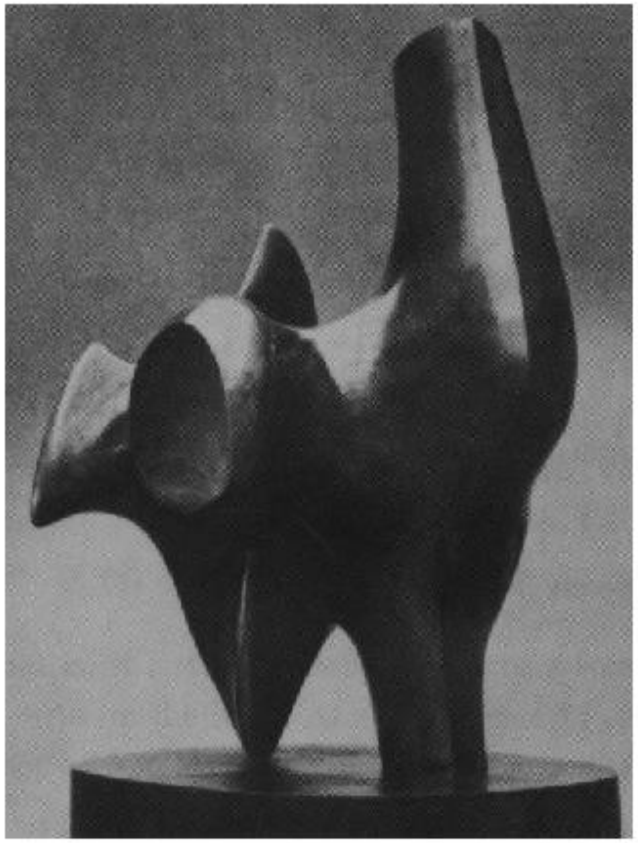
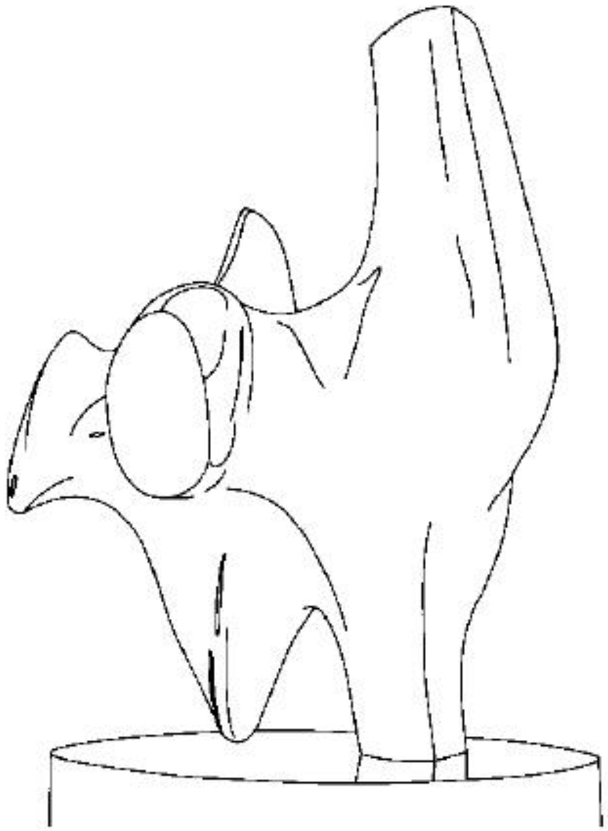
Edge Detection

- Edges characterize boundaries and are therefore an important fundamental problem in image processing.
- Edges in images are areas with strong intensity contrasts – a jump in intensity from one pixel to the next.

- Detecting Edges in an image **significantly reduces the amount of data and filters out useless information, while preserving the important structural properties in an image.**

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

- Edge detection is one of the most commonly used operations in image analysis
- There are probably more algorithms in the literature for enhancement and detecting edges than for any other single subject.
  - The reason for this is that edges form the outline of an object and the background

- If the edges in an image can be identified accurately, all of the objects can be located and basic properties such as area, perimeter, and shape can be measured
- Since computer vision involves the identification and classification of objects in an image, edge detection is an essential tool

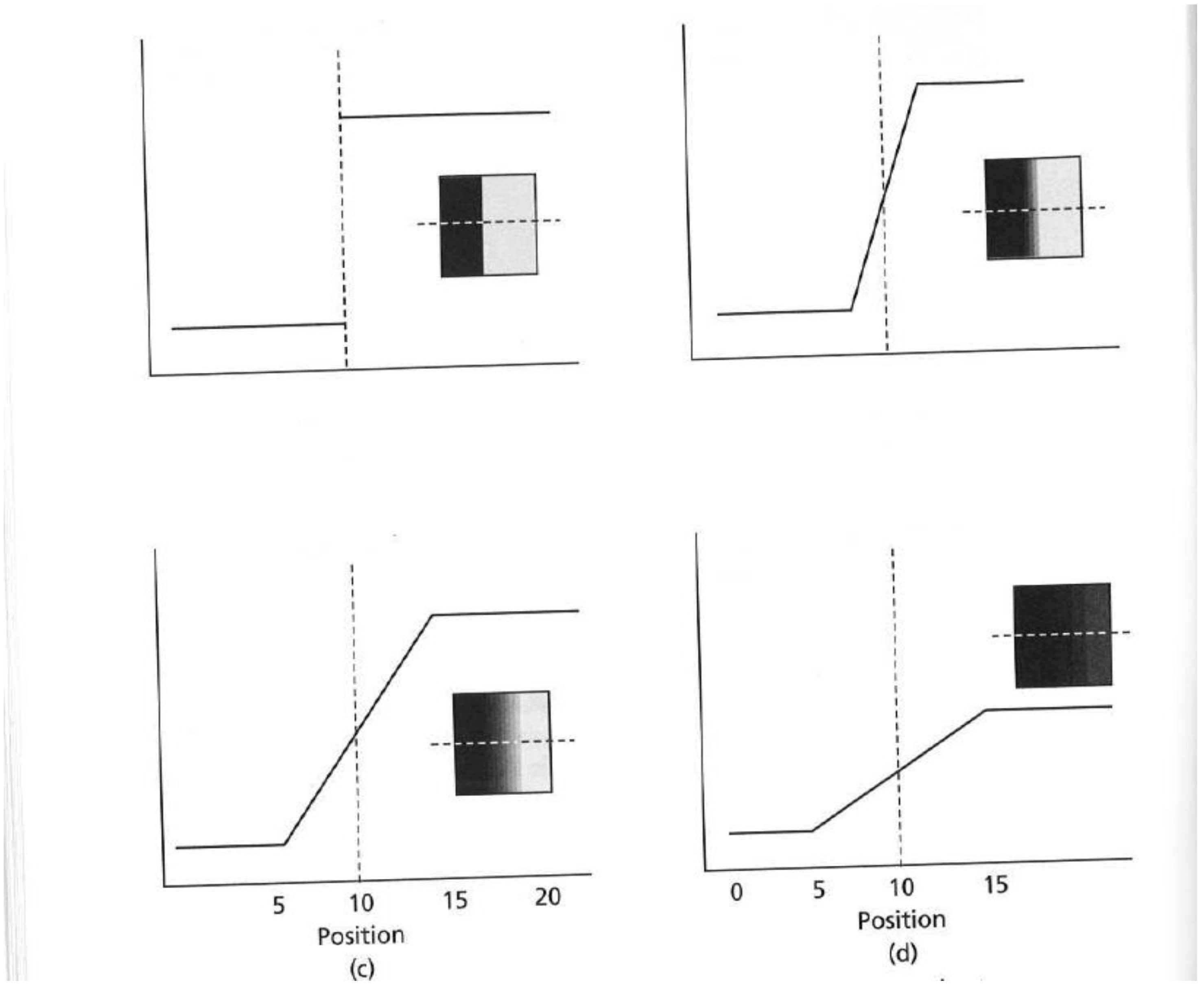
Importance of edge detection in computer vision

- **Information reduction**
  - Replace image by a cartoon in which objects and surface markings are outlined
  - These are the most informative parts of the image
- **Biological application**
  - Initial stages of mammalian vision systems involve detection of edges and local features

- Edge detection is part of a process called ***segmentation*** – the identification of regions within an image.
- ***Edge detection*** is the process of locating the edge pixels
- ***Edge enhancement*** increases the contrast between the edges and the background so that the edges become more visible
- ***Edge tracing*** is the process of following the edges, usually collecting the edge pixels into a list.

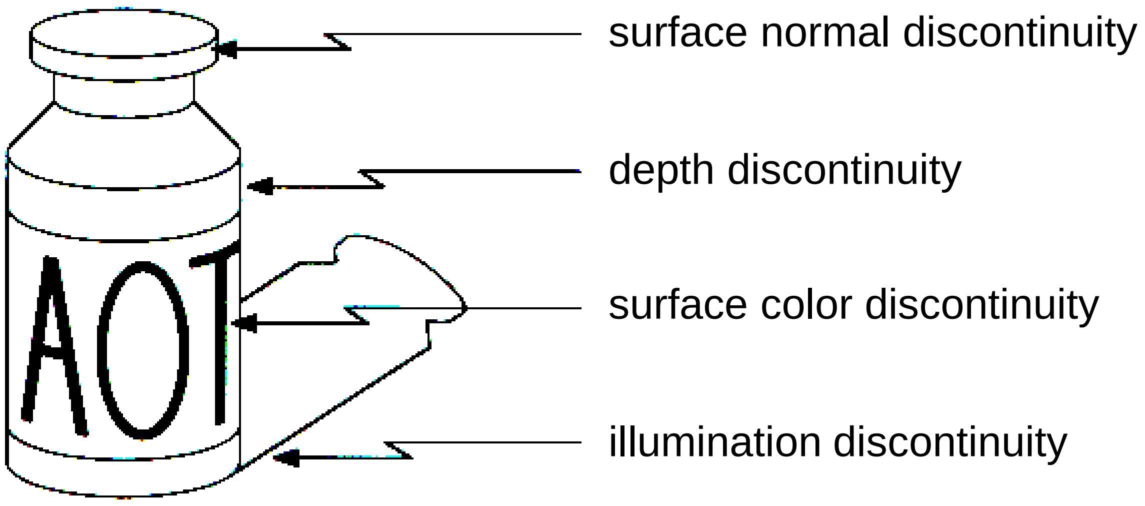
- Most good algorithms begin with a clear statement of the problem to be solved, and a cogent analysis of the possible method of solution and the conditions under which the methods will operate correctly.
  - Using this paradigm to define an edge-detection algorithm means first defining what an edge is, then using this definition to suggest methods of enhancement

- There are a number of possible definitions of an edge, each being applicable in various specific circumstances
  - One of the most common and most general definitions is ***the ideal step edge***.


5
10
15
20
Position
(c)
0
5
10
15
Position
(d)

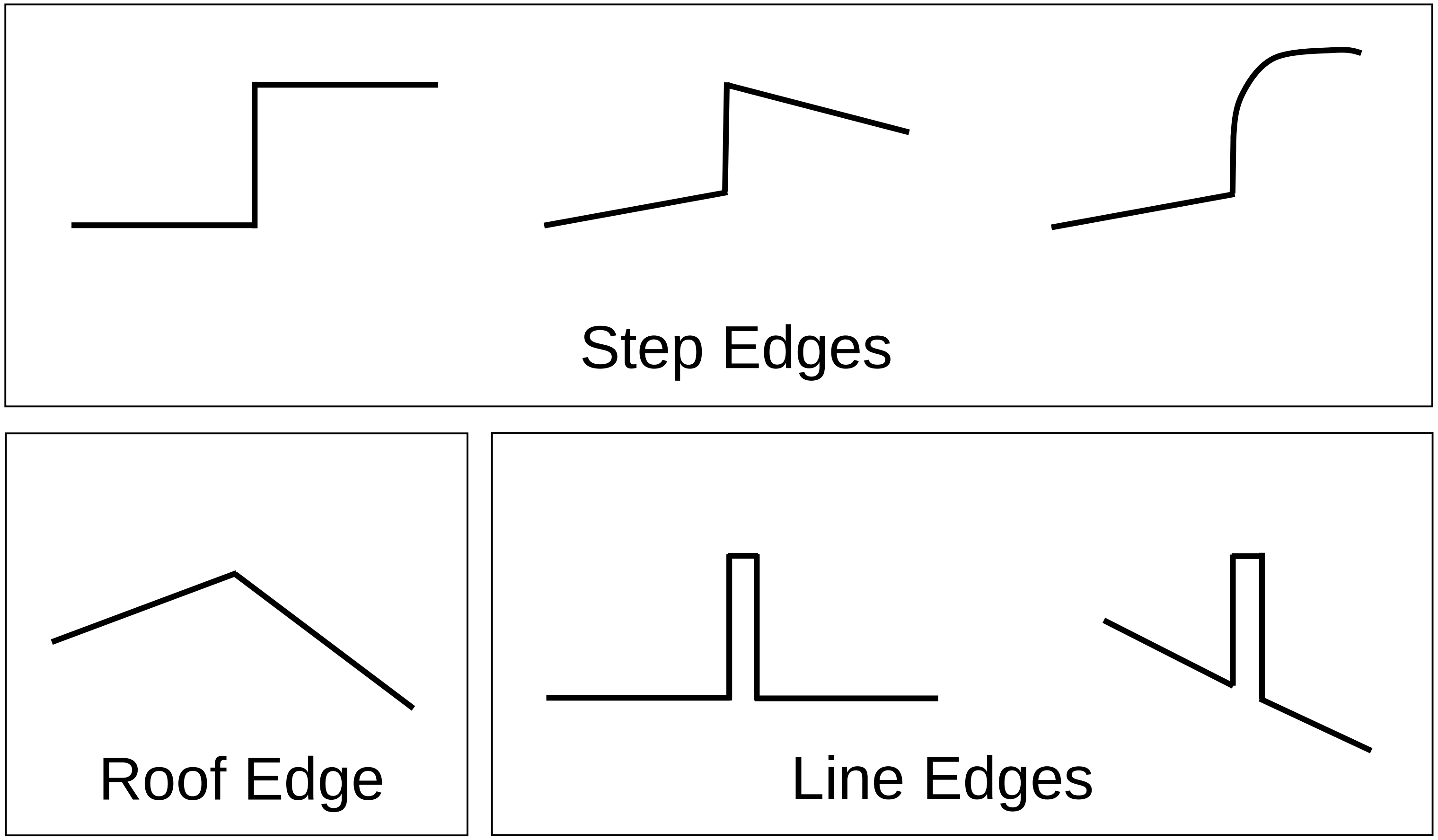
- The step edge is ideal because it is easy to detect: In the absence of noise, any significant change in grey level would indicate an edge.
- A step edge never really occurs in an image because:
  - Objects rarely have such a sharp outline
  - A scene is never sampled so that edge occur exactly at the margins of a pixel
  - Due to noise

Origin of Edges

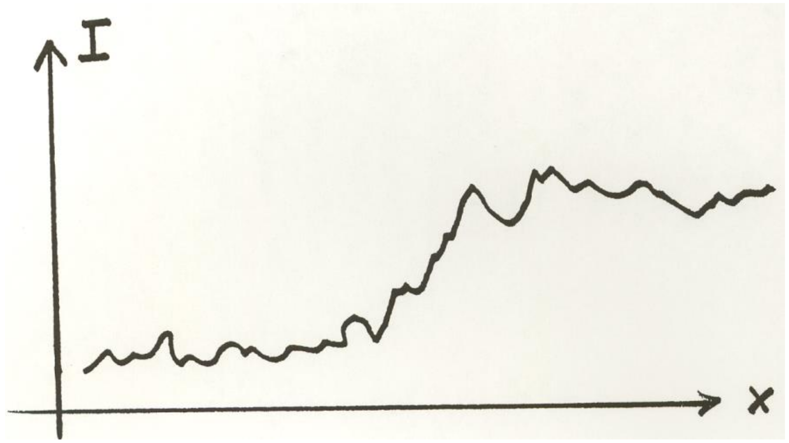

surface normal discontinuity
depth discontinuity
surface color discontinuity
illumination discontinuity
AOT

- Edges are caused by a variety of factors

Edge Types



Real Edges



Noisy and Discrete!

We want an **Edge Operator** that produces:

- Edge **Magnitude**
- Edge **Orientation**
- High **Detection** Rate and Good **Localization**

Noise

- All image-acquisition process are subject to noise of some type.
- The ideal situation (no noise) never occurs in practice.
- Noise cannot be predicted accurately because of its random nature, and cannot even be measured accurately from a noisy image, since the contribution to the grey levels of the noise can not be distinguished from the pixel data.

- However, noise can sometimes be characterized by its effect on the image and is usually expressed as a probability distribution with a specific mean and standard deviation.

- There are two types of noise that are of specific interest in image analysis.

1. ***Signal-independent*** noise is a random set of grey levels, statistically independent of the image data.

- This kind of noise occurs when an image is transmitted electronically from one place to another
- If ***A*** is a perfect image and ***N*** is the noise that occurs during transmission, then the final image ***B*** is:

  $\boldsymbol{B = A + N}$

- ***A*** and ***N*** are unrelated to each other

- The noise image N could have any statistical properties, but a common assumption is that it follows the normal distribution with a mean of zero and some measured or presumed standard deviation
- It is simple matter to create an artificially noisy image have known characteristics, and such images are very useful tools for experimenting with edge-detection algorithm

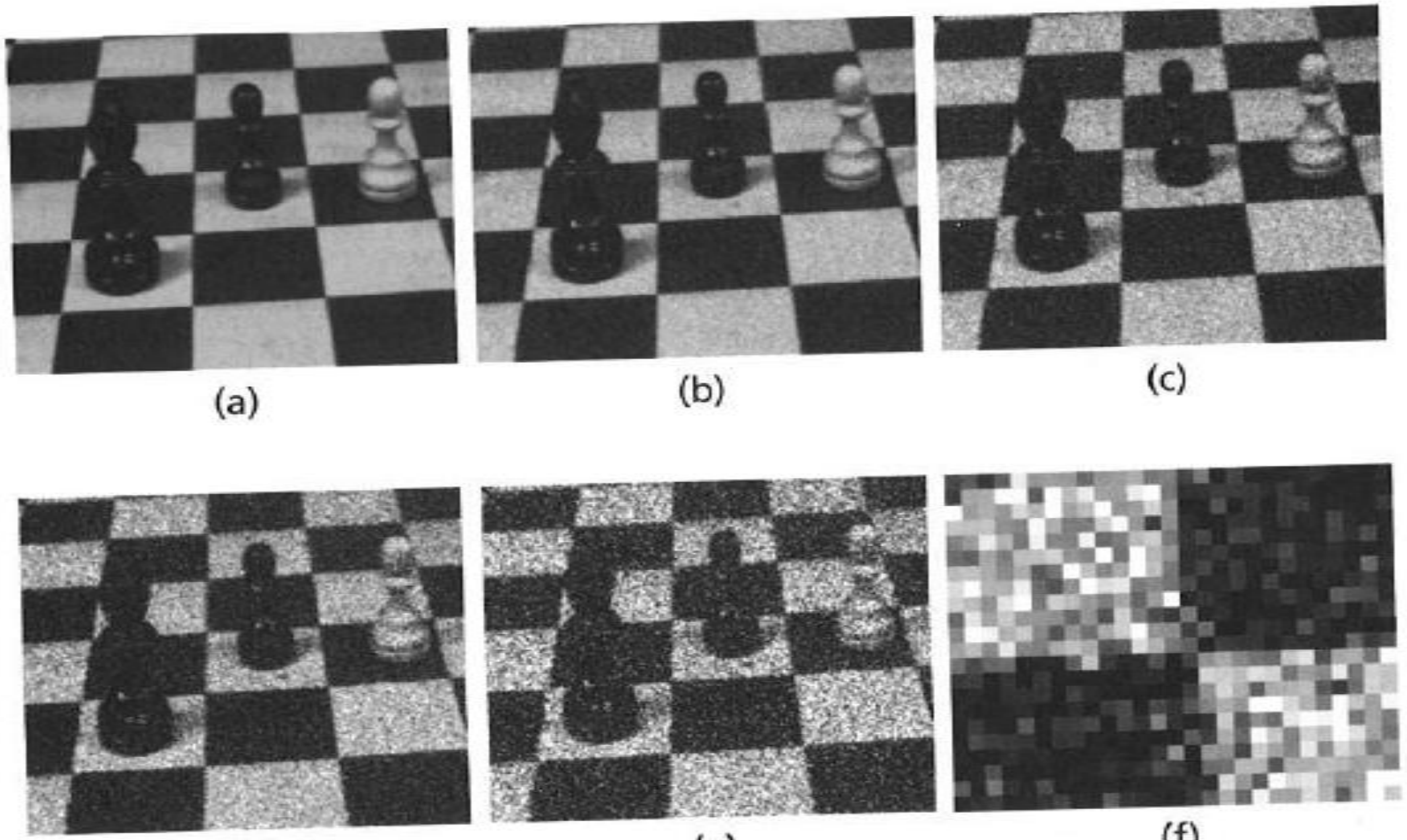


Figure 1.5 Normally distributed noise and its effect on an image. (a) Original image. (b) Noise having σ = 10. (c) Noise having σ = 20. (d) Noise having σ = 30. (e) Noise having σ = 50. (f) Expanded view of an intersection of four regions in the σ = 50 image.

