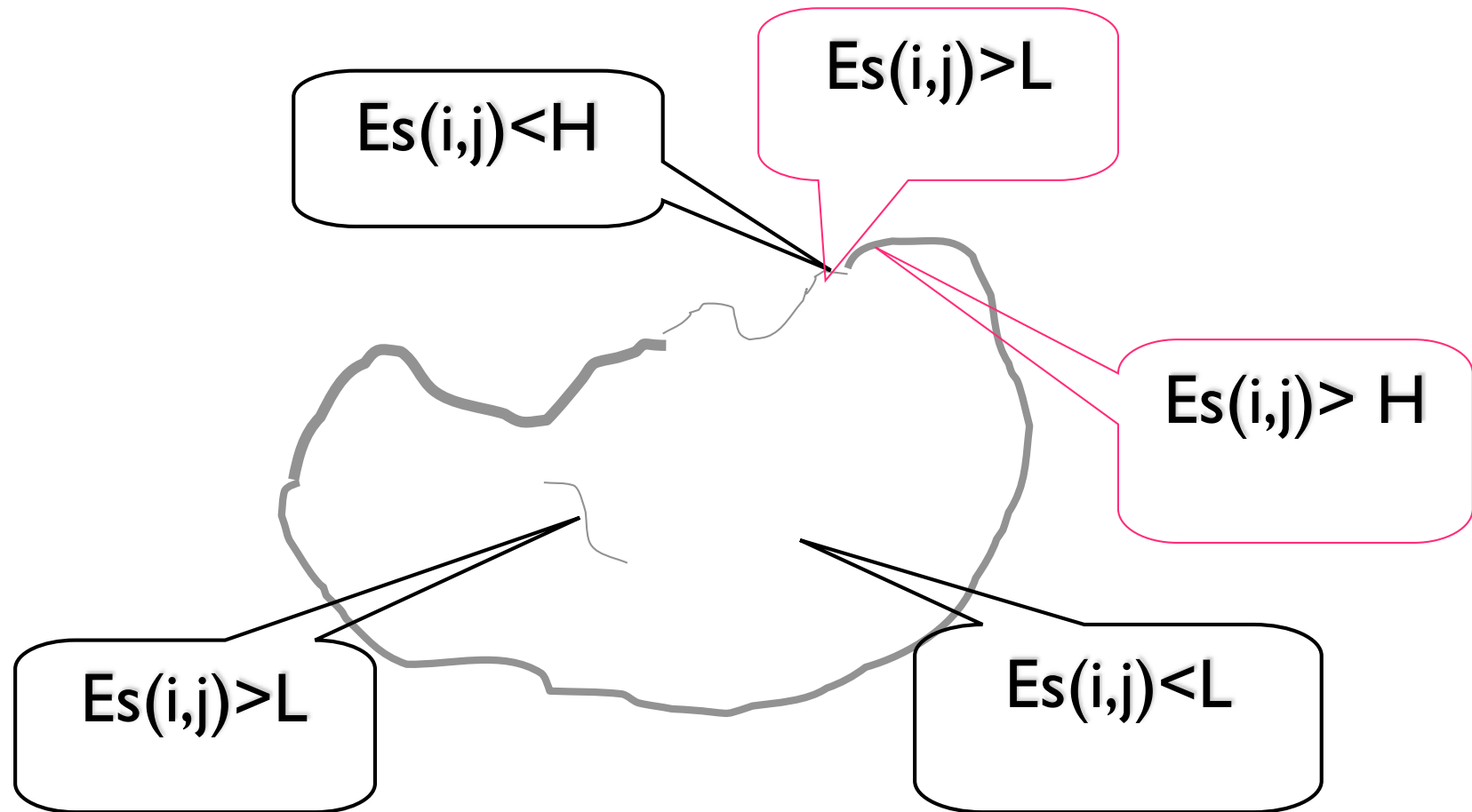
- High MISDETECTIONS, many gaps

If the threshold is too low:

- Too many (all pixels) edges

- High FALSE POSITIVES, many extra edges

SOLUTION: Hysteresis Thresholding



Strong edges reinforce adjacent weak edges

HYSTERESIS_THRESH

Inputs:

I_N (output of NONMAX_SUPPRESSION),

E_o (output of CANNY_ENHANCER),

thresholds L and H .

