Digital Image Processing

Image Enhancement - Filtering

Derivative

- Derivative is defined as a rate of change.

Discrete Derivative Finite Distance

$$\frac{df}{dx} = f(x) - f(x-1) = f'(x)$$ Backward difference

$$\frac{df}{dx} = f(x) - f(x+1) = f'(x)$$ Forward difference

$$\frac{df}{dx} = f(x+1) - f(x-1) = f'(x)$$ Central difference

Example

$$
\begin{array}{lllllllll}
f(x) = & 10 & 15 & 10 & 10 & 25 & 20 & 20 & 20 \\
f'(x) = & 0 & 5 & -5 & 0 & 15 & -5 & 0 & 0 \\
f''(x) = & 0 & 5 & -10 & 5 & 15 & 20 & 5 & 0
\end{array}
$$

Derivative Masks

Backward difference	[-1 1]
Forward difference	[1 -1]
Central difference	[-1 0 1]

Derivatives in 2-dimension

Given function $f(x,y)$

Gradient vector $\nabla f(x,y) = \begin{bmatrix} \dfrac{\partial f(x,y)}{\partial x} \\ \dfrac{\partial f(x,y)}{\partial y} \end{bmatrix} = \begin{bmatrix} f_x \\ f_y \end{bmatrix}$

Gradient magnitude $|\nabla f(x,y)| = \sqrt{f_x^2 + f_y^2}$

Gradient direction $\theta = \tan^{-1} \dfrac{f_x}{f_y}$

Derivatives of Images

Derivative masks $$f_x \Rightarrow \frac{1}{3}\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \qquad f_y \Rightarrow \frac{1}{3}\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

$$I = \begin{bmatrix} 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \end{bmatrix} \qquad I_x = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 10 & 10 & 0 & 0 \\ 0 & 10 & 10 & 0 & 0 \\ 0 & 10 & 10 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Derivatives of Images

$$I = \begin{bmatrix} 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \end{bmatrix} \qquad I_y = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Correlation

$$f \otimes h = \sum_{k} \sum_{l} f(k,l)\,h(i+k, j+l)$$

f = Image

f = Kernel

$$\underset{f}{\begin{bmatrix} f_1 & f_2 & f_3 \\ f_4 & f_5 & f_6 \\ f_7 & f_8 & f_9 \end{bmatrix}} \otimes \underset{h}{\begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix}} \rightarrow$$

$$\begin{aligned} f * h = {} & f_1 h_1 + f_2 h_2 + f_3 h_3 \\ & + f_4 h_4 + f_5 h_5 + f_6 h_6 \\ & + f_7 h_7 + f_8 h_8 + f_9 h_9 \end{aligned}$$

Convolution

$$f * h = \sum_{k} \sum_{l} f(k,l)\,h(i-k, j-l)$$

$f = \text{Image}$

$h = \text{Kernel}$

$$h = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix} \xrightarrow{X-flip} \begin{bmatrix} h_7 & h_8 & h_9 \\ h_4 & h_5 & h_6 \\ h_1 & h_2 & h_3 \end{bmatrix} \xrightarrow{Y-flip} \begin{bmatrix} h_9 & h_8 & h_7 \\ h_6 & h_5 & h_4 \\ h_3 & h_2 & h_1 \end{bmatrix}$$

$$f = \begin{bmatrix} f_1 & f_2 & f_3 \\ f_4 & f_5 & f_6 \\ f_7 & f_8 & f_9 \end{bmatrix} \otimes \begin{bmatrix} h_9 & h_8 & h_7 \\ h_6 & h_5 & h_4 \\ h_3 & h_2 & h_1 \end{bmatrix} \longrightarrow$$

$$\begin{aligned} f * h = {} & f_1 h_9 + f_2 h_8 + f_3 h_7 \\ & + f_4 h_6 + f_5 h_5 + f_6 h_4 \\ & + f_7 h_3 + f_8 h_2 + f_9 h_1 \end{aligned}$$

Averages

- Mean

$$I = \frac{I_1 + I_2 + \ldots I_n}{n} = \frac{\sum_{i=1}^{n} I_i}{n}$$

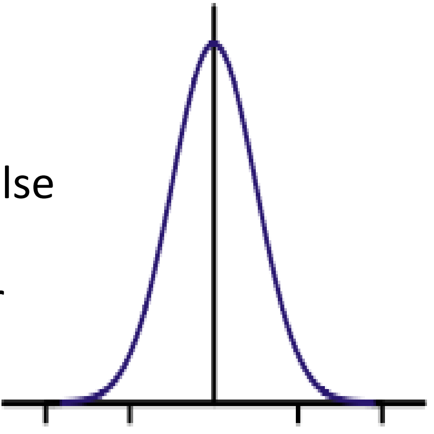
- Weighted mean

$$I = \frac{w_1 I_1 + w_2 I_2 + \ldots + w_n I_n}{n} = \frac{\sum_{i=1}^{n} w_i I_i}{n}$$