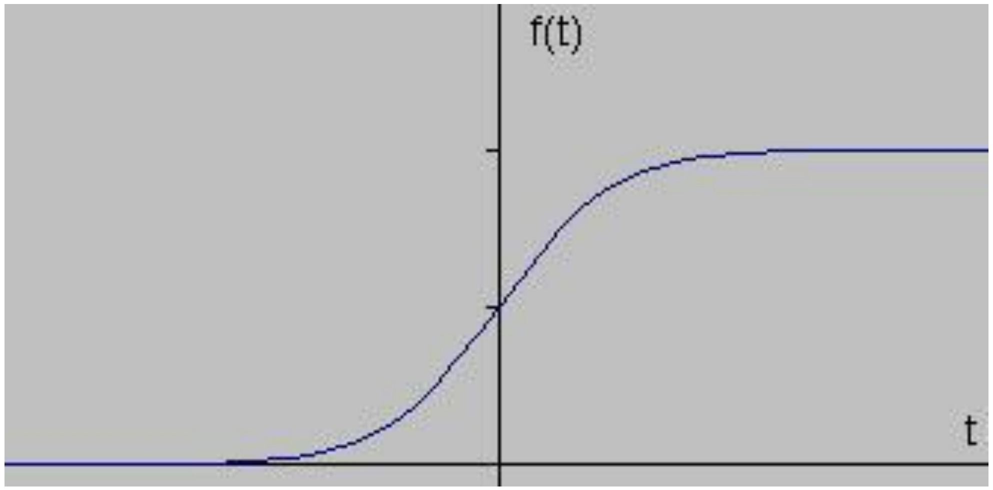
2. ***Signal-dependent noise:*** the level of the noise value at each point in the image is a function of the grey level there.

- This kind of noise is generally harder to deal with.

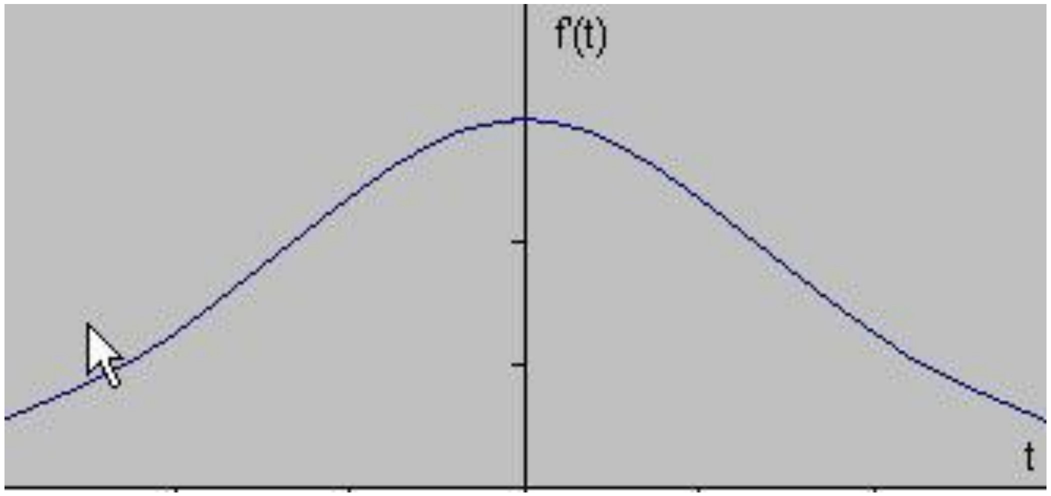
- There are many ways to perform edge detection. However, the majority of different methods may be grouped into two categories,
  - **Gradient** （梯度）
  - **Laplacian**

- The ***gradient method*** detects the edges by looking for the maximum and minimum in the **first derivative** of the image.
- The ***Laplacian method*** searches for zero crossings in the **second derivative** of the image to find edges.

- An edge has the one-dimensional shape of a ramp and calculating the derivative of the image can highlight its location.
- Suppose we have the following signal, with an edge shown by the jump in intensity below:



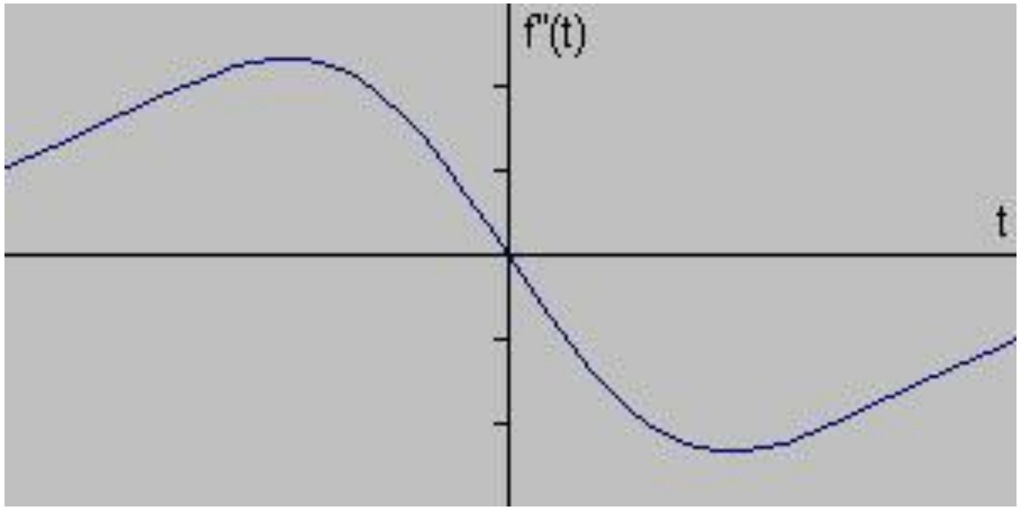
- If we take the gradient of this signal (which, in one dimension, is just the first derivative with respect to t) we get the following:



- The derivative shows a maximum (changing rate) located at the center of the edge in the original signal.

- This method of locating an edge is characteristic of the "gradient filter" family of edge detection filters
- A pixel location is declared an edge location if the value of the gradient exceeds some threshold.

- When the first derivative is at a maximum, the second derivative is zero.
- Another alternative to finding the location of an edge is to locate the zeros in the second derivative.



Gradient operators

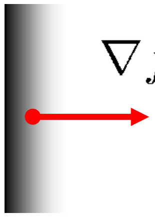
1. **Derivative operators**: identify places where there are large intensity changes
2. **Template-matching schemes**: the edge is modeled by a small image showing the abstracted properties of a perfect edge

Derivative operators

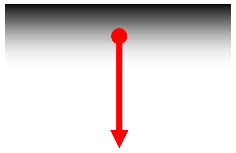
- Since an edge is defined by a change in grey level, an operator that is sensitive to this change will operate as an edge detector
- Derivative: is the rate of change of a function
  - The rate of change of the grey levels in an image is large near an edge and small in constant areas
  - To a two-dimensional image, the partial derivatives of the image are used

Gradient

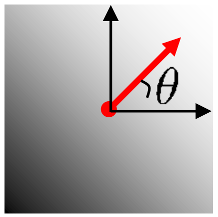
- Gradient equation: $$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$
- Represents direction of most rapid change in intensity



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



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



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

- Gradient direction: $$\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} \Big/ \frac{\partial f}{\partial x}\right)$$
- The *edge strength* is given by the gradient magnitude

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

- Because an image is discrete, we use *differences* instead.

$$\nabla_{x1} f(x,y) = f(x,y) - f(x-1,y)$$
$$\nabla_{y1} f(x,y) = f(x,y) - f(x,y-1)$$

$$\nabla_{x2} f(x,y) = f(x+1,y) - f(x-1,y)$$
$$\nabla_{y2} f(x,y) = f(x,y+1) - f(x,y-1)$$

- The edge magnitude will be a real number, and is usually converted to an integer by rounding.
- Any pixel having a gradient exceeds a specified threshold value is said to be an edge pixel, and others are not.
- Usually the **middle value** in the range of grey levels will be used as a threshold.

Derivative Operator Algorithm

1. Calculate the thresholding $T = \dfrac{f_{\max} - f_{\min}}{2}$
2. Calculate the ∇_x, ∇_y for each image pixels
3. Calculate $\| \nabla f \|$
4. If $\| \nabla f \| > T$, then *(x, y)* is edge;

   otherwise, *(x, y)* is not edge

Attention

- An edge detector can report an edge where none exists
  - This can be due to noise, or simply poor design or thresholding, and is called a ***false positive***
- An edge detector could fail to report an edge pixel that does exist
  - This is called a ***false negative***
- The position of the edge pixel could be wrong

Template-based Edge Detection

- The idea behind template-based edge detection is to use a small, discrete template as a model of an edge instead of using a derivative operator directly.
- The template
  - Model the level changes in the edge
  - Attempt to approximate a derivative operator

Review

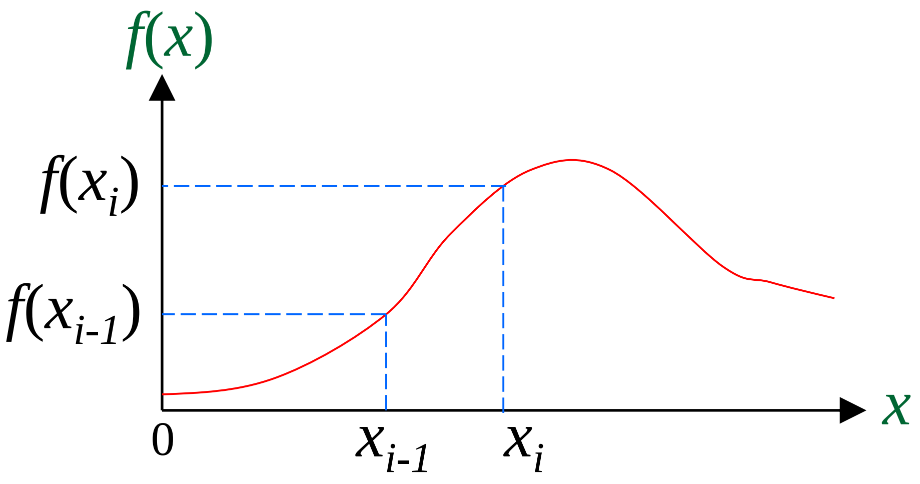
- We start by using **one-dimensional** (1D) signals.
- The 1D signals could just be rows or columns of a 2D image.

- **Mathematic formula** of derivatives :

$$f'(x) = \frac{dy}{dx}, \quad f''(x) = \frac{d^2y}{dx^2}$$

- Given that the signal **S** is a sequence of samples from some function *f*, then

$$f'(x_i) \approx \frac{\Delta f(x)}{\Delta x} = \frac{f(x_i) - f(x_{i-1})}{x_i - x_{i-1}}$$

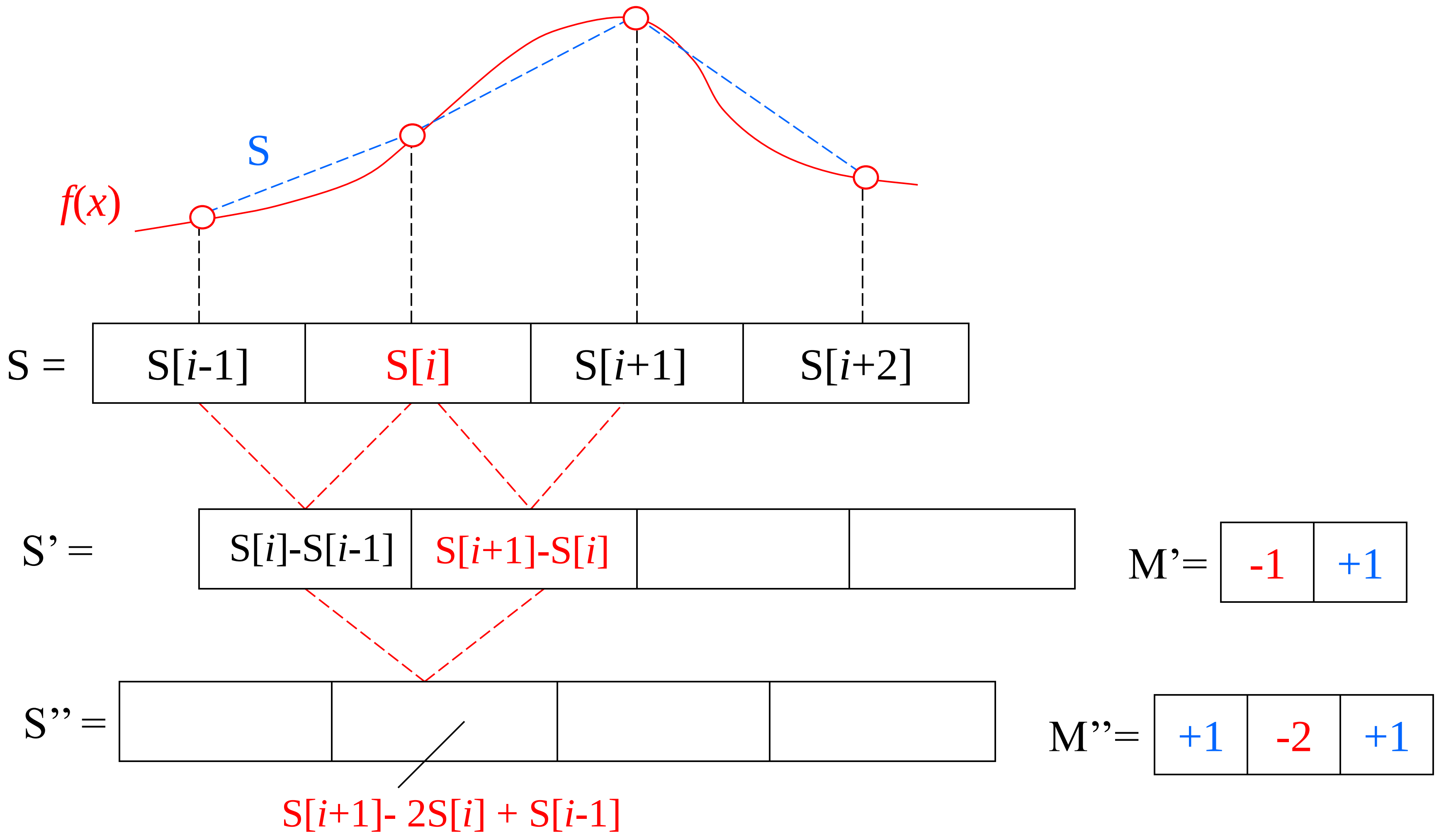


- Assuming that the sample spacing is $\Delta x = 1$, the derivative of $f(x)$ can be approximated by

$$f'(x_i) \approx f(x_i) - f(x_{i-1})$$

- The first (S') and second (S'') difference signals are scaled approximations to the first and second derivatives of the signal S

Masks M' and M" represent the derivative operations



- The first derivative of *f*(*x*) can be approximated by applying the mask M'= [-1, 1] to the samples in S， it can obtain an output signal S'.
- It's convenient to think of the values of S' as occurring in between the samples of S.
- A **high absolute value** of S'[*i*] indicates where the signal is undergoing rapid change or ***high*** contrast.

- Signal S' itself can be differentiated a second time using mask M’ to produce output S”, which corresponds to the second derivative of the original function *f*.
- The approximate second derivative can be computed by applying the mask M'' to the original sequence of samples S.

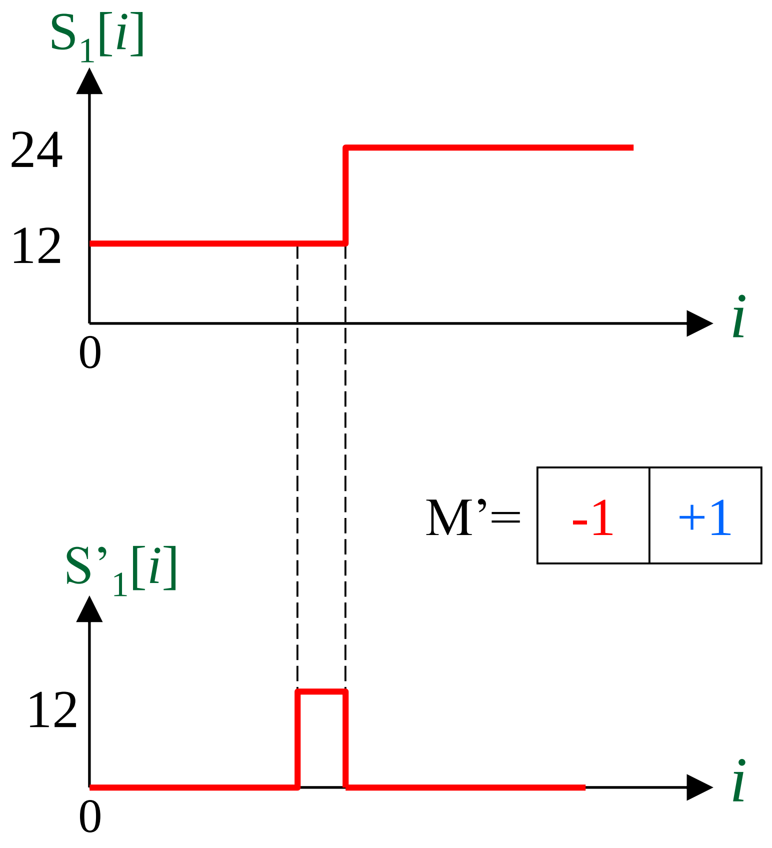
$$S'[i] = -S[i-1] + S[i]$$

$$\text{mask } M' = [-1,+1]$$

$$\begin{aligned} S''[i] &= -S'[i] + S'[i+1] \\ &= -(S[i] - S[i-1]) + (S[i+1] - S[i]) \\ &= S[i-1] - 2S[i] + S[i+1] \end{aligned}$$

$$\text{mask } M'' = [1,-2,1]$$

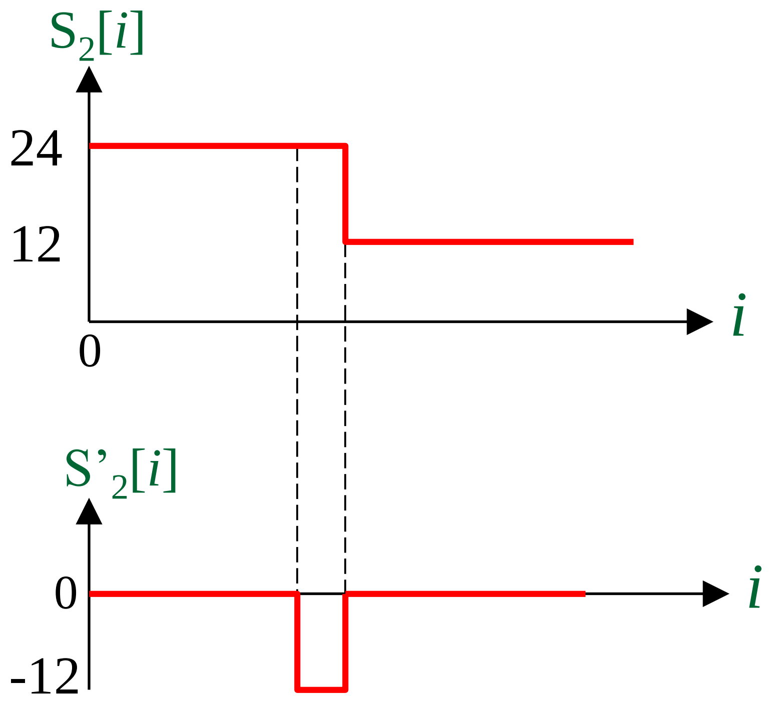
Examples



$M' = \boxed{-1}\boxed{+1}$

$S_1 =$	12	12	12	12	12	24	24	24	24	24
$S_1 \otimes M' =$	0	0	0	0	12	0	0	0	0	0

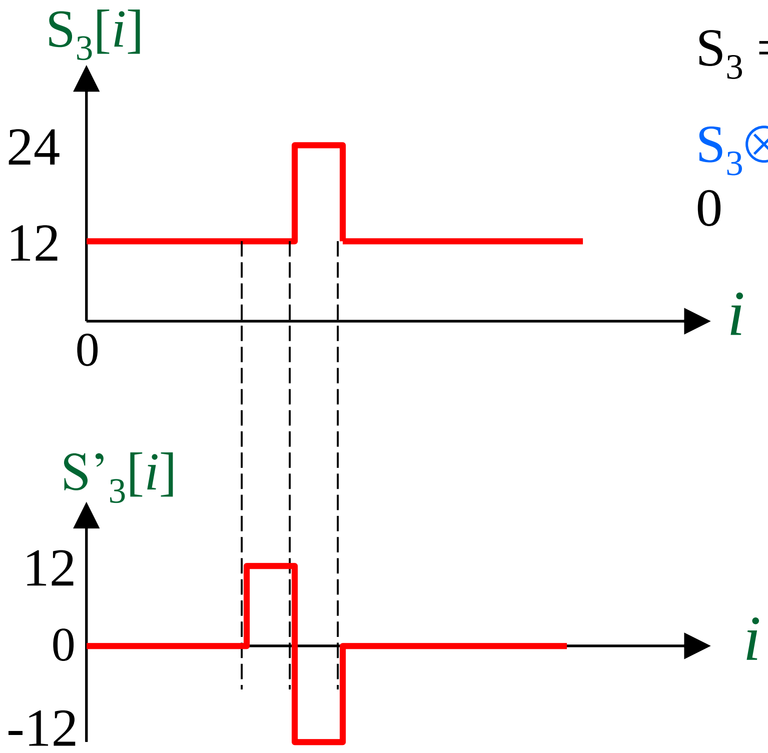
- S_1 is an ***upward step edge***, the first derivative mask M' can be used to detect the edge (border)