For all pixels in I_N and scanning in a fixed order:

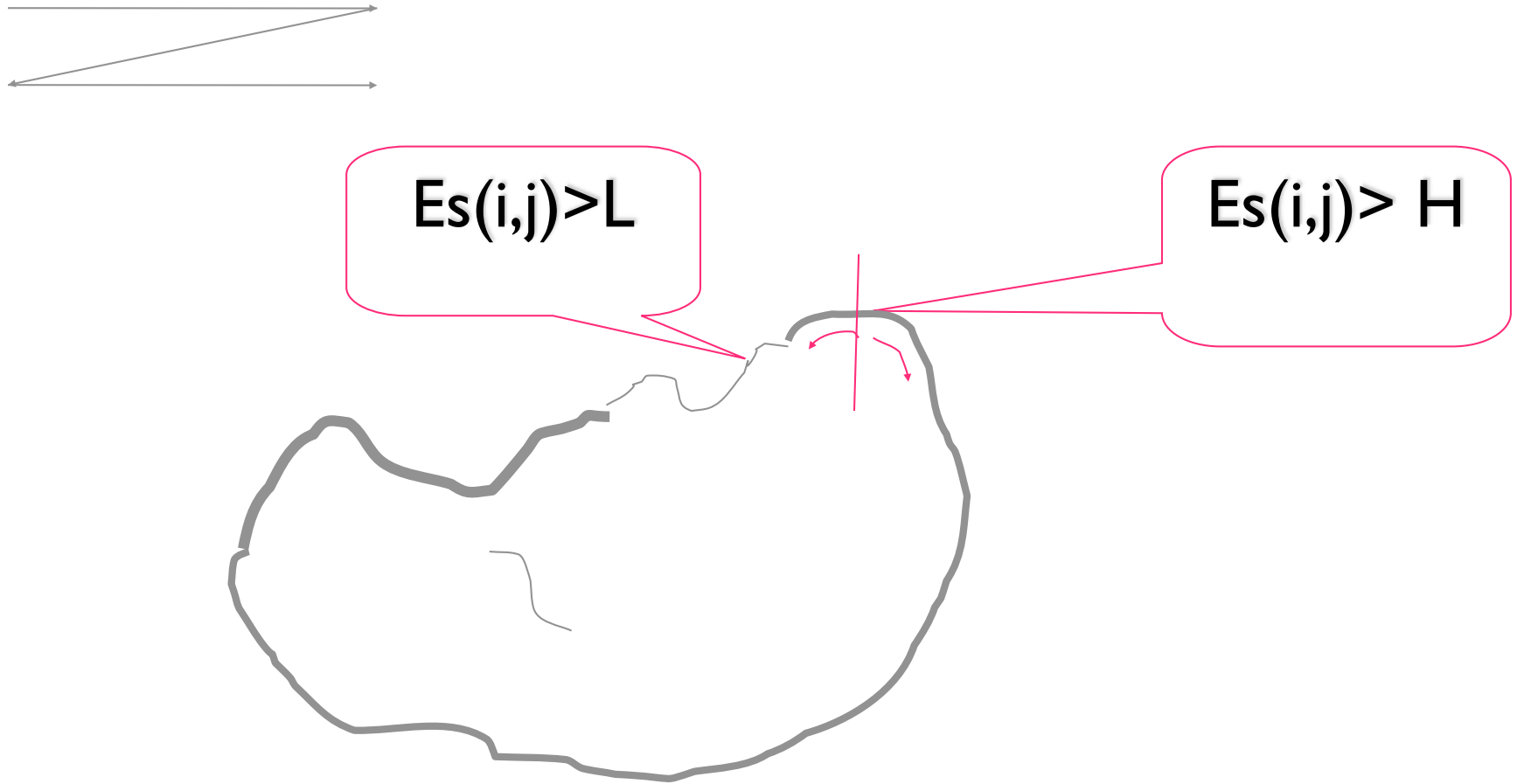
Locate the next unvisited pixel s.t. $I_N(i,j) > H$

Starting from $I_N(i,j)$, follow the chains of connected local maxima, in both directions perpendicular to the edge normal, as long as $I_N > L$.