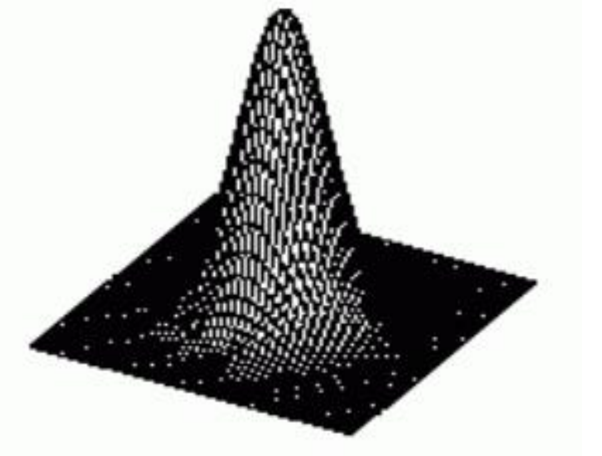
Gaussian Filtering

- The Gaussian smoothing operator is a 2-D convolution operator that is used to `blur' images and remove detail and noise. In this sense it is similar to the mean filter, but it uses a different kernel that represents the shape of a Gaussian (`bell-shaped') hump.

Shape of Impulse Response for Gaussian Filter



2D Gaussian



Scale of Gaussian

- As sigma increases, more pixels are involved in averaging
- As sigma increase, image is more blurred
- As sigma increase, noise is more effectively suppressed

Gaussian Filter

- Gaussian smoothing

$$\overbrace{S}^{\text{smoothed image}} = \overbrace{g}^{\text{Gaussian filter}} * \overbrace{I}^{\text{image}} \qquad g = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

$$\frac{1}{2\pi\sigma^2}.e^{\frac{-(x^2+y^2)}{2\sigma^2}}$$

Consider sigma = 0.6 and kernel size = 3x3

$$\frac{1}{2\pi\sigma^2} = \frac{1}{2\times 3.14\times 0.6\times 0.6} = \frac{1}{2.2619}$$

Gaussian Smoothing

Kernel width; X= 3, Kernel Height; Y = 3

$$X = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \text{ and } Y = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

$$e^{\frac{-(x^2+y^2)}{2\sigma^2}} = \begin{bmatrix} -2.7778 & -1.3889 & -2.7778 \\ -1.3889 & 0 & -1.3889 \\ -2.7778 & -1.3889 & -2.7778 \end{bmatrix}$$

The Gaussian kernel's center part (Here 0.4421) has the highest value and intensity of other pixels decrease as the distance from the center part increases.

Gaussian Smoothing

The Gaussian kernel takes the form as:

$$\begin{bmatrix} 0.0275 & 0.1102 & 0.0275 \\ 0.1102 & 0.4421 & 0.1102 \\ 0.0275 & 0.1102 & 0.0275 \end{bmatrix}$$

Convolve the kernel with the region in the given image

$$\begin{bmatrix} 72 & 68 & 88 & 159 \\ 69 & 66 & 87 & 162 \\ 70 & 66 & 83 & 161 \\ 70 & 66 & 78 & 154 \end{bmatrix}$$

Gaussian Smoothing

Performing Convolution:

$$\begin{bmatrix} 68 & 88 & 159 \\ 66 & 87 & 162 \\ 66 & 83 & 161 \end{bmatrix} .* \begin{bmatrix} 0.0275 & 0.1102 & 0.0275 \\ 0.1102 & 0.4421 & 0.1102 \\ 0.0275 & 0.1102 & 0.0275 \end{bmatrix} = \begin{bmatrix} 1.8692 & 9.7009 & 4.3706 \\ 7.2757 & 38.4624 & 17.8585 \\ 1.8142 & 9.1497 & 4.4256 \end{bmatrix}$$

On convolution of the local region and the Gaussian kernel gives the highest intensity value to the center part of the local region **(38.4624)** and the remaining pixels have less intensity as the distance from the center increases.
Sum up the result and store it in the current pixel location **(Intensity = 94.9269)** of the image.

Gaussian Smoothing

$$\begin{bmatrix} [\,] & [\,] & [\,] & [\,] \\ [\,] & [\,] & 94.9269 & [\,] \\ [\,] & [\,] & [\,] & [