$S_2 =$ 24 24 24 24 24 12 12 12 12 12

$S_2 \otimes M' =$ 0 0 0 0 -12 0 0 0 0
0

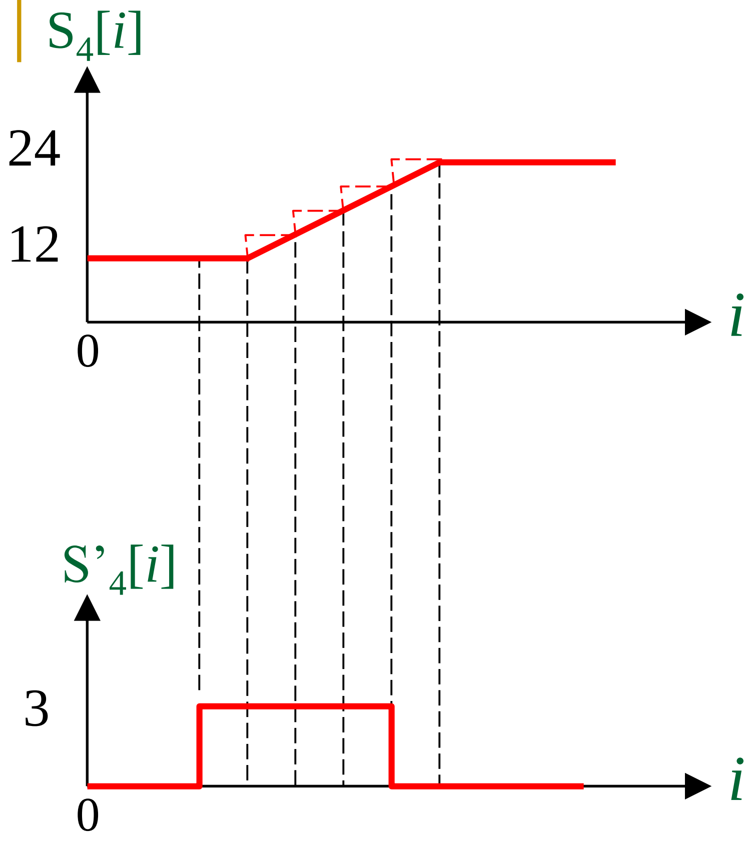
- S_2 is a ***downward step edge***, the first derivative mask M' can be used to detect the edge (border)



$S_3 =$ 12 12 12 12 24 12 12 12 12 12

$S_3 \otimes M' =$ 0 0 0 12 -12 0 0 0 0
0

- S_3 is a ***bright impulse or line***, the first derivative mask M' can be used to detect the edge (border)



$S_4 =$ 12 12 12 12 15 18 21 24 24 24

$S_4 \otimes M' =$ 0 0 0 3 3 3 3 0 0 0

M'=

-1	+1

- S_4 is an ***upward ramp***, the first derivative mask M' can be used to detect the edge (border)

First Derivative Mask M’ = [-1, 0, 1]

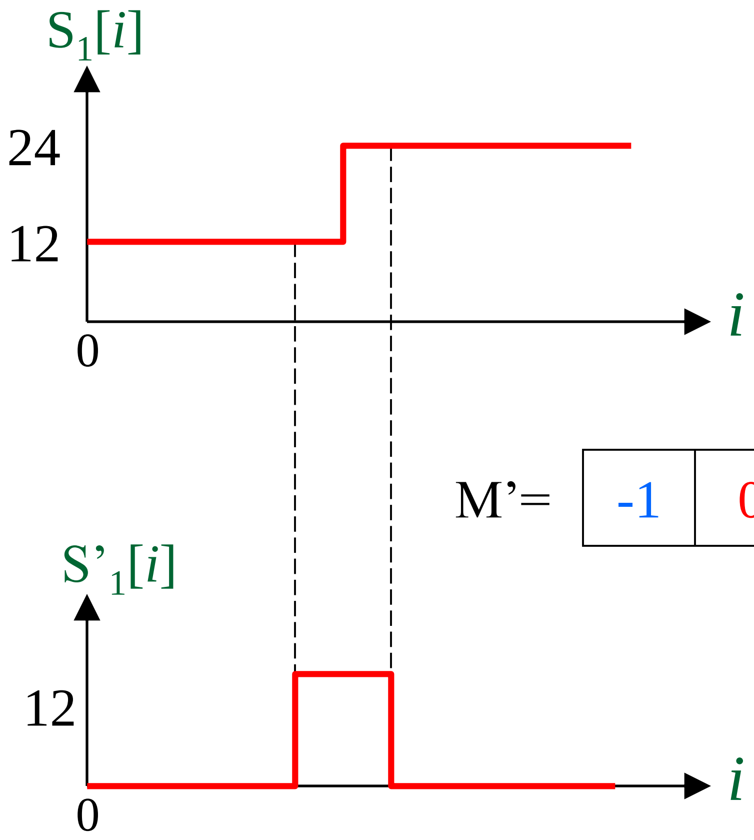
- Use of another common first derivative mask, which has 3 coordinates and is centered at signal point S[i], so that it computes the signal difference across the adjacent values

M’=

-1	0	+1

- Because $\Delta x = 2$, it will give a high estimate of the actual derivative unless the result is divided by 2.
- Moreover, this mask is known to give a response on perfect step edges that is 2 samples wide.

Examples



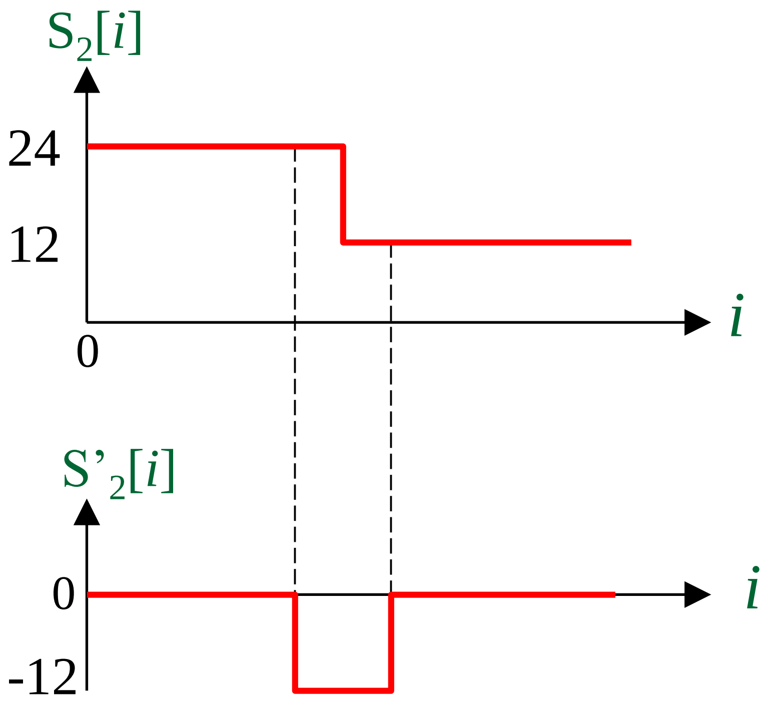
$S_1 =$ 12 12 12 12 12 24 24 24 24 24

$S_1 \otimes M' =$ 0 0 0 0 12 12 0 0 0 0

M'=

-1	0	+1

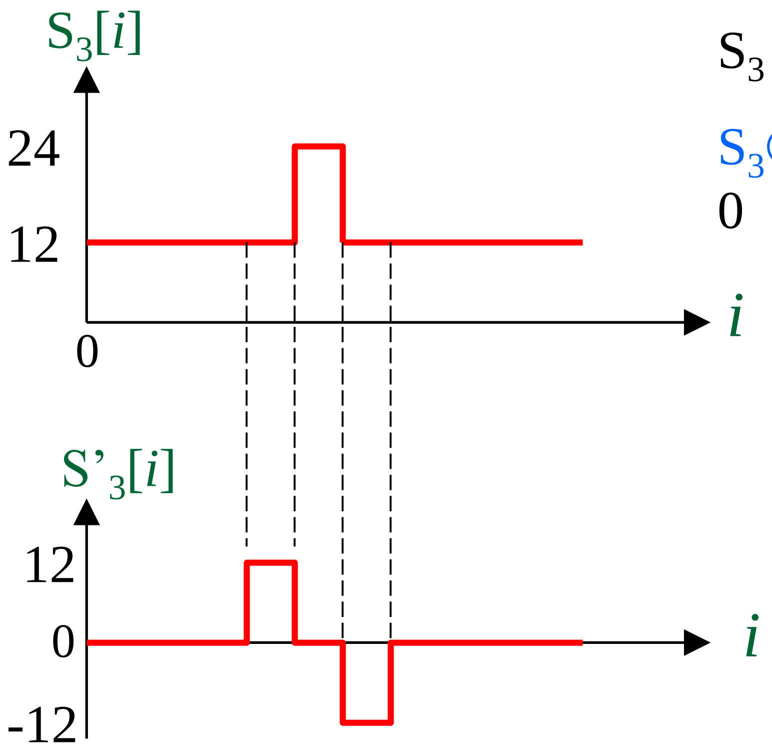
- S_1 is an ***upward step edge***, the first derivative mask M' can be used to detect the edge (border) with 2 samples wide.



$S_2 =$ 24 24 24 24 24 12 12 12 12 12

$S_2 \otimes M' =$ 0 0 0 0 -12 -12 0 0 0 0

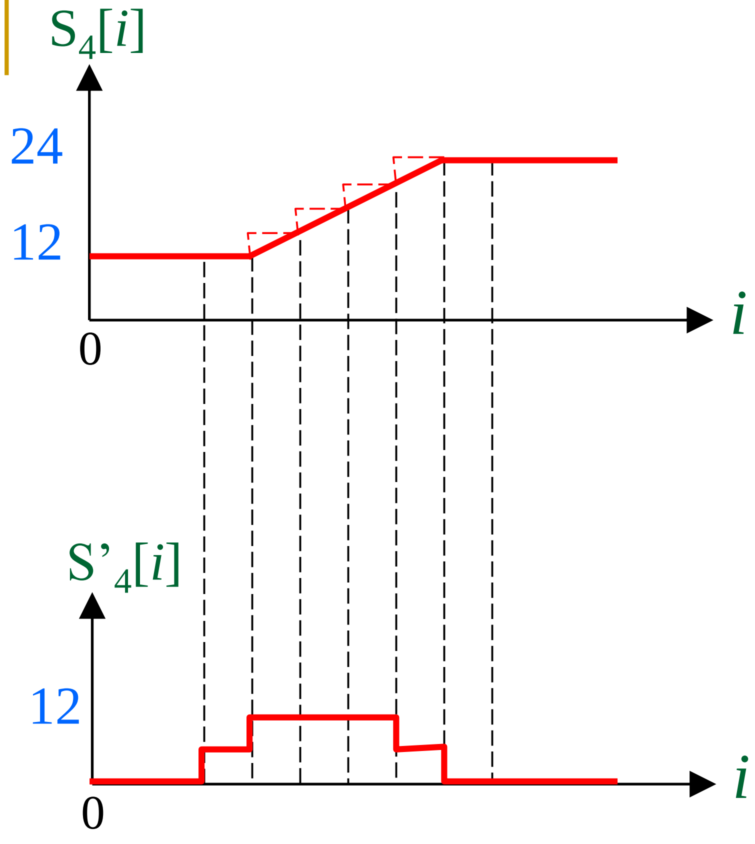
- S_2 is a ***downward step edge***



$S_3 =$ 12 12 12 12 24 12 12 12 12 12

$S_3 \otimes M' =$ 0 0 0 12 0 -12 0 0 0
0

- S_3 is a ***bright impulse or line***



$S_4 =$ 12 12 12 12 15 18 21 24 24 24

$S_4 \otimes M' =$ 0 0 0 3 6 6 6 3 0 0

M'=

-1	0	+1

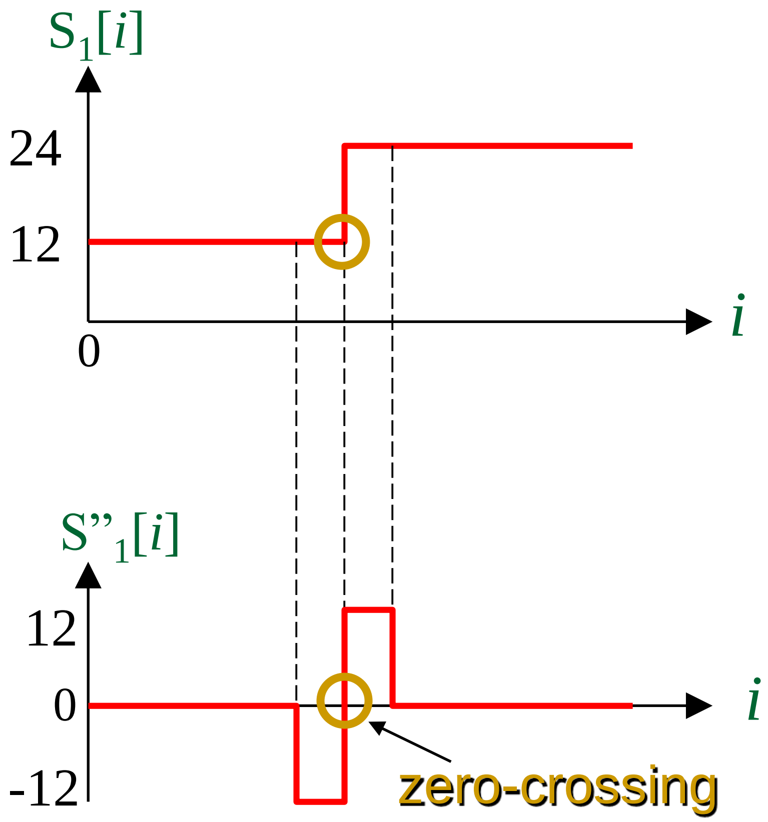
- **S_4 is an *upward ramp***

Second Derivative Mask M” = [-1, 2, -1]

- Use of second derivative mask, signal contrast is detected by a zero-crossing, which localizes and amplifies the change between two successive signal values

M”=

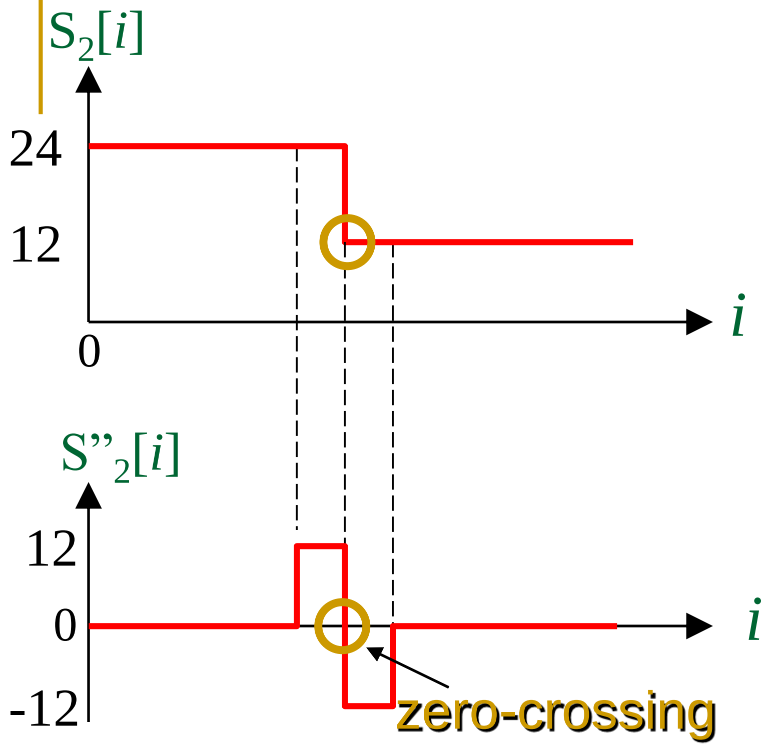
-1	2	-1



$S_1 =$ 12 12 12 12 12 24 24 24 24 24

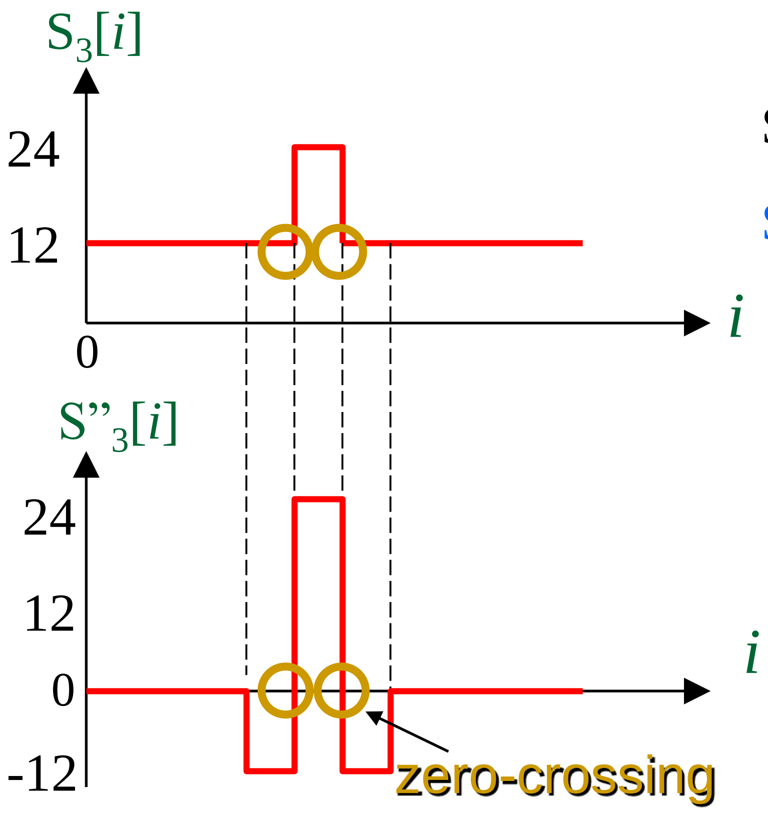
$S_1 \otimes M'' =$ 0 0 0 0 -12 12 0 0 0 0

- S_1 is a ***upward step edge***



$S_2 =$	24	24	24	24	24	12	12	12	12	12
$S_2 \otimes M'' =$	0	0	0	0	12	-12	0	0	0	0

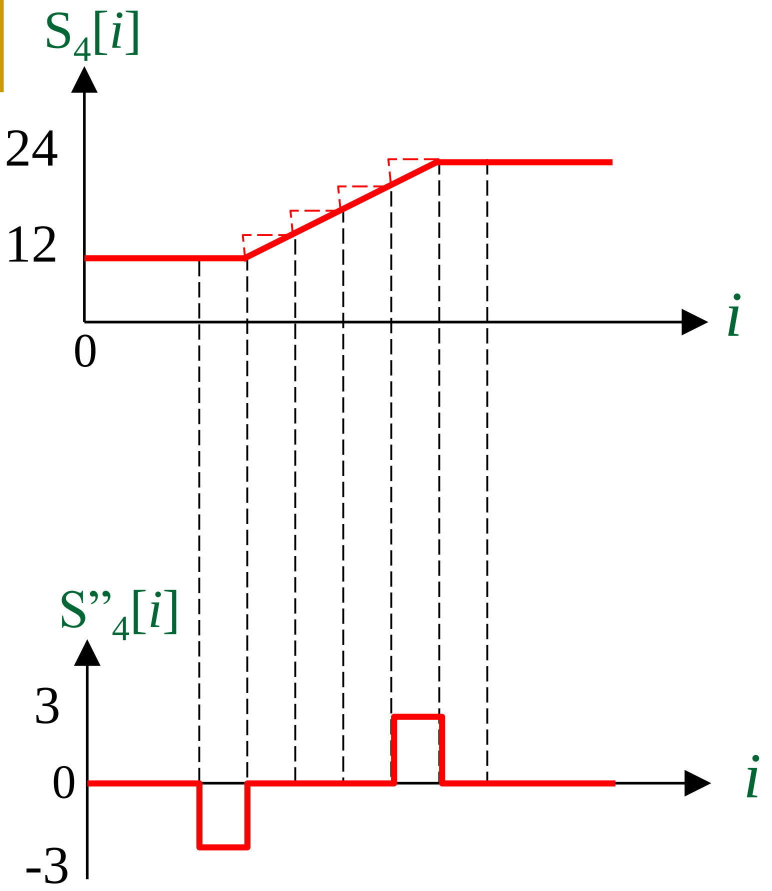
- S_2 is a ***downward step edge***



$S_3 =$ 12 12 12 12 24 12 12 12 12 12

$S_3 \otimes M'' =$ 0 0 0 -12 24 -12 0 0 0 0

- S_3 is a ***bright impulse or line***



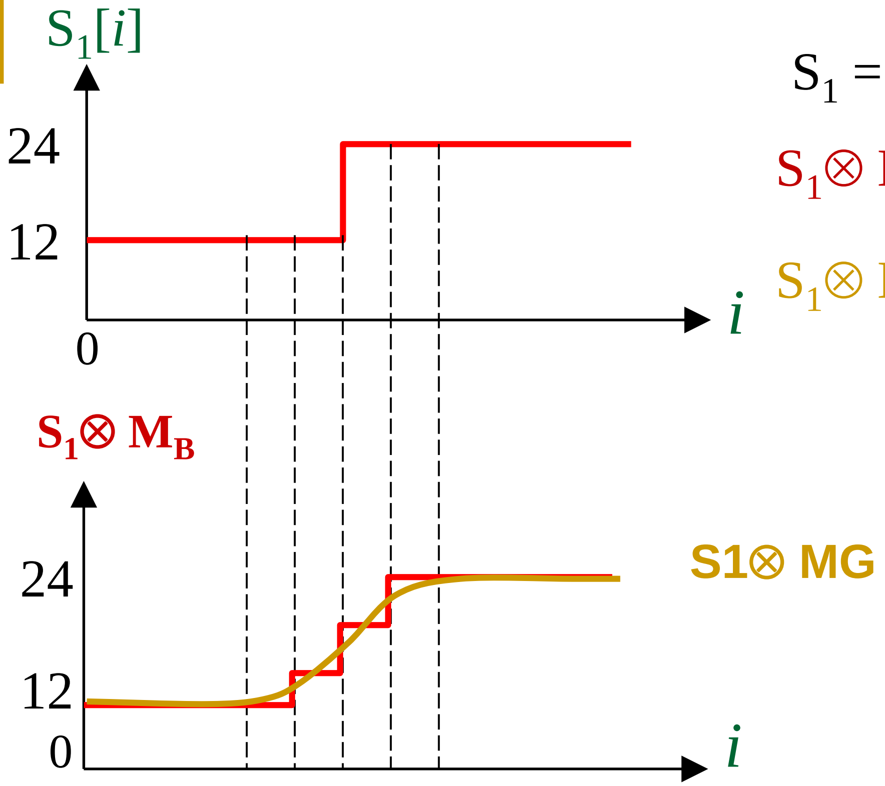
$M'' =$		
-1	2	-1

$S_4 =$	12	12	12	12	15	18	21	24	24	24
$S_4 \otimes M'' =$	0	0	0	-3	0	0	0	3	0	0

- S_4 is an ***upward ramp***

Smoothing Mask

- Smoothing of signals can be put into the same framework as differencing
  - **Box smoothing Mask MB=[1/3, 1/3, 1/3]**
  - **Gaussian smoothing Mask MG=[1/4, 1/2, 1/4]**