Mark all visited points, and save the location of the contour points.

Output: a set of lists describing the contours.

Hysteresis Thresholding

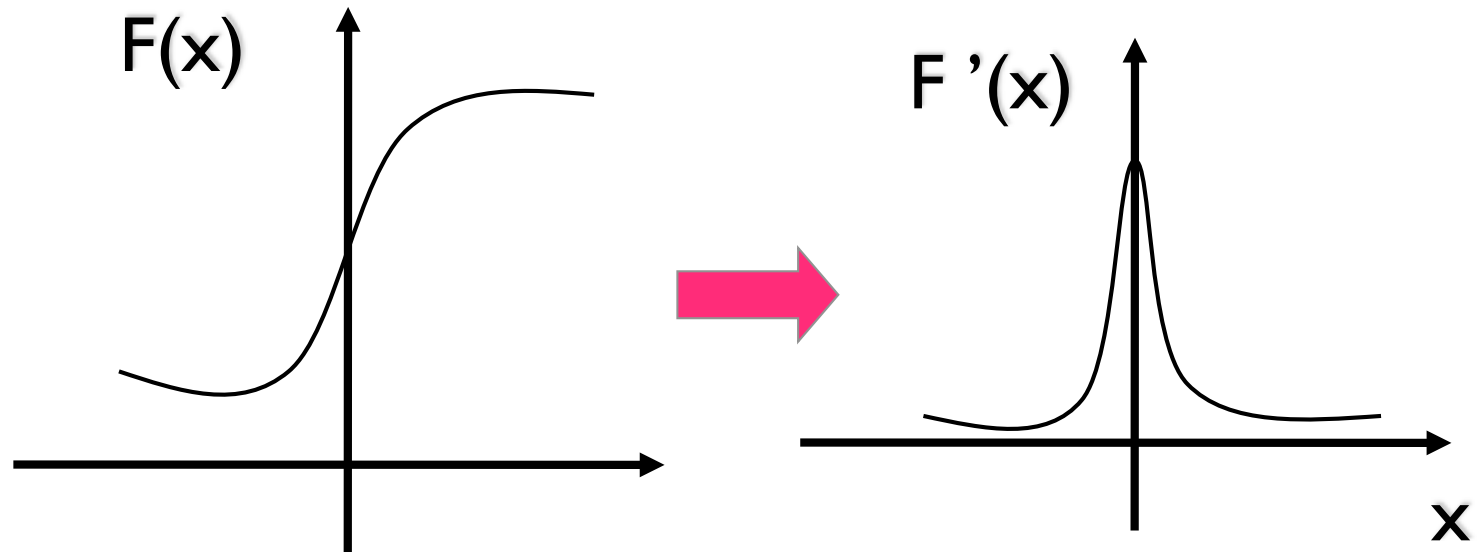


Other Edge Detectors

(2nd order derivative filters)

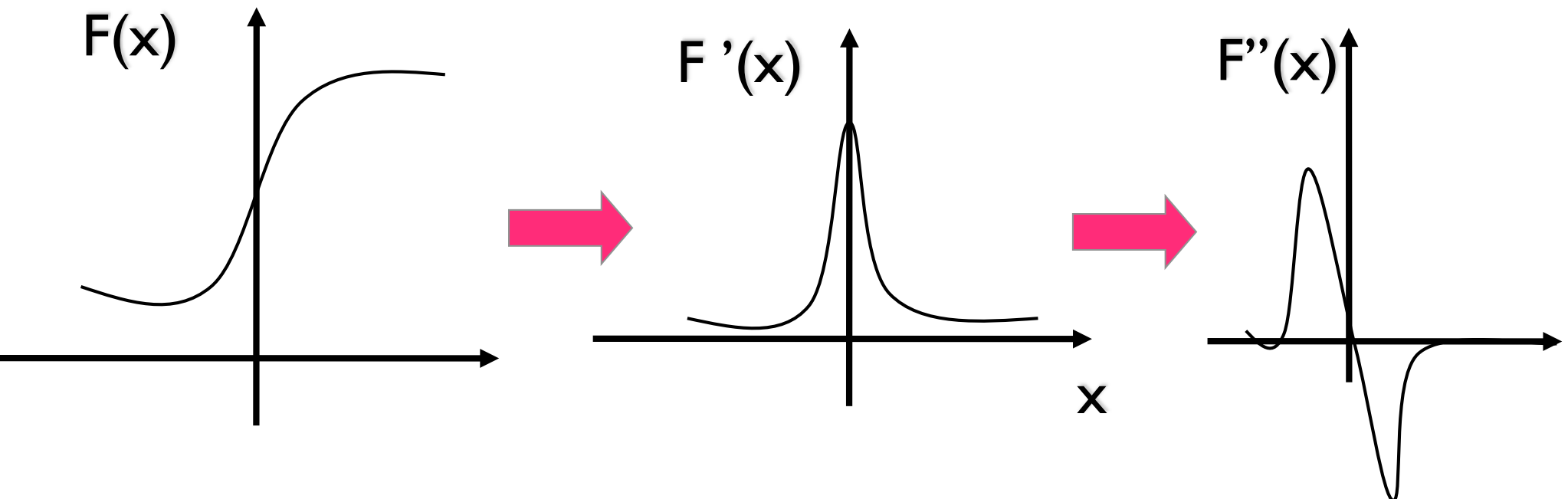
First-order derivative filters (1D)

Sharp changes in gray level of the input image correspond to “peaks” of the first-derivative of the input signal.



Second-order derivative filters (1D)

Peaks of the first-derivative of the input signal, correspond to “zero-crossings” of the second-derivative of the input signal.



NOTE:

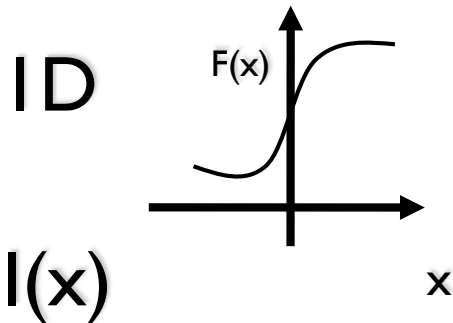
$F''(x)=0$ is not enough!

$F'(x) = c$ has $F''(x) = 0$, but there is no edge

The second-derivative **must change sign**, -- i.e. from (+) to (-) or from (-) to (+)

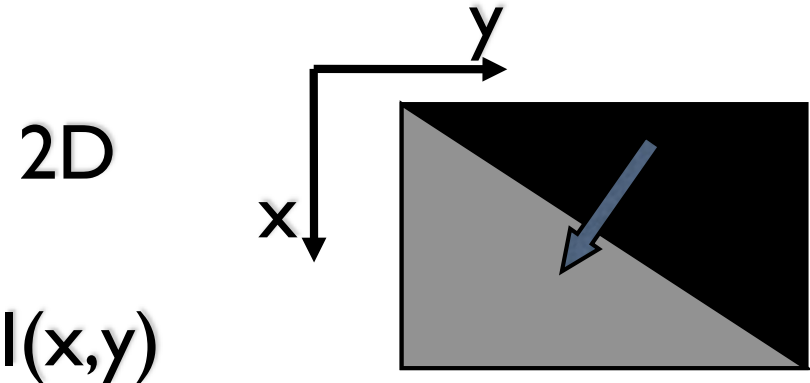
The sign transition depends on the intensity change of the image – i.e. from dark to bright or vice versa.