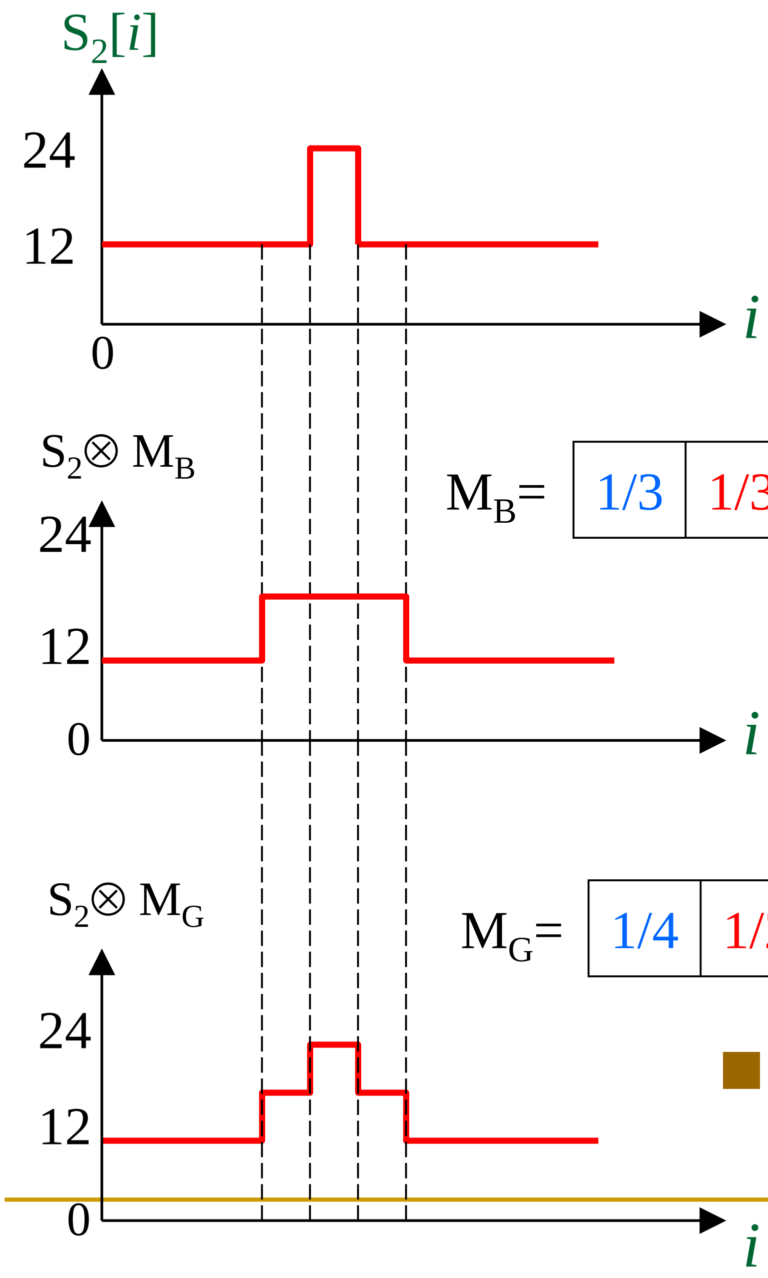


$S_1 =$ 12 12 12 12 12 24 24 24 24 24

$S_1 \otimes M_B =$ 12 12 12 12 16 20 24 24 24 24

$S_1 \otimes M_G =$ 12 12 12 12 15 21 24 24 24 24

- S_1 is a ***upward step edge***



$S_2 =$ 12 12 12 12 24 12 12 12 12 12

$S_2 \otimes M_B =$ 12 12 12 16 16 16 12 12 12 12

$S_2 \otimes M_G =$ 12 12 12 15 18 15 12 12 12 12

$M_B =$

1/3	1/3	1/3

$M_G =$

1/4	1/2	1/4

- S_2 is a ***bright impulse or line***

Some properties of derivative masks

- Coordinates of **derivative masks** have ***opposite*** signs in order to obtain a high response in signal regions of high contrast.
- The sum of coordinates of **derivative masks** is ***zero*** so that a zero response is obtained on constant regions.
- ***First derivative masks*** produce high absolute values at points of high contrast.
- ***Second derivative masks*** produce zero-crossings at points of high contrast.

Some properties of smoothing masks

- Coordinates of **smoothing masks** are ***positive and sum to one*** so that output on constant regions is the same as the input.
- The amount of smoothing and noise reduction is ***proportional to the mask size***.
- Step edges are blurred in ***proportion to the mask size***.

Using masks to estimate the gradient

- **Sobel edge detection**
- Uses templates in the form of convolution masks having the following values

$$s_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$$

$$s_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

The Sobel operator

- Better approximations of the derivatives exist
  - The *Sobel* operators below are very commonly used

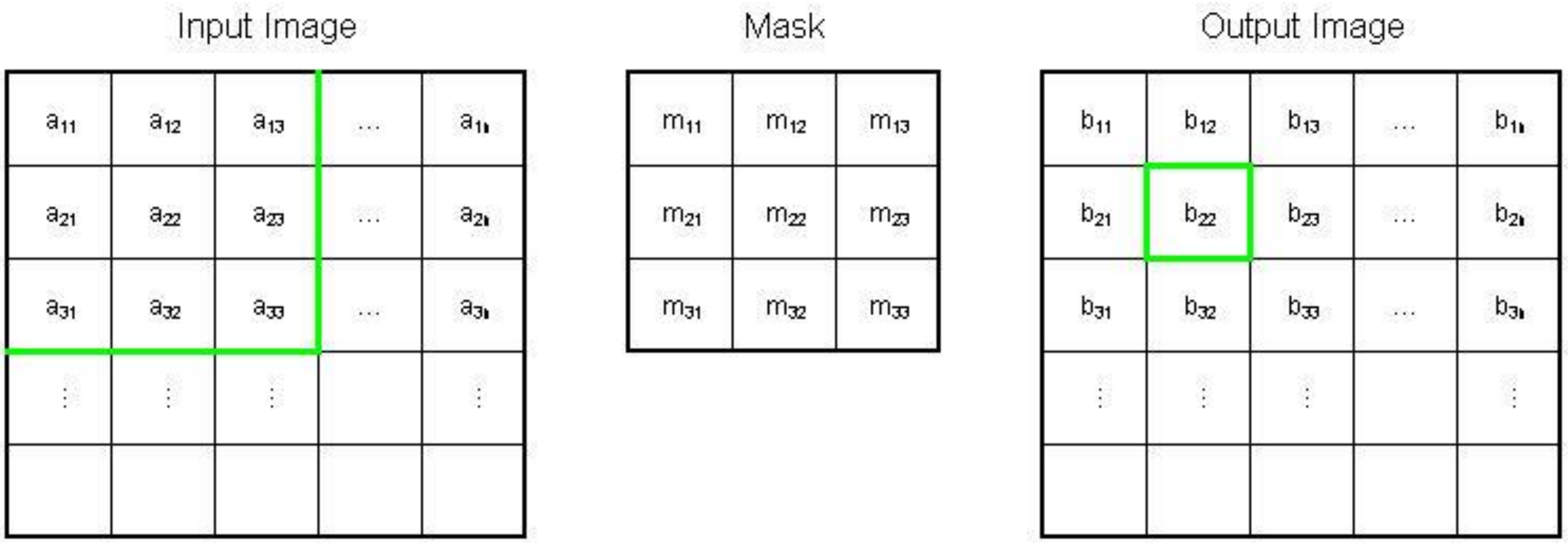
$$\frac{1}{8}\begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline \end{array} \qquad \frac{1}{8}\begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -2 & -1 \\ \hline \end{array}$$

$$s_x \qquad\qquad\qquad s_y$$

  - The standard deviation 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

Principle of Soble edge detection

- The mask is slid over an area of the input image, changes that pixel's value and then shifts one pixel to the right and continues to the right until it reaches the end of a row.
- It then starts at the beginning of the next row.



$b_{22} = (a_{11}*m_{11}) + (a_{12}*m_{12}) + (a_{13}*m_{13}) + (a_{21}*m_{21}) + (a_{21}*m_{21}) + (a22*m_{22}) + (a_{23}*m_{23}) + (a_{31}*m_{31}) + (a_{32}*m_{32}) + (a_{33}*m_{33})$

- The Sx mask highlights the edges in the horizontal direction while the Sy mask highlights the edges in the vertical direction.
- After taking the magnitude of both, the resulting output detects edges in both directions.

- After Sx and Sy are computed for every pixel in an image, we have two image files Gx and Gy.
- The resulting magnitudes must be thresholded.

- The magnitude of the gradient is then calculated using the formula:

$$|G| = \sqrt{Gx^2 + Gy^2}$$

- An approximate magnitude can be calculated using:

$$|G| = |Gx| + |Gy|$$

- ***It is important to notice that pixels in the first and last rows, as well as the first and last columns cannot be manipulated by a 3x3 mask. This is because when placing the center of the mask over a pixel in the first row (for example), the mask will be outside the image boundaries.***

Algorithm

1. Calculate the thresholding $$T = \frac{f_{\max} - f_{\min}}{2}$$
2. Apply Sobel masks to the whole image, respectively.

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

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

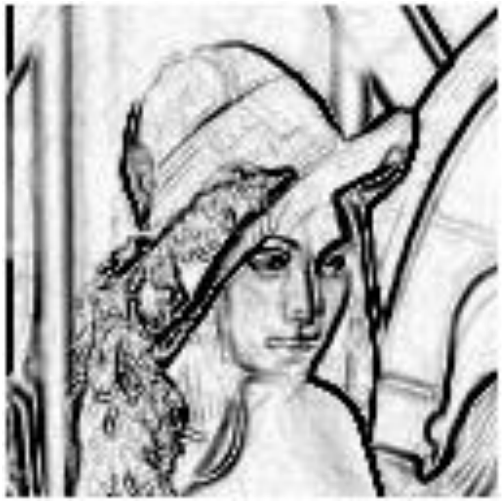
3. There are two result images Gx and Gy. Calculate

$$|G| = |Gx| + |Gy|$$

4. If $|G|>T$, then *(x, y)* is edge;

   otherwise, *(x, y)* is not edge

Detect edges using the Sobel method



Gradient operators

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

(a)

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

(b)

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

(c)

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

(d)

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

Roberts Operator

- Mark edge point only
- No information about edge orientation
- Work best with binary images
- Primary disadvantage:
  - High sensitivity to noise
  - Few pixels are used to approximate the gradient

- Prewitt masks

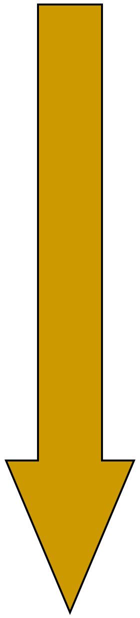
$$M_x = \begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -1 & 0 & 1 \\ \hline -1 & 0 & 1 \\ \hline \end{array} \qquad M_y = \begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -1 & -1 \\ \hline \end{array}$$

Comparing Edge Operators

Gradient:

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

Good Localization
Noise Sensitive
Poor Detection



Poor Localization
Less Noise Sensitive
Good Detection

Roberts (2 x 2):

0	1
-1	0

1	0
0	-1

Prewitt (3 x 3):

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

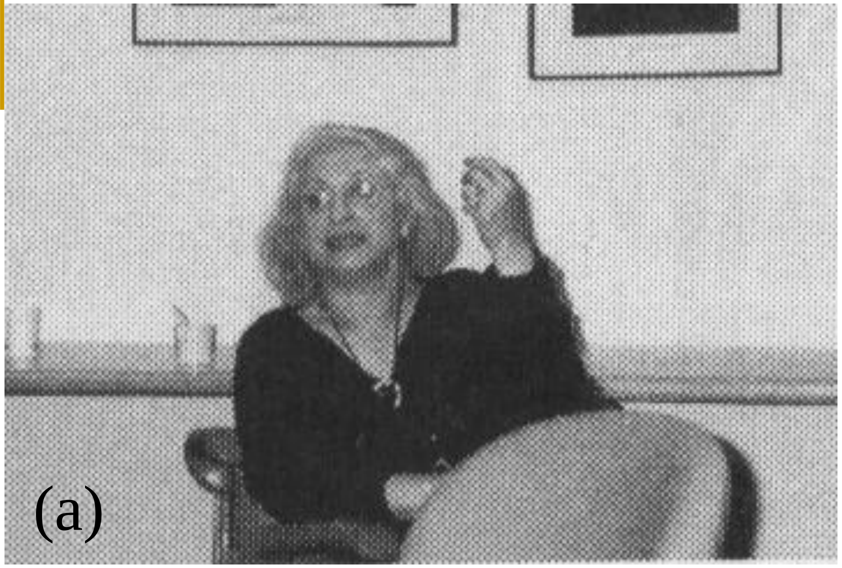
1	1	1
0	0	0
-1	-1	1

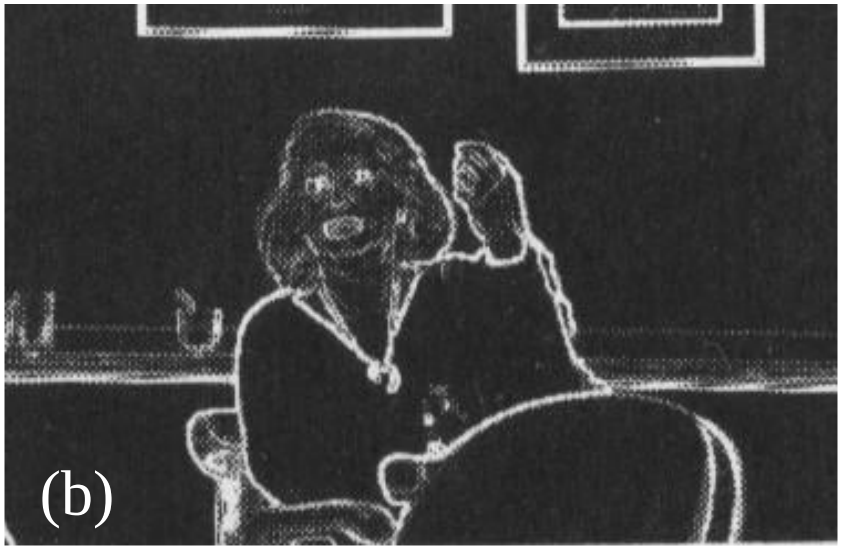
Sobel (5 x 5):

-1	-2	0	2	1
-2	-3	0	3	2
-3	-5	0	5	3
-2	-3	0	3	2
-1	-2	0	2	1

1	2	3	2	1
2	3	5	3	2
0	0	0	0	0
-2	-3	-5	-3	-2
-1	-2	-3	-2	-1

Example



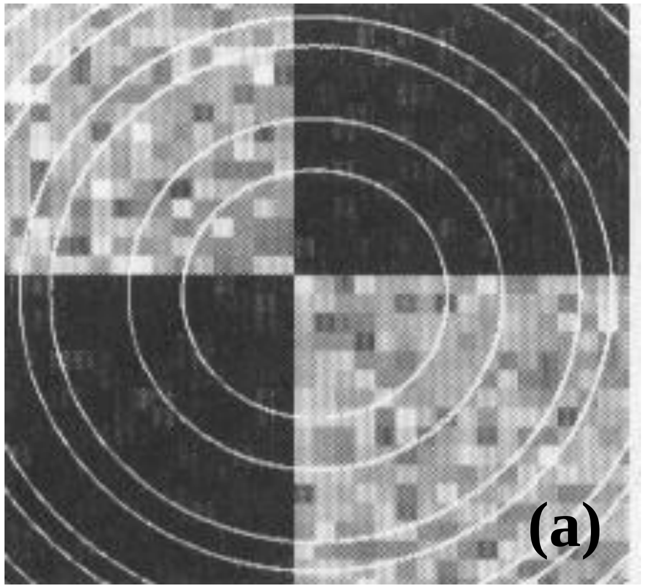


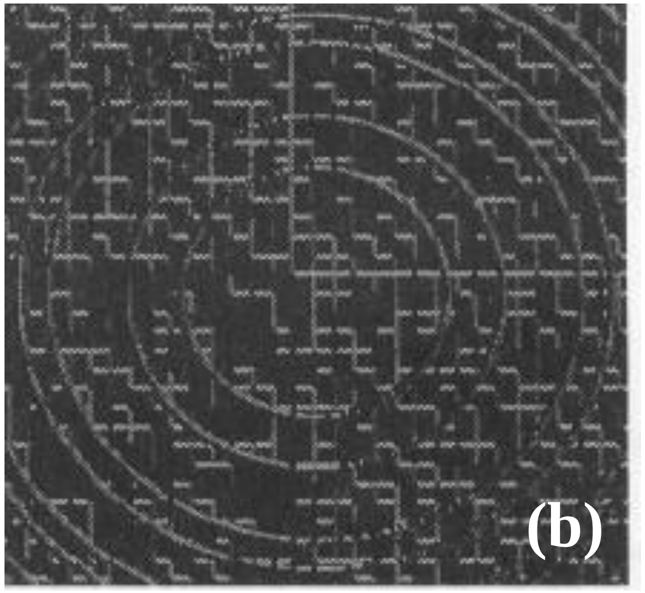
(a) Image of Judith Prewitt

(b) Gradient image showing result using the Prewitt 3×3 operator

Sobel masks: the Sobel operator represents many, but not all, of the image edges.

$$M_x = \begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline \end{array} \qquad M_y = \begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -2 & -1 \\ \hline \end{array}$$

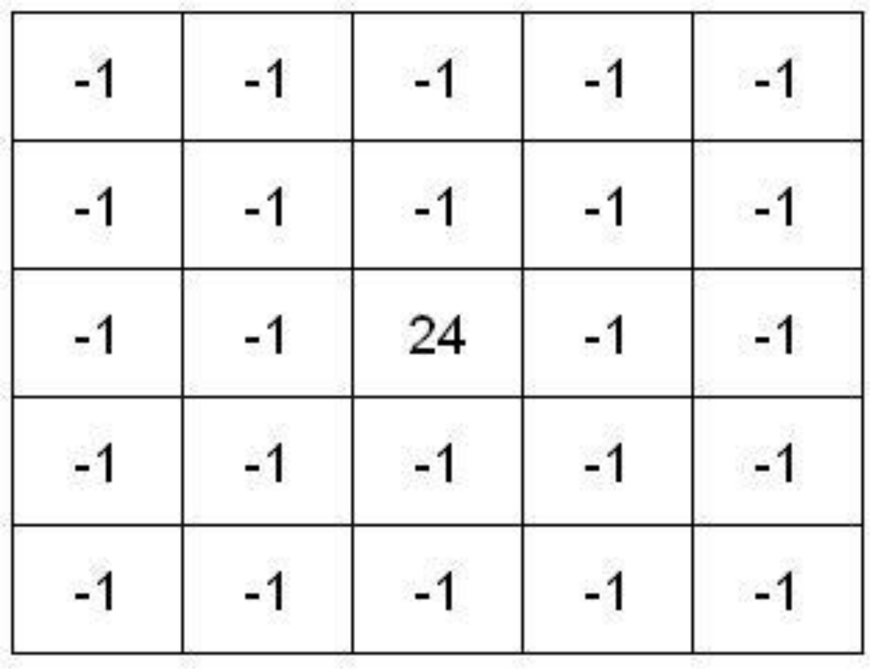




(a) Image of noisy squared and rings, (b) Edges detected by 3×3 Sobel operator.

Laplacian Edge Detector

- The 5x5 Laplacian used is a convoluted mask to approximate the second derivative, unlike the Sobel method which approximates the gradient.
- Instead of 2 3x3 Sobel masks, one for the x and y direction, Laplace uses 1 5x5 mask for the 2nd derivative in both the x and y directions.