Edge Detection (2D)



$$\left| \frac{dI(x)}{dx} \right| > Th$$

$$\frac{d^2I(x)}{dx^2} = 0$$



$$|\nabla I(x,y)| = (I_x^2(x,y) + I_y^2(x,y))^{1/2} > Th$$

$$\tan \theta = I_x(x,y) / I_y(x,y)$$

$$\nabla^2 I(x,y) = I_{xx}(x,y) + I_{yy}(x,y) = 0$$

Laplacian

Notes about the Laplacian:

$\nabla^2 I(x,y)$ is a SCALAR

↑ Can be found using a SINGLE mask

↓ Orientation information is lost

$\nabla^2 I(x,y)$ is the sum of SECOND-order derivatives

But taking derivatives increases noise

Very noise sensitive!

It is always combined with a smoothing operation

LOG Filter

First smooth (Gaussian filter),

Then, find zero-crossings (Laplacian filter):

$$O(x,y) = \nabla^2(I(x,y) * G(x,y))$$

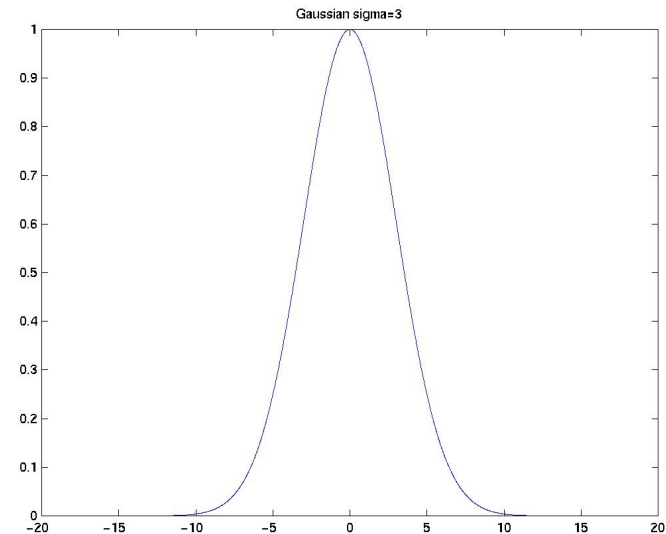
Using linearity:

$$O(x,y) = \nabla^2 G(x,y) * I(x,y)$$

This filter is called: “Laplacian of the Gaussian” (LOG)

1D Gaussian

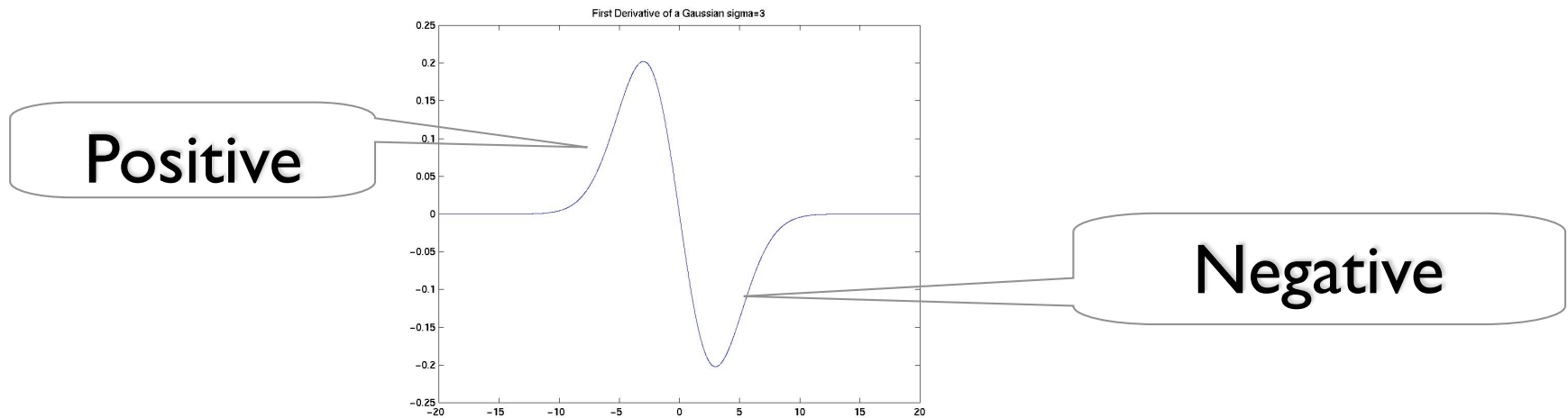
$$g(x) = e^{-\frac{x^2}{2\sigma^2}}$$



$$g'(x) = -\frac{1}{2\sigma^2} 2xe^{-\frac{x^2}{2\sigma^2}} = -\frac{x}{\sigma^2} e^{-\frac{x^2}{2\sigma^2}}$$

First Derivative of a Gaussian

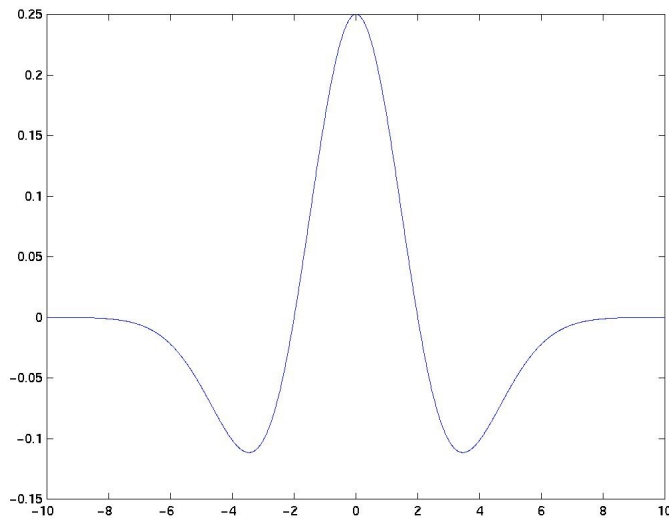
$$g'(x) = -\frac{1}{2\sigma^2} 2xe^{-\frac{x^2}{2\sigma^2}} = -\frac{x}{\sigma^2} e^{-\frac{x^2}{2\sigma^2}}$$



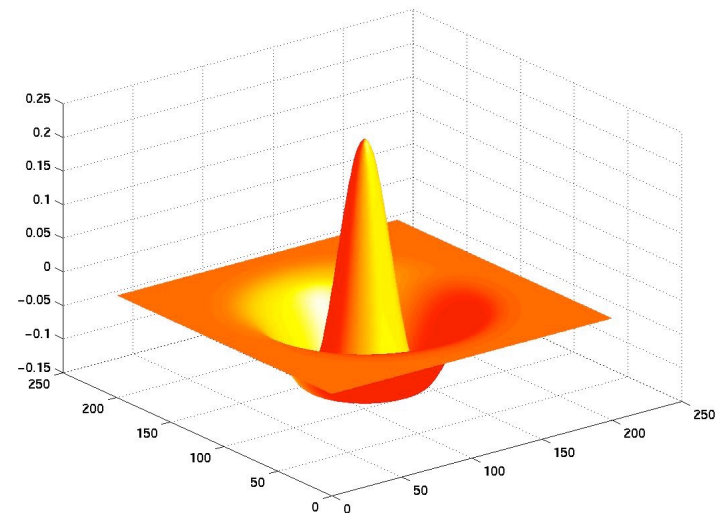
As a mask, it is also computing a difference (derivative)

Second Derivative of a Gaussian

$$g''(x) = \left(\frac{x^2}{\sigma^3} - \frac{1}{\sigma} \right) e^{-\frac{x^2}{2\sigma^2}}$$



2D



“Mexican Hat”