-1	-1	-1	-1	-1
-1	-1	-1	-1	-1
-1	-1	24	-1	-1
-1	-1	-1	-1	-1
-1	-1	-1	-1	-1

Laplacian operators

0	−1	0
−1	4	−1
0	−1	0

−1	−1	−1
−1	8	−1
−1	−1	−1

5	5	5	5	5	5	5
4	5	5	5	5	5	5
3	4	5	5	5	5	5
3	3	4	5	5	5	5
3	3	3	4	4	4	4
3	3	3	3	3	3	3
3	3	3	3	3	3	3

Image

2	0	0	0	0
0	2	0	0	0
−2	0	2	1	1
0	−2	1	0	0
0	0	−1	−1	−1

4	1	0	0	0
0	4	1	0	0
−4	0	5	3	3
−1	−4	2	0	0
0	−1	−2	−3	−3

Image after Laplacian operator

- However, because these masks are approximating a second derivative measurement on the image, they are very sensitive to noise.

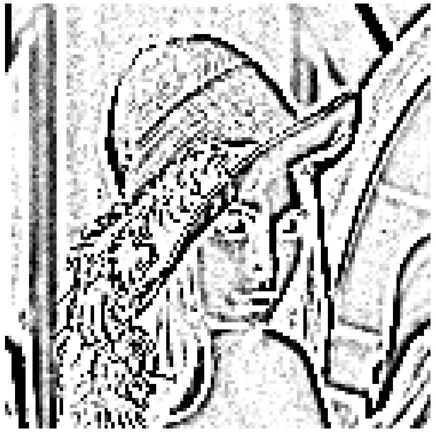
- Edge detection by Laplacian operator followed by **zero-crossing detection**
- If in the neighborhood (3x3, 5x5, 7x7, etc.) of a given pixel there exist both polarities, i.e., pixel values greater than and smaller than 0, then the pixel is a zero-crossing. (Note that the pixel in question does not have to be zero.)
- Specifically, we find the maximum and minimum among all pixels in the neighborhood of a pixel under consideration. If the maximum is greater than zero and the minimum is smaller than zero, the pixel is a zero-crossing.

- Due to the existence of random noise some false zero-crossing may be detected. In this case we check whether the **difference between the maximum and the minimum is greater than a threshold value**. If so the pixel is on an edge, otherwise the zero-crossing is assumed to be caused by noise and suppressed.

Algorithm

- Apply Laplacian mask to the whole image.
- Zero-crossing process.
  - Find the maximum and minimum among all pixels in the neighborhood of a pixel.
  - Using either 1 or 2 for the zero-crossing process
  1. If the maximum is greater than zero and the minimum is smaller than zero, the pixel is a zero-crossing.
  2. If the difference between the maximum and the minimum is greater than a threshold value, the pixel is on an edge.

Detect edges using the Laplacian method



Performance

- Sobel and Prewitt methods are very effectively providing good edge maps.
- Roberts and Laplacian methods are not very good as expected.

Canny edge detector

- In 1986, John Canny defined a set of goals for an edge detector and described an optimal method for achieving them

- Canny specified three issues that an edge detector must address
  - **Error rate:** The edge detector should respond only to edges, and should find all of them; no edges should be missed.
  - **Localization:** The distance between the edge pixels found by the edge detector and the actual edge should be as small as possible
  - **Response:** The edge detector should not identify multiple edge pixels where only a single edge exists

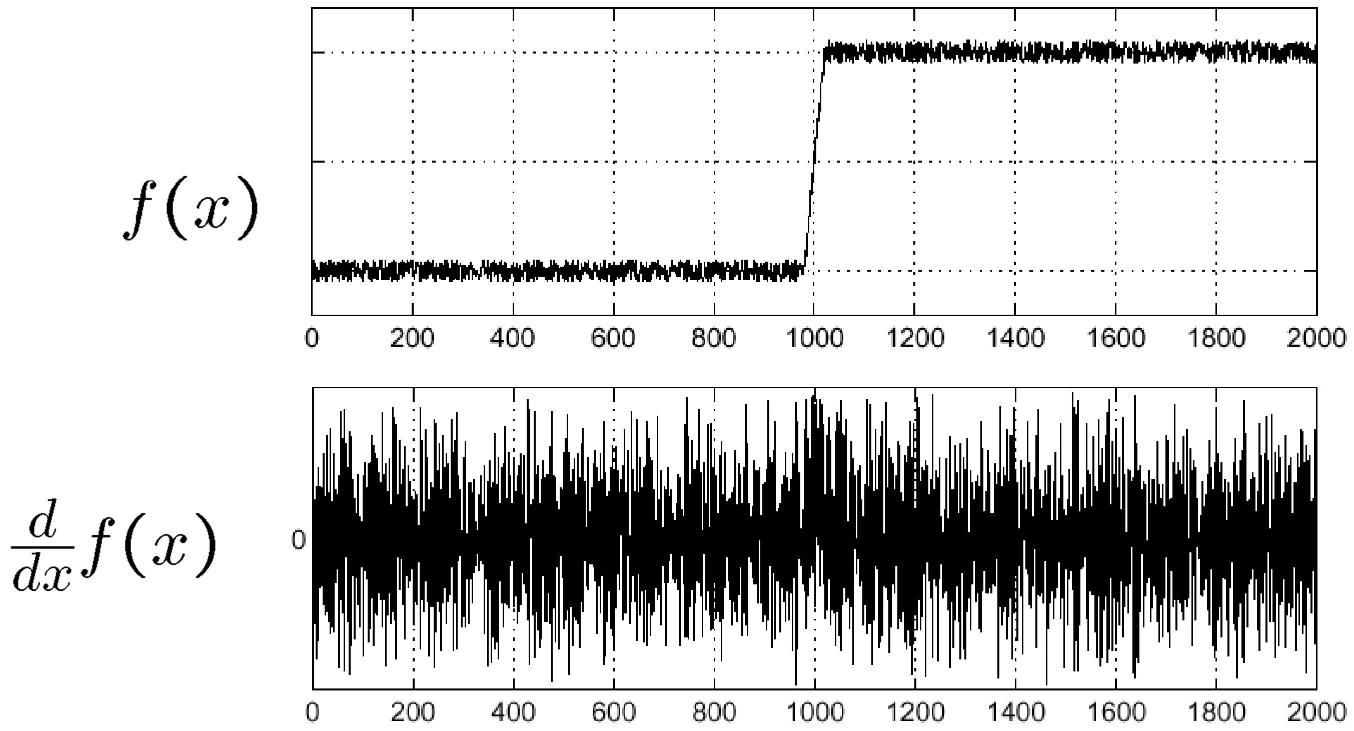
- The Canny edge detection algorithm is known to many as the optimal edge detector.
- Canny's intentions were to enhance the many edge detectors already out at the time he started his work.
- He was very successful in achieving his goal and his ideas and methods can be found in his paper, ***"A Computational Approach to Edge Detection".***

Some practical issues

- The differential masks act as high-pass filters which tend to amplify noise
- To reduce the effects of noise, the image needs to be smoothed first with a lowpass filter
- A larger filter reduces noise, but worsens localization (i.e., it adds uncertainty to the location of the edge) and vice versa

Effects of Noise

- Consider a single row or column of the image
  - Plotting intensity as a function of position gives a signal



Where is the edge??

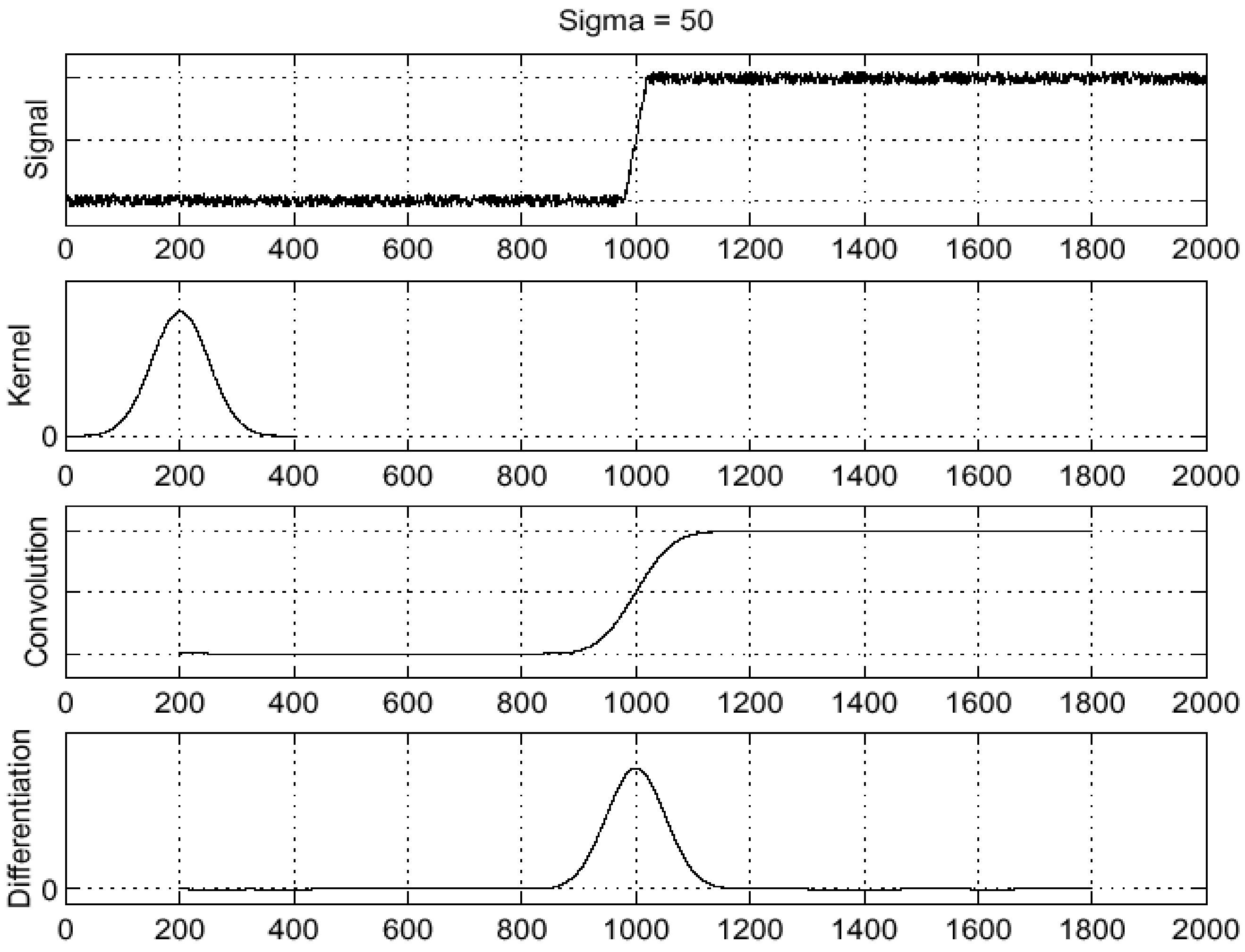
Solution: Smooth First

f

h

$h \star f$

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



Where is the edge?

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

- Based on the criteria, the canny edge detector first smooths the image to eliminate the noise.
- It then finds the image gradient to highlight regions with high spatial derivatives.
- The algorithm then tracks along these regions and suppresses any pixel that is not at the maximum (nonmaximum suppression).
- The gradient array is further reduced by hysteresis (後處理，減少假邊緣數量的方法).

- Hysteresis is used to track along the remaining pixels that have not been suppressed.
- Hysteresis uses **two thresholds** and if the magnitude is below the first threshold, it is set to zero (made a nonedge).
- If the magnitude is above the high threshold, it is made an edge.
- And if the magnitude is between the 2 thresholds, then it is set to zero unless there is a path from this pixel to a pixel with a gradient above high threshold

Canny edge detector algorithm

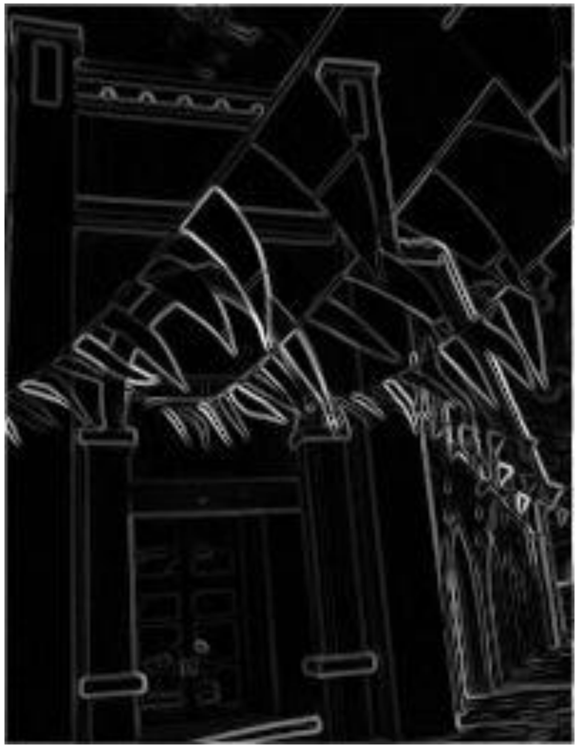
1. The first step is to filter out any noise in the original image before trying to locate and detect any edges.
   - Because the Gaussian filter can be computed using a simple mask, it is used exclusively in the Canny algorithm.
   - **The larger the width of the Gaussian mask, the lower is the detector's sensitivity to noise**.
   - The localization error in the detected edges also increases slightly as the Gaussian width is increased.

$$\frac{1}{115}\begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix}$$

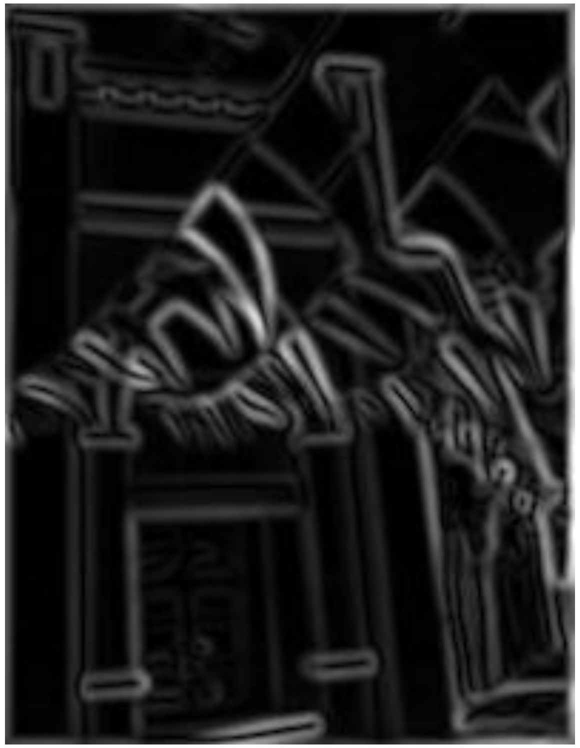
Figure 3 Discrete approximation to Gaussian function with $\sigma=1.4$

Effect of σ (Gaussian kernel size)

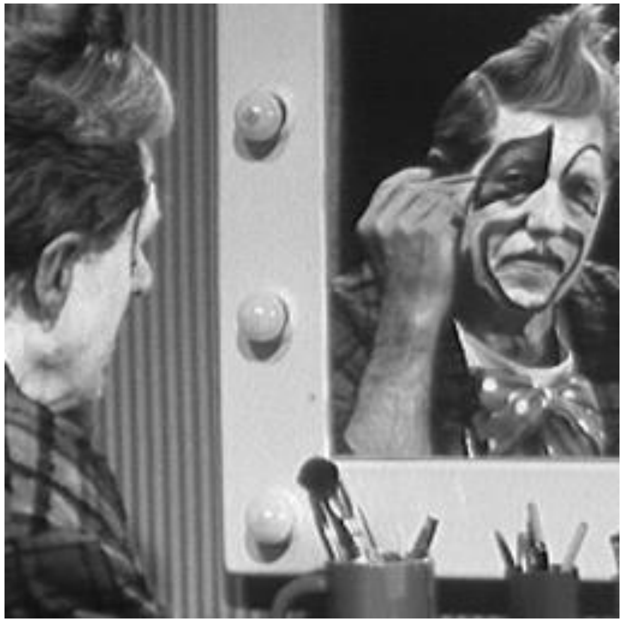
sigma = 0.5

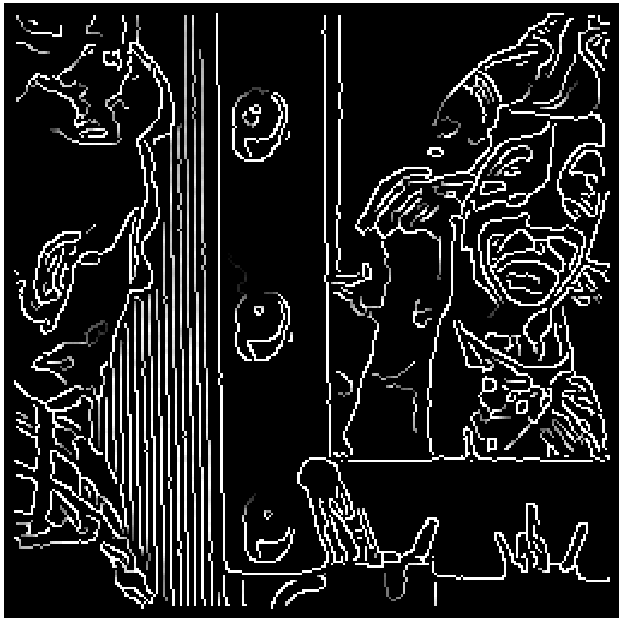


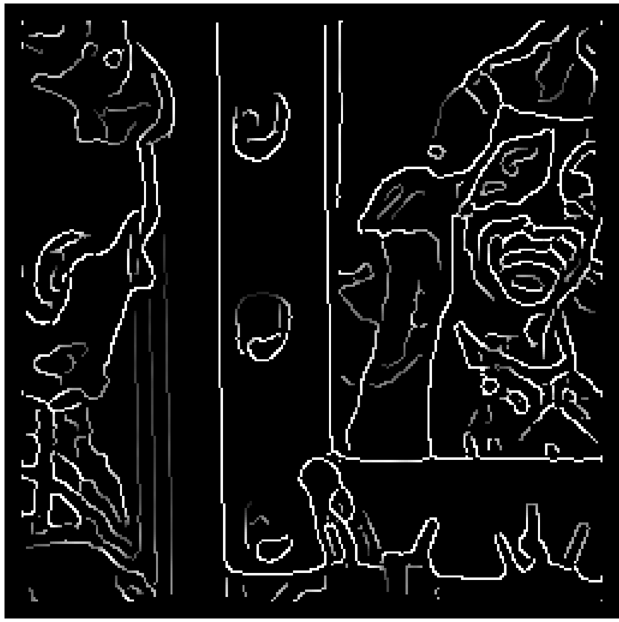
sigma = 2



Edge images after Gaussian and Sobel


original


Canny with $\sigma = 1$


Canny with $\sigma = 2$

- The choice of σ depends on desired behavior
  - large σ detects large scale edges
  - small σ detects fine features

2. After smoothing the image and eliminating the noise, the next step is to find the edge strength by taking the gradient of the image.
   - The Sobel operator performs a 2-D spatial gradient measurement on an image.
   - Then, the approximate absolute gradient magnitude (edge strength) at each point can be found.

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

Gx

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

Gy

The magnitude, or EDGE STRENGTH, of the gradient is then approximated using the formula:

$$|G| = |Gx| + |Gy|$$

3. Finding the edge direction： it is trivial once the gradient in the x and y directions are known. However, you will ***generate an error whenever*** G_x ***is equal to zero***.

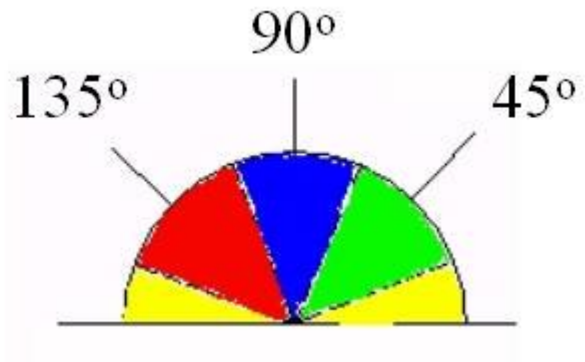
- So in the code there has to be a restriction set whenever this takes place. Whenever the gradient in the x direction is equal to zero, the edge direction has to be equal to 90 degrees or 0 degrees, depending on what the value of the gradient in the y-direction is equal to.
- If **Gy has a value of zero, the edge direction will equal 0 degrees**. **Otherwise the edge direction will equal 90 degrees**. The formula for finding the edge direction is just:

$$\theta = arctg(G_y / G_x)$$

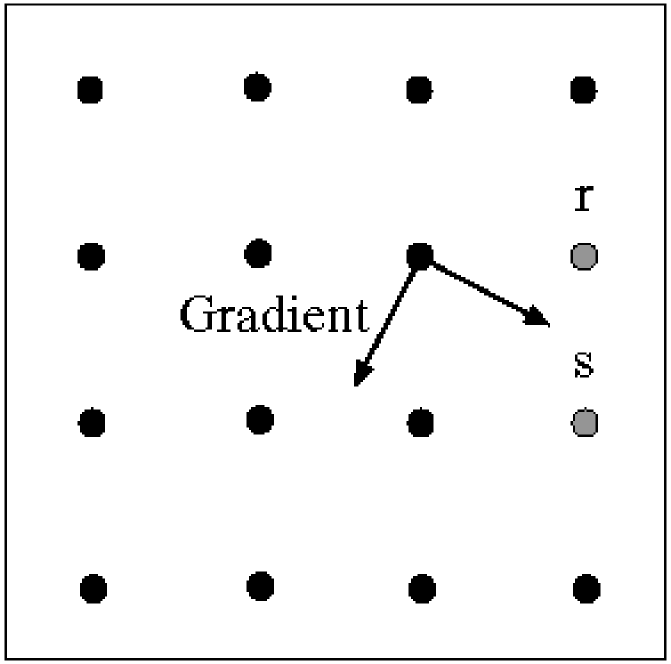
4. Once the edge direction is known, the next step is to relate the edge direction to a direction that can be traced in an image. So if the pixels of a 5x5 image are aligned as follows:

x	x	x	x	x
x	x	x	x	x
x	x	a	x	x
x	x	x	x	x
x	x	x	x	x

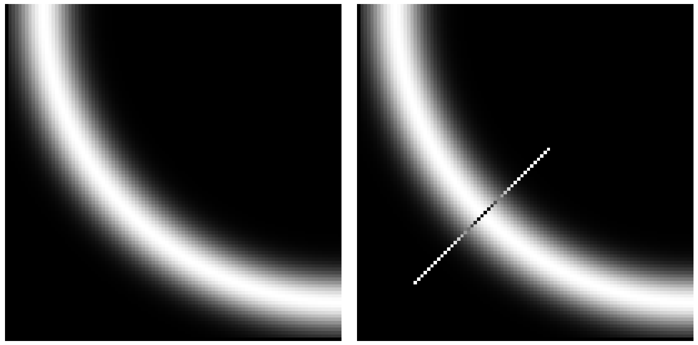
- Then, it can be seen by looking at pixel "a", there are only four possible directions when describing the surrounding pixels - 0 degrees (in the horizontal direction), 45 degrees (along the positive diagonal), 90 degrees (in the vertical direction), or 135 degrees (along the negative diagonal). So now the edge orientation has to be resolved into one of these four directions depending on which direction it is closest to (e.g. if the orientation angle is found to be 3 degrees, make it zero degrees). Think of this as taking a semicircle and dividing it into 5 regions.



- Therefore, any edge direction falling within the **yellow range** (0 to 22.5 & 157.5 to 180 degrees) is set to 0 degrees.
- Any edge direction falling in the **green range** (22.5 to 67.5 degrees) is set to 45 degrees.
- Any edge direction falling in the **blue range** (67.5 to 112.5 degrees) is set to 90 degrees.
- And finally, any edge direction falling within the **red range** (112.5 to 157.5 degrees) is set to 135 degrees.



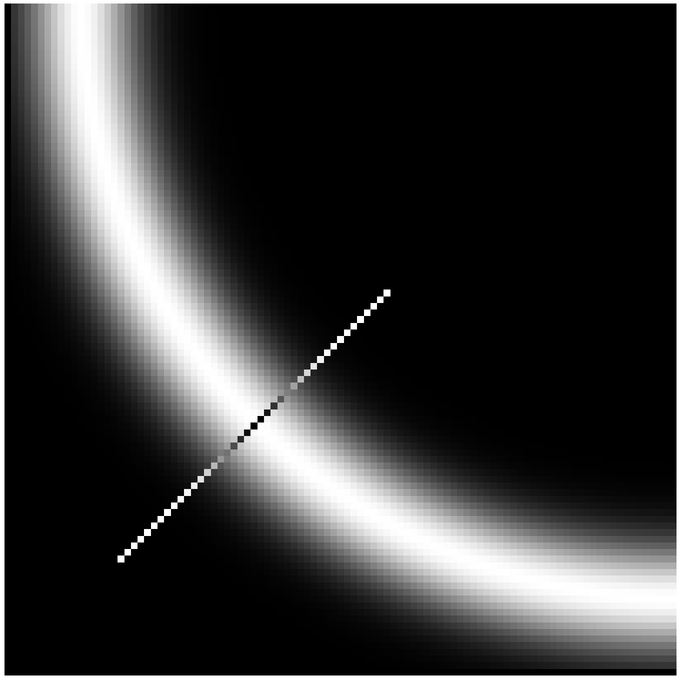
Predicting the next edge point

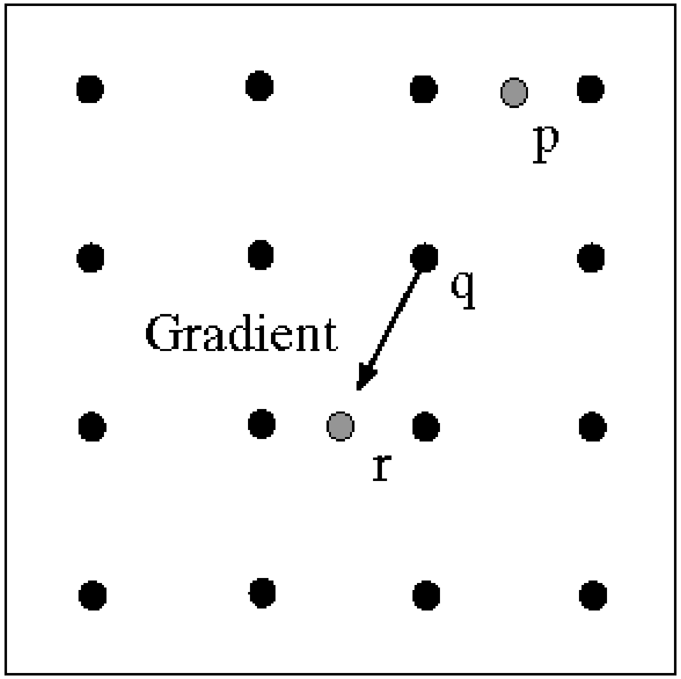


Assume the marked point is an edge point. Then we construct the tangent to the edge curve (which is normal to the gradient at that point) and use this to predict the next points (here either r or s).

5. After the edge directions are known, nonmaximum suppression now has to be applied. Nonmaximum suppression is used to trace along the edge in the gradient direction and suppress any pixel value (sets it equal to 0) that is not considered to be an edge. This will give a thin line in the output image.

Non-maximum Suppression





- Check if pixel is local maximum along gradient direction
  - requires checking interpolated pixels p and r

Algorithm

- For each pixel (x,y) do:

```
if $magn(iq, jq) < magn(ip, jp)$ or $magn(iq, jq) < magn(ir, jr)$
then $I(x, y) = 0$
else $I(x, y) = magn(iq, jq)$
```

- To find the edge points, we need to find the local maxima of the gradient magnitude.
- Broad ridges must be thinned so that only the magnitudes at the points of greatest local change remain.
- All values along the direction of the gradient that are not peak values of a ridge are suppressed.

- The output of non-maxima suppression still contains the local maxima created by noise.
- Can we get rid of them just by using a single threshold?
    - if we set a low threshold, some noisy maxima will be accepted too.
    - if we set a high threshold, true maxima might be missed (the value of true maxima will fluctuate above and below the threshold, fragmenting the edge).

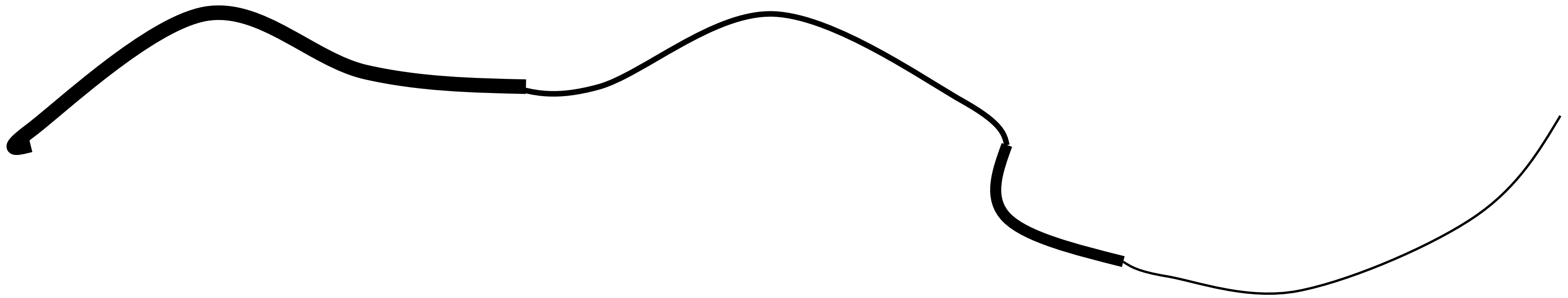
- Finally, hysteresis is used as a means of eliminating streaking.
- Streaking is the breaking up of an edge contour caused by the operator output fluctuating above and below the threshold.
- If a single threshold, T1 is applied to an image, and an edge has an average strength equal to T1, then due to noise, there will be instances where the edge dips below the threshold.

- Equally it will also extend above the threshold making an edge look like a dashed line.
- To avoid this, hysteresis uses 2 thresholds, a high and a low.
  - Any pixel in the image that has a value **greater than T2 is presumed to be an edge pixel**, and is marked as such immediately.
  - Then, any pixels that are **connected to this edge pixel and that have a value greater than T1 are also selected as edge pixels**.
  - If you think of following an edge, you need a gradient of T2 to start but you don't stop till you hit a gradient below T1.

- a low threshold *tl*
- a high threshold *th*
- usually, *th* » 2*tl*

Hysteresis

- Check that maximum value of gradient value is sufficiently large
  - drop-outs? use **hysteresis**
    - use a high threshold to start edge curves and a low threshold to continue them.



Algorithm

1. Produce two thresholded images $I1(i, j)$ and $I2(i, j)$.

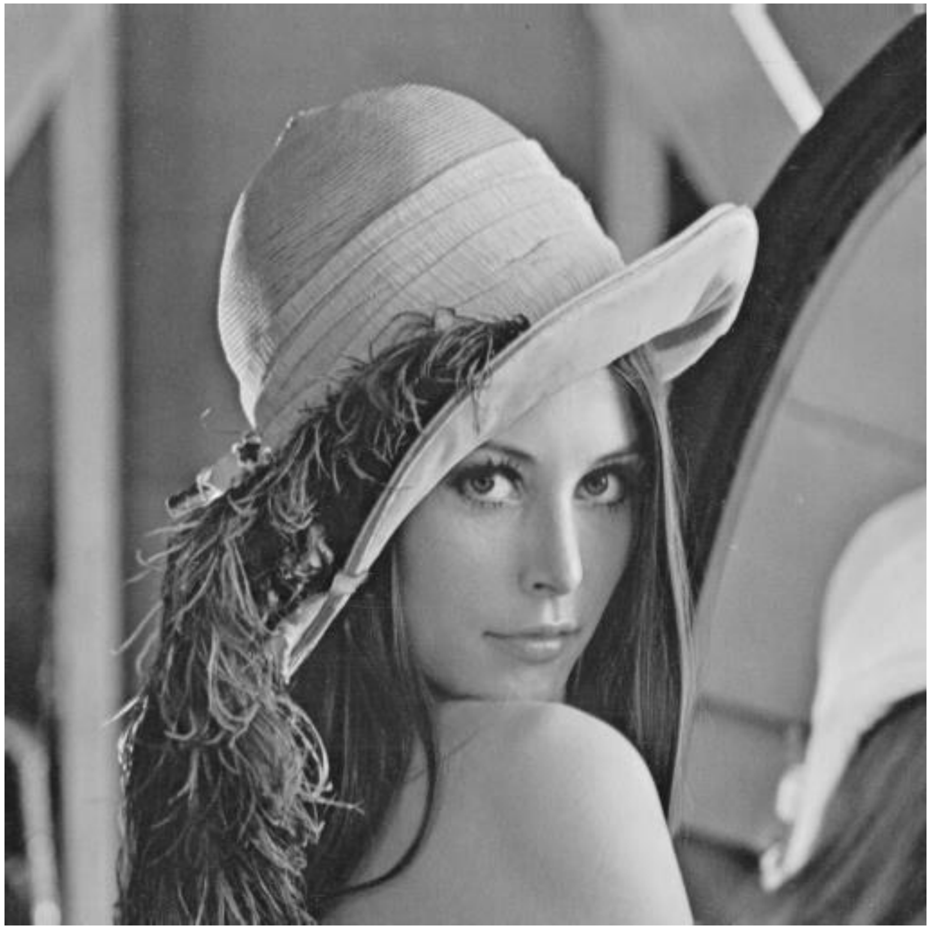
   (note: since $I2(i, j)$ was formed with a high threshold, it will contain fewer false edges but there might be gaps in the contours)

2. Link the edges in $I2(i, j)$ into contours
   - Look in $I1(i, j)$ when a gap is found.
   - By examining the 8 neighbors in $I1(i, j)$, gather edge points from $I1(i, j)$ until the gap has been bridged to an edge in $I2(i, j)$.

Canny Algorithm

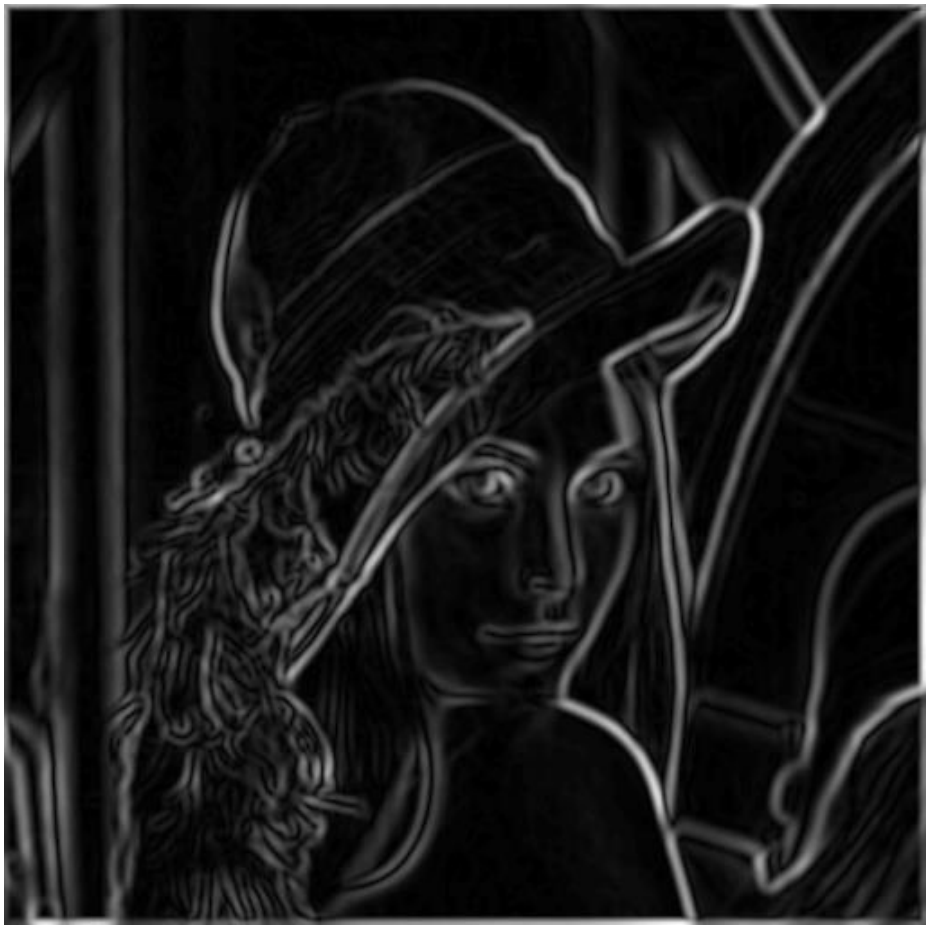
1. Apply Gaussian smoothing.
2. Apply Sobel edge detector. Compute the gradient magnitude and edge direction
3. Apply non-maxima suppression.
4. Apply hysteresis thresholding/edge linking.

The Canny edge detector

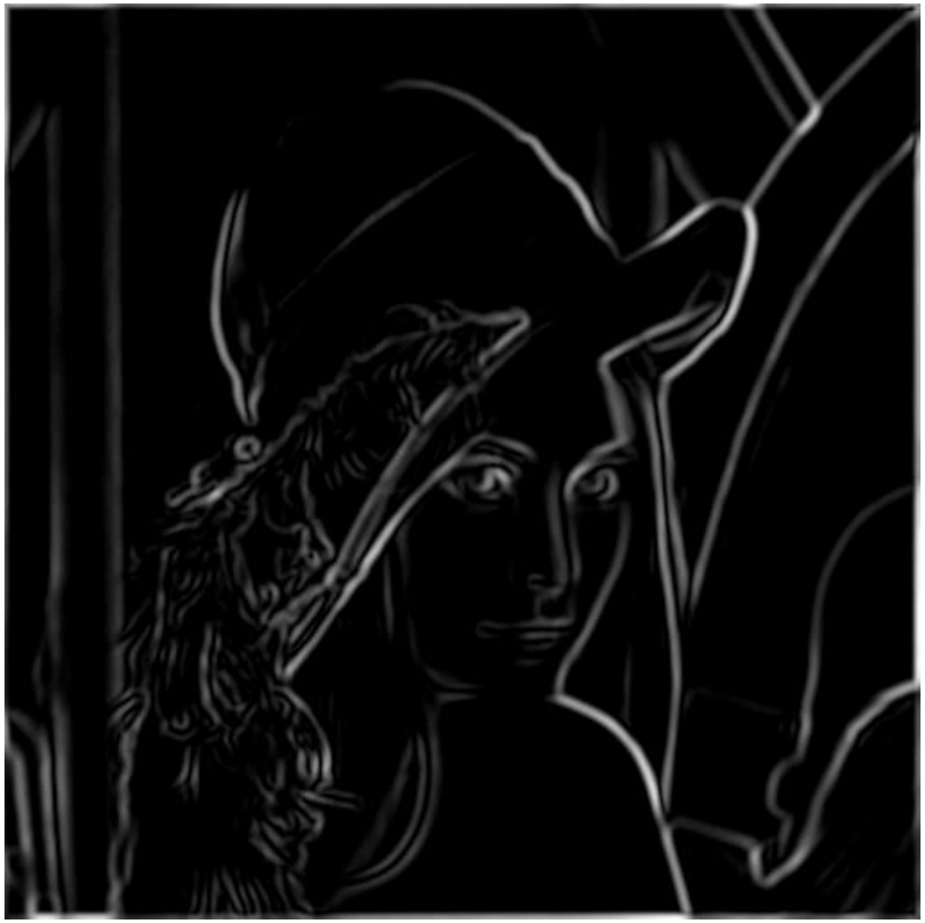


- Original image (Lena)

The Canny edge detector

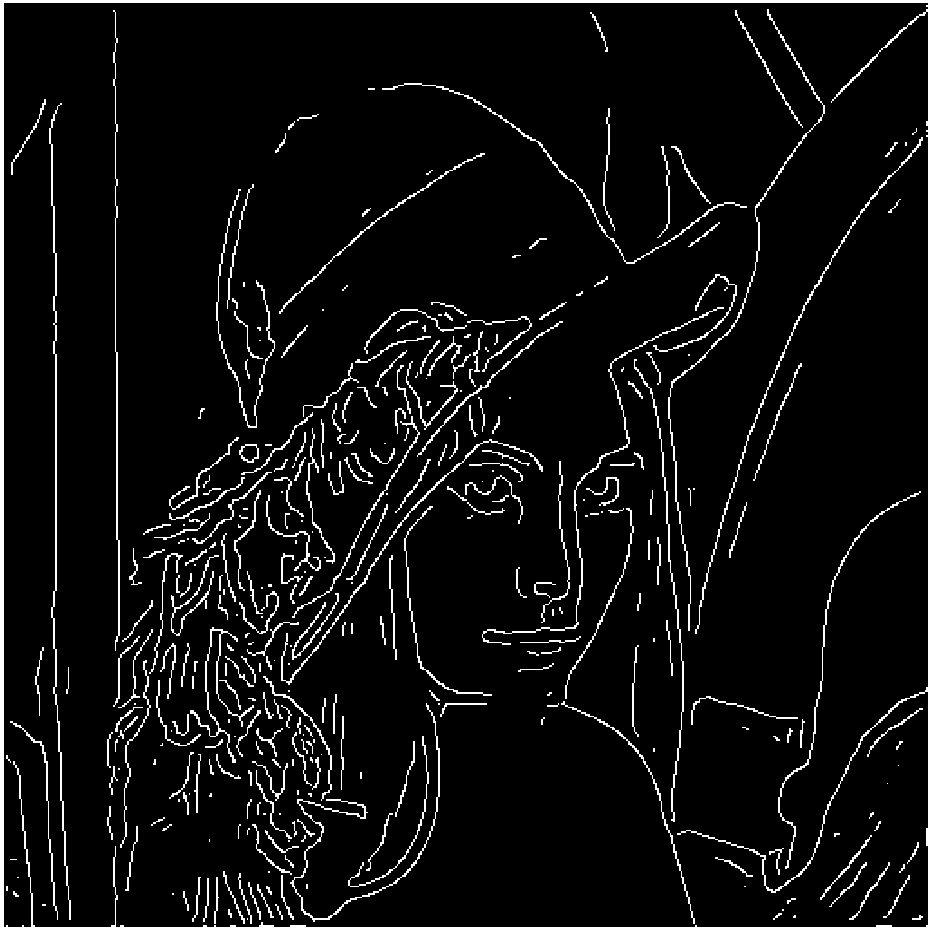


The Canny edge detector

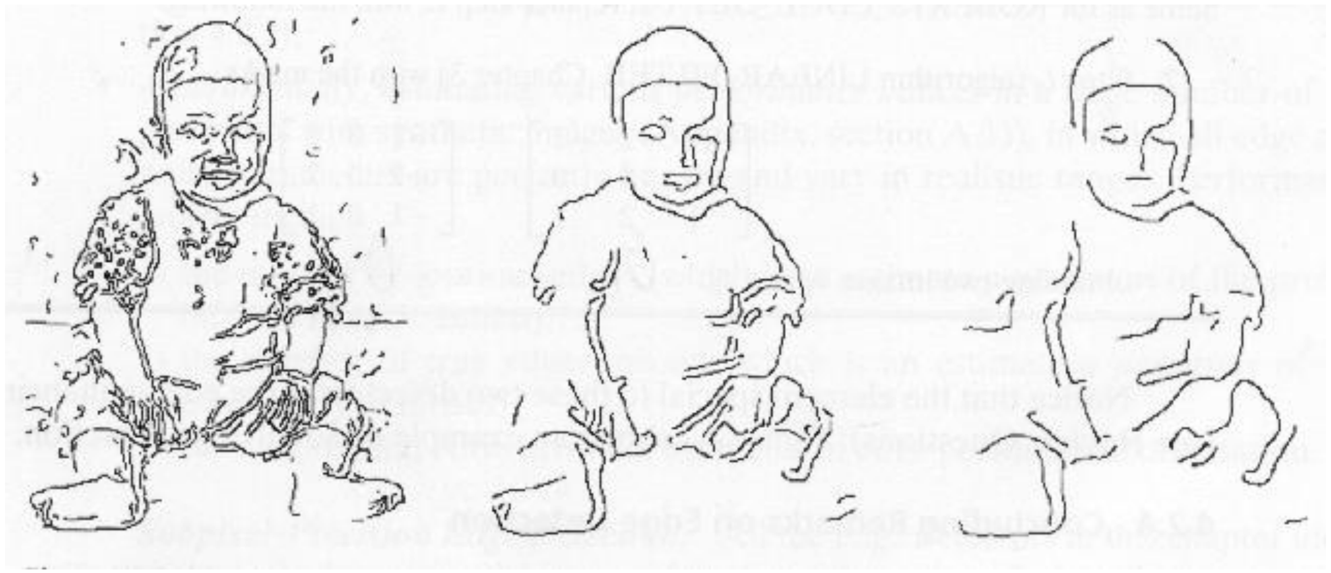


- Thresholding

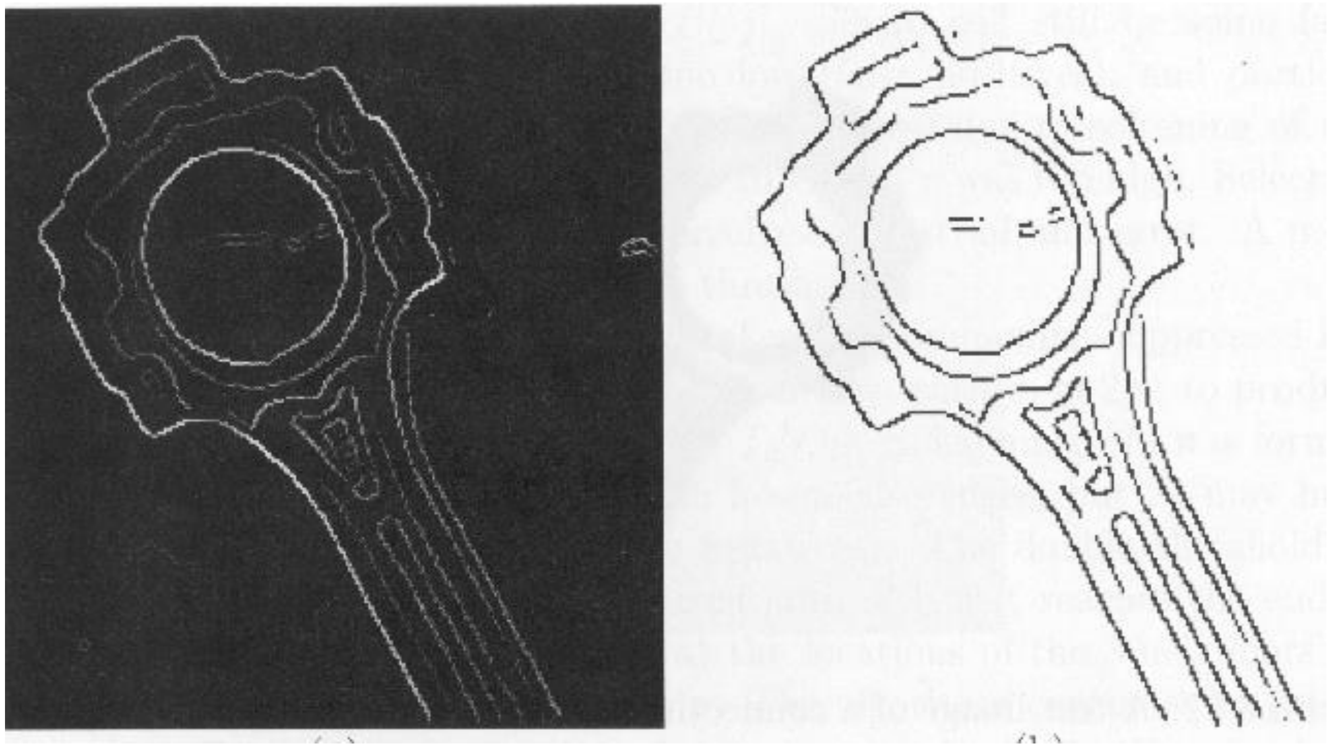
The Canny edge detector



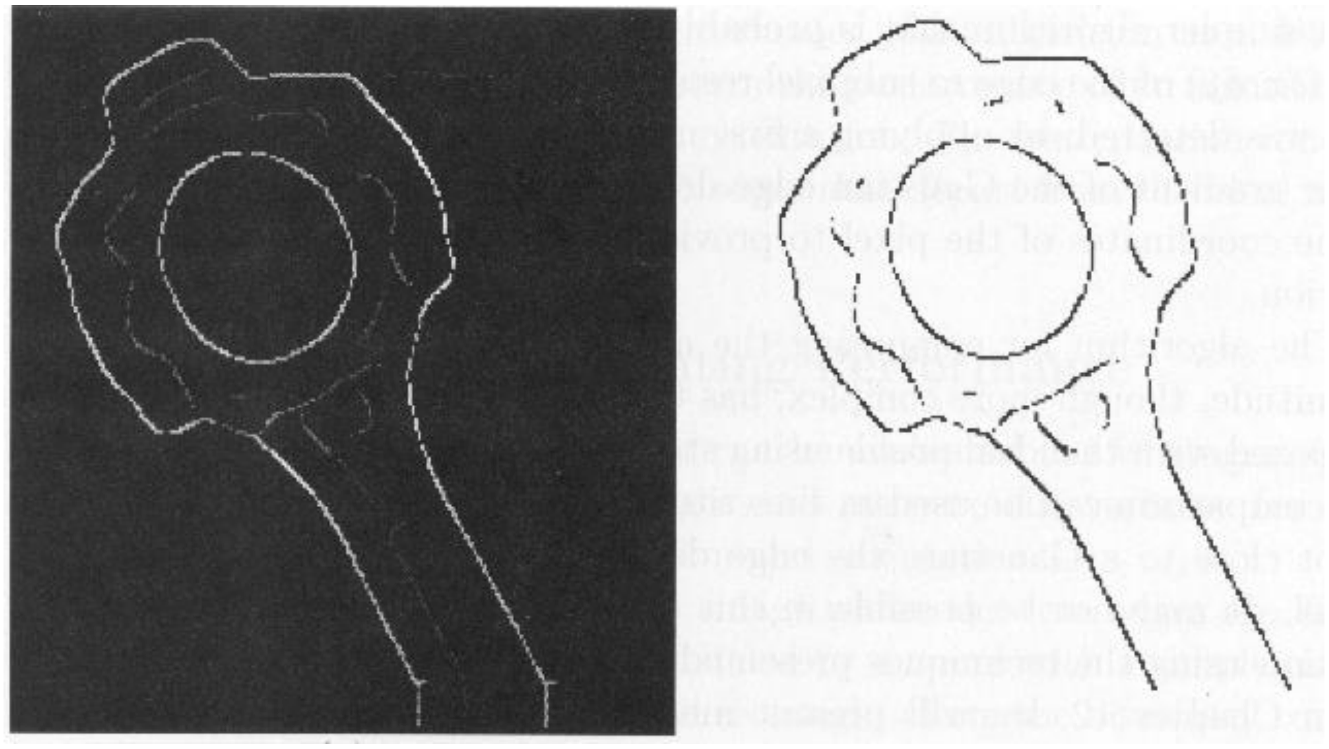
- Thinning (non-maximum suppression)



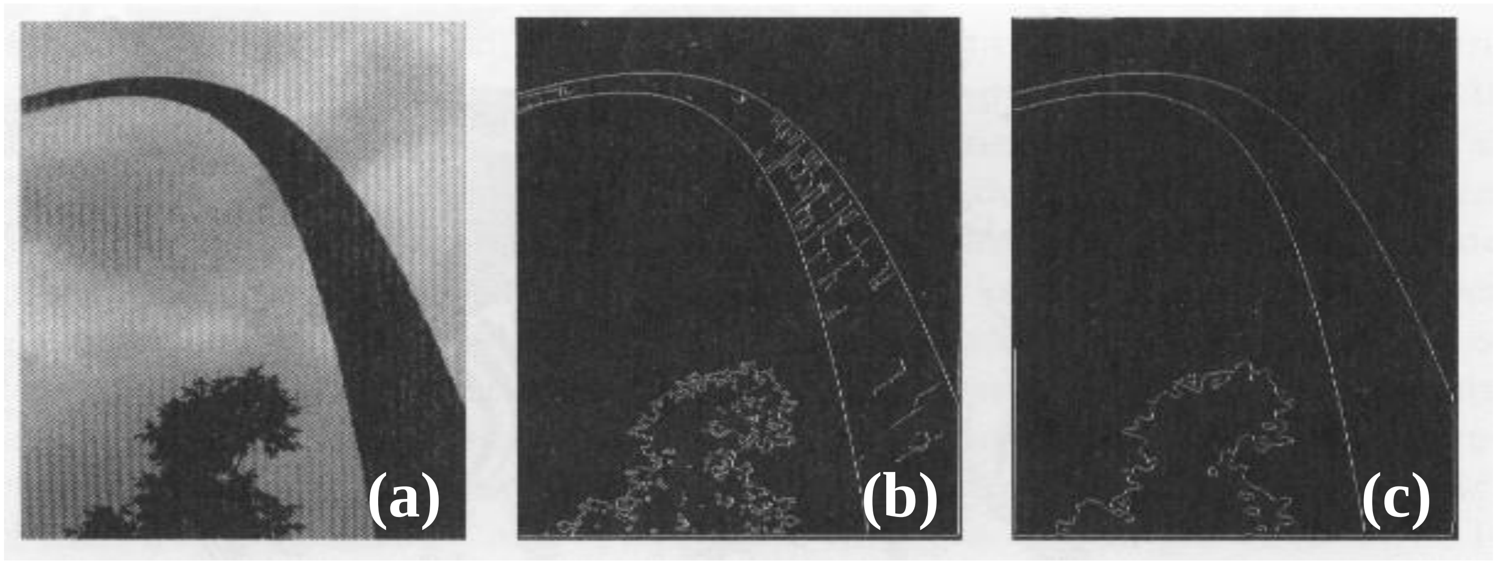
Canny - left: σ =1, middle: σ =2, right: σ =3



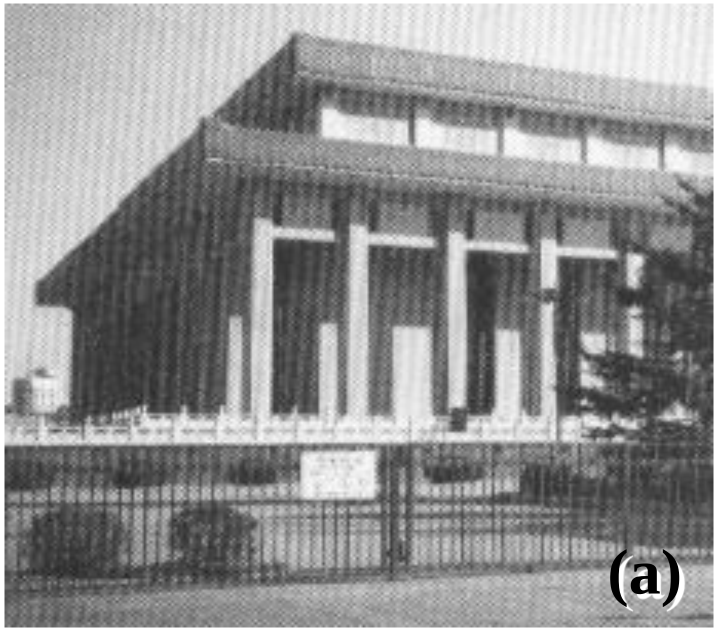
Canny - 7x7 Gaussian, more details



Canny - 31x31 Gaussian, less details



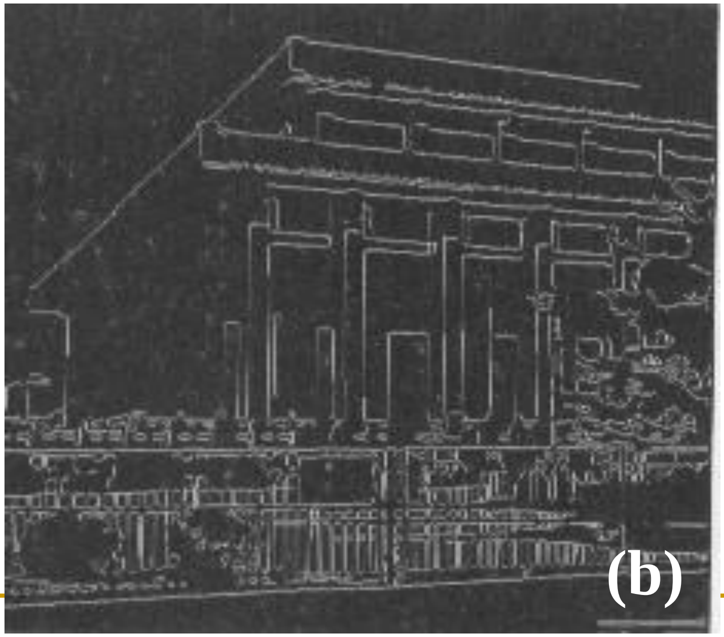
Example: (a) Image of the great arch in St. Louis; (b) results of Canny operator with $\sigma = 1$; (c) results of Canny operator with $\sigma = 4$;

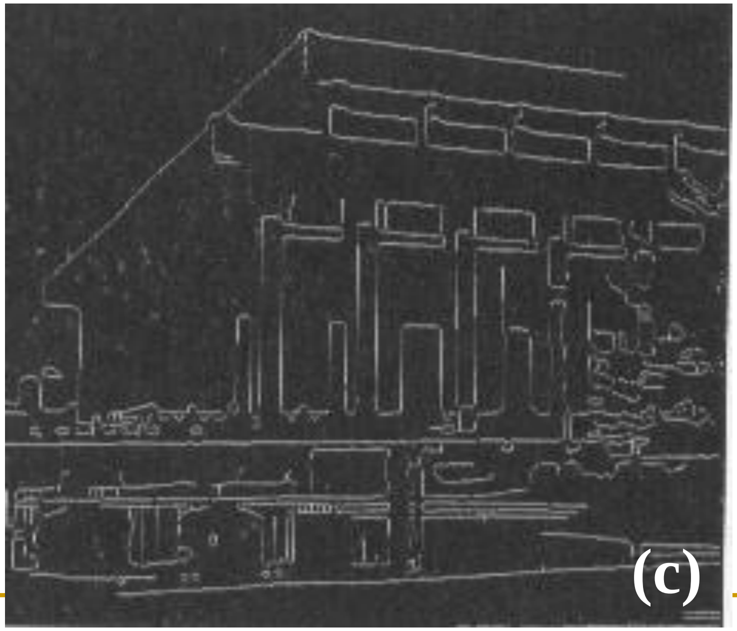


Examples

(a) Image of Mao’s Tomb.
(b) Result of applying Canny operator with σ =1.
(c) Result of σ =2.

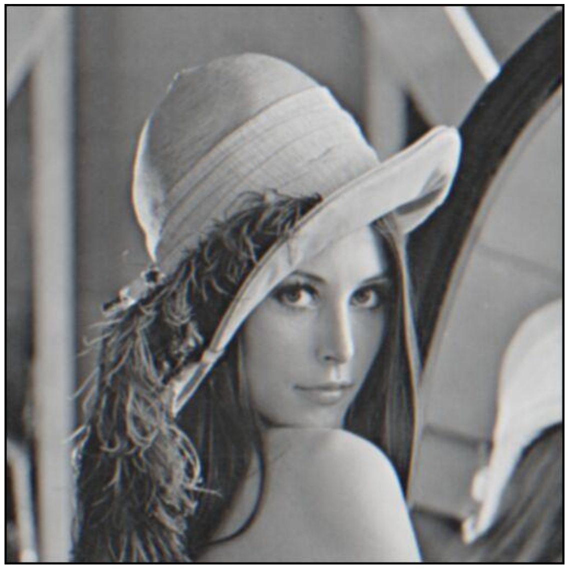
Some objects are detected very well, so are some shadows.

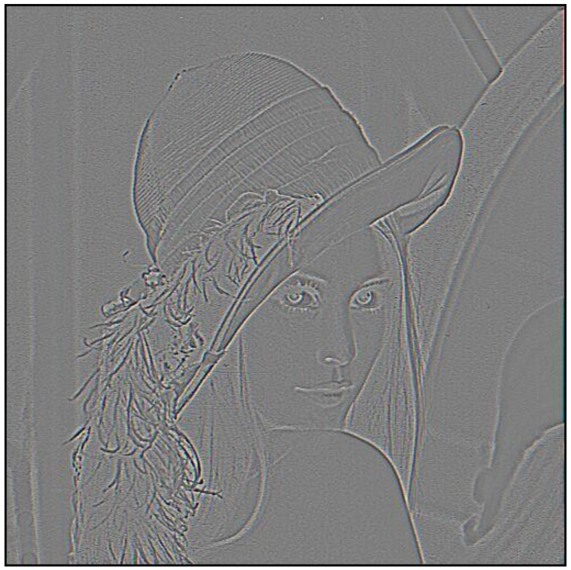


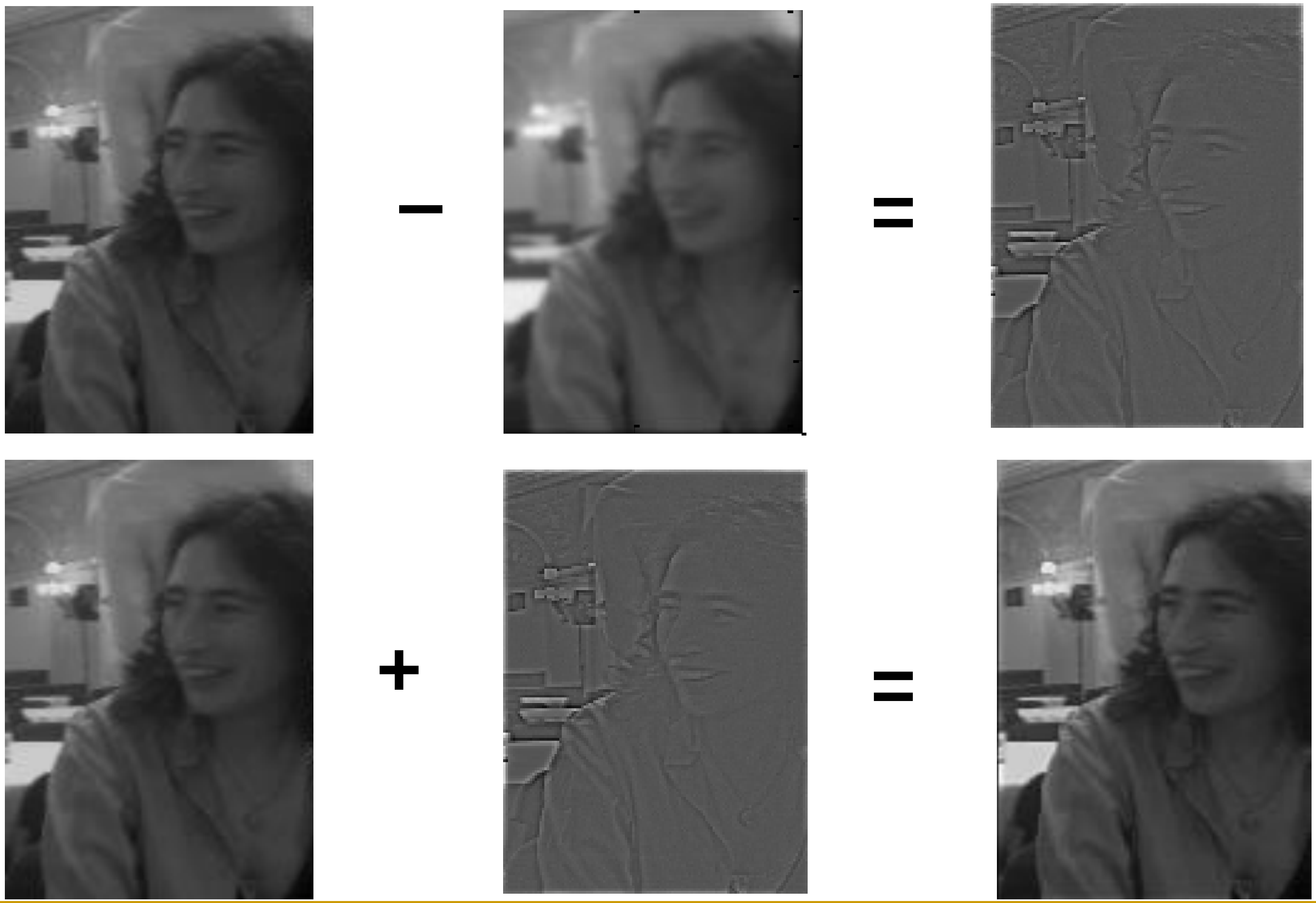


Edge detection by subtraction

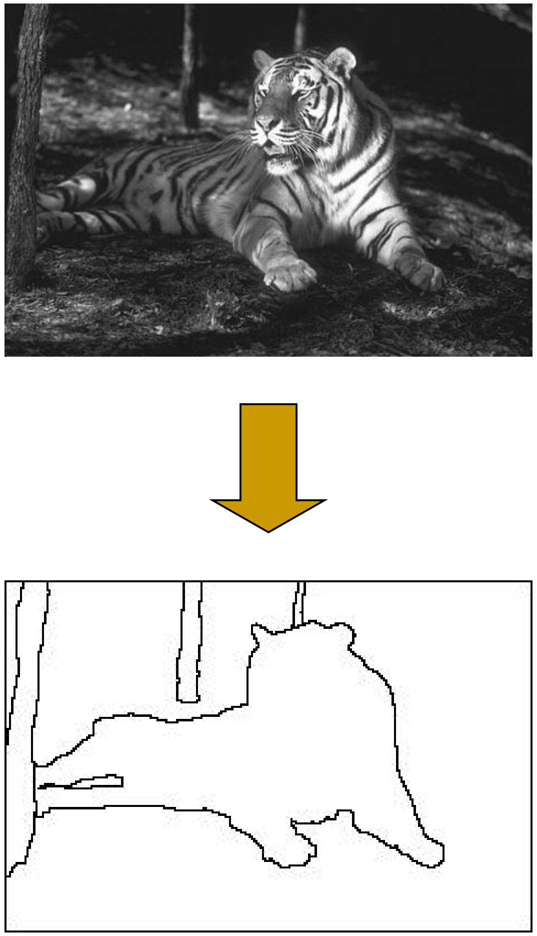
original


smoothed (5x5 Gaussian)


Original -smoothed

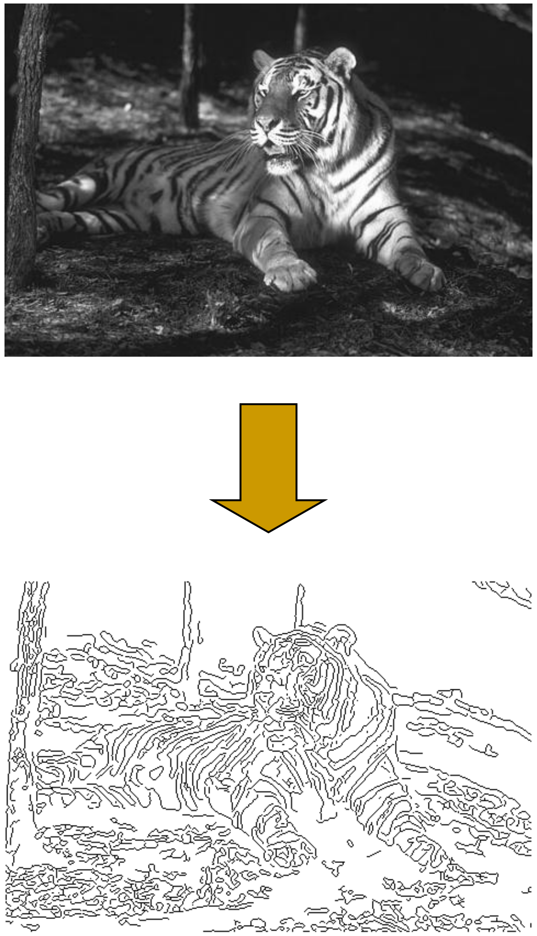


Why edges?

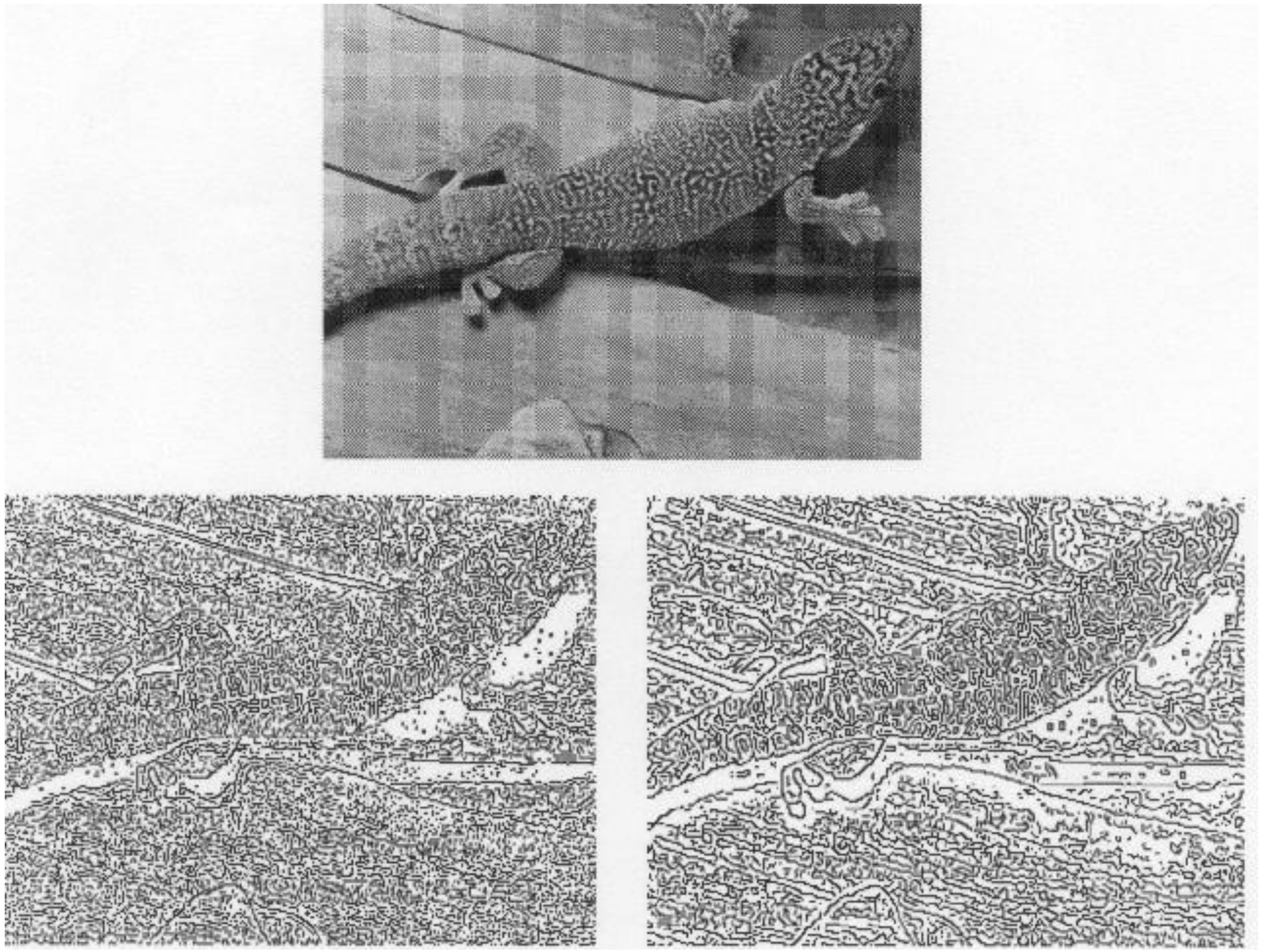


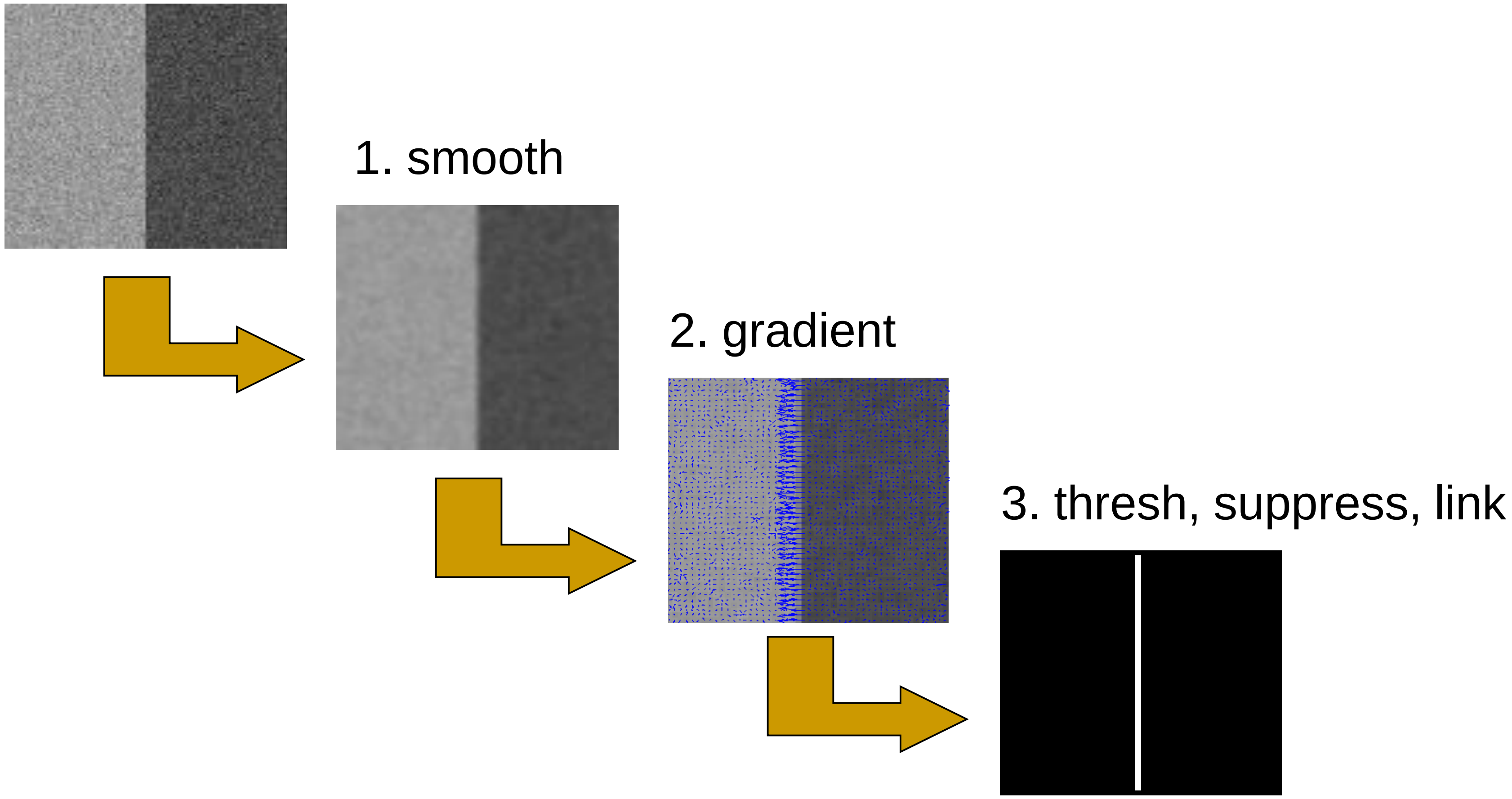
- Reduce dimensionality of data
- Preserve content information
- Useful in applications such as:
  - object detection
  - structure from motion
  - tracking

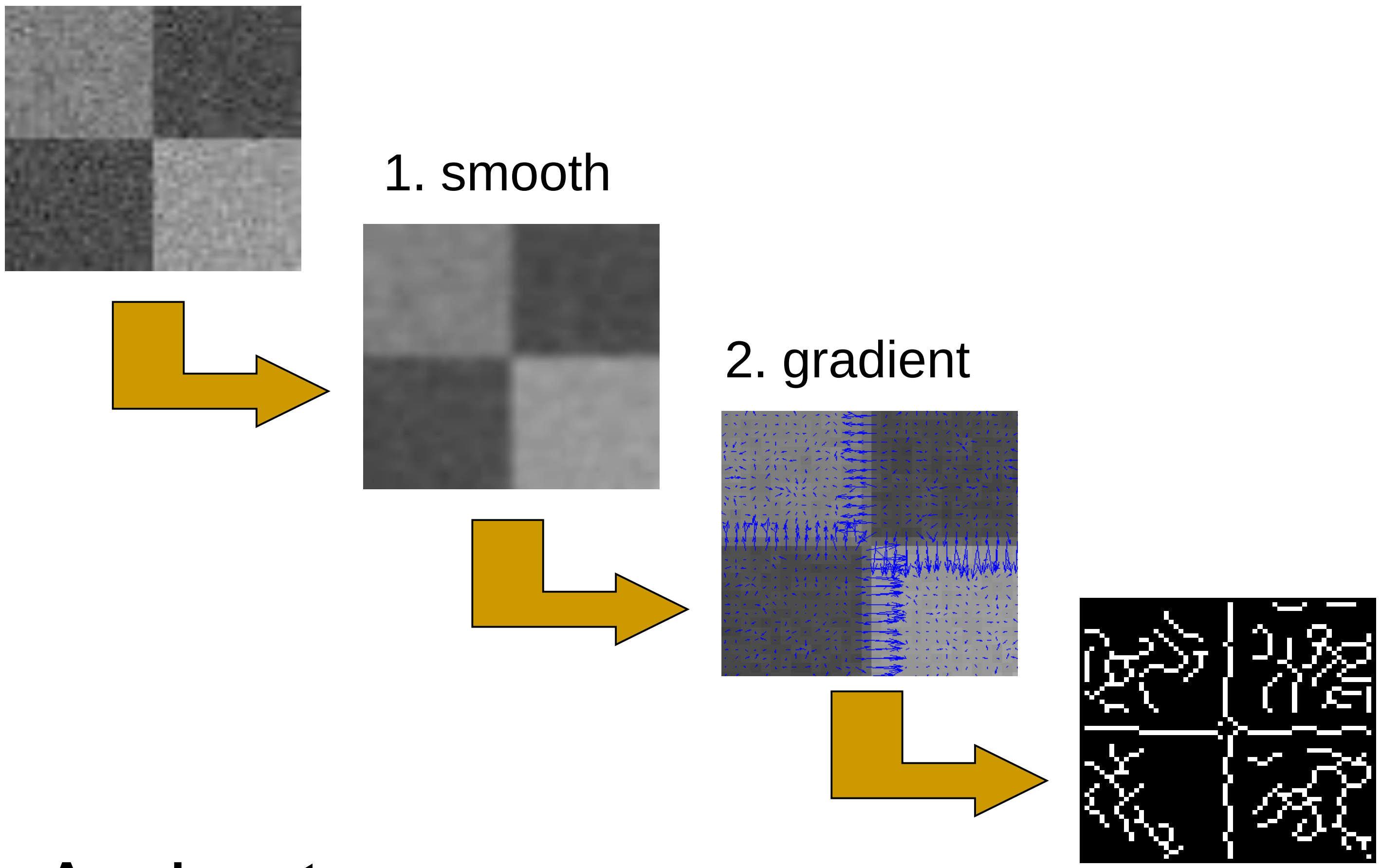
Why not edges?



- But, not that useful, **why**?
- Difficulties:
  1. Modeling assumptions
  2. Parameters
  3. Multiple sources of information (brightness, color, texture, …)
  4. Real world conditions




1. smooth
2. gradient
3. thresh, suppress, link



And yet…

Canny difficulties

- Modeling assumptions

  Step edges, junctions, etc.

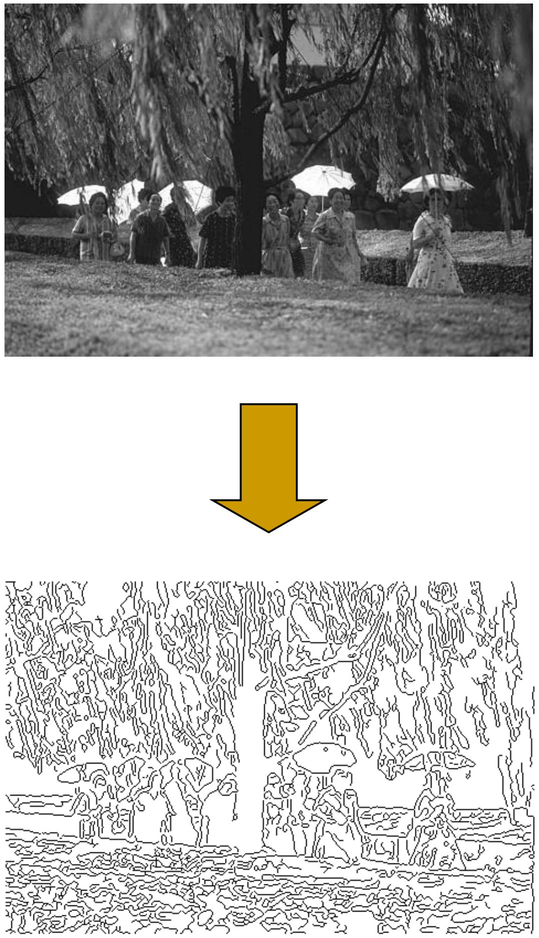
- Parameters

  Gaussian Scales, threshold, etc.

- Multiple sources of information

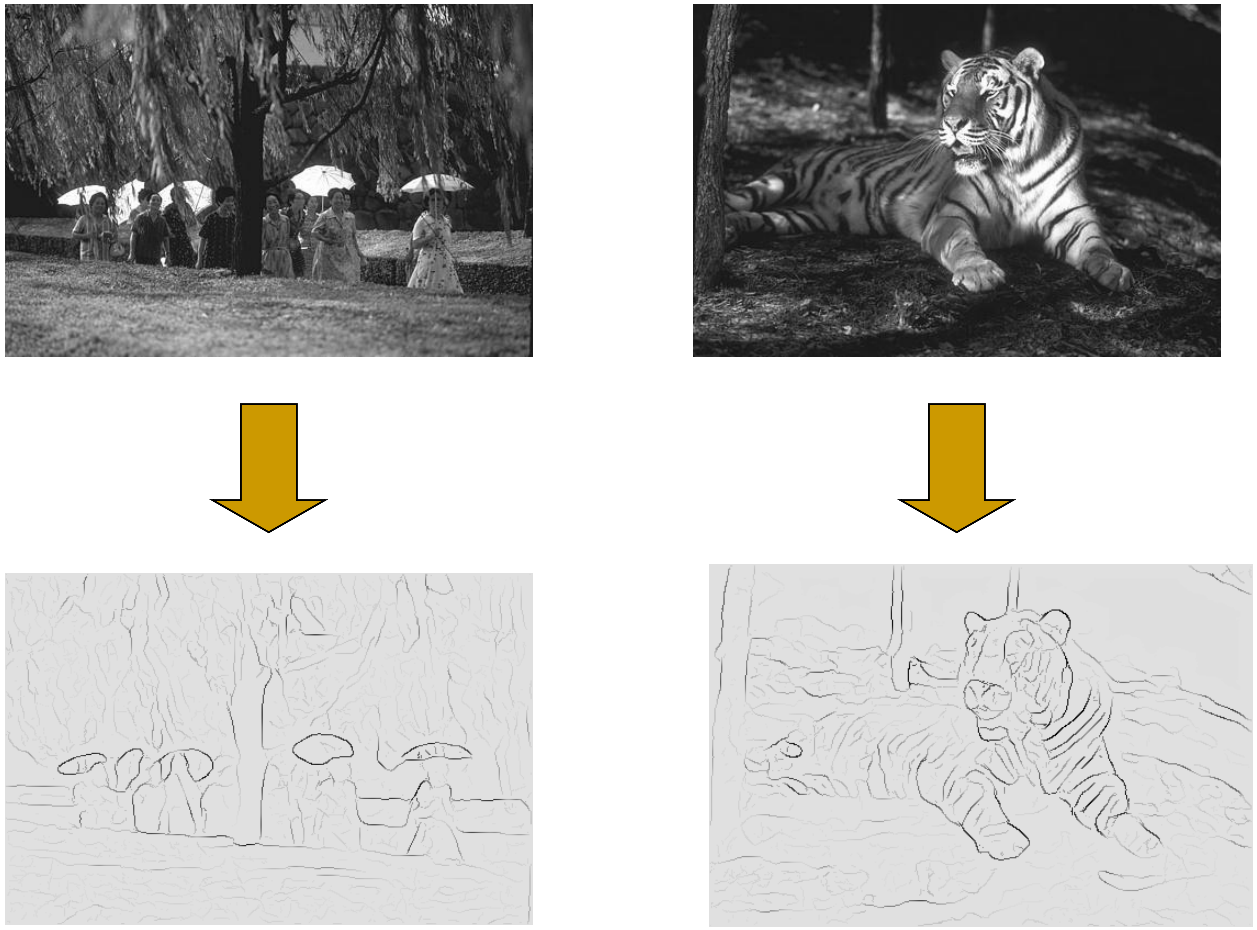
  Only handles brightness

- Real world conditions



Modern methods

- Modeling assumptions
  Complex models
- Parameters
  Many, may use learning to help tune
- Multiple sources of information
  Typically brightness, color, and texture cues
- Real world conditions
  Aimed at real images



Learning

- Modeling assumptions
  minimal
- Parameters
  none
- Multiple sources of information
  automatically incorporated
- Real world conditions
  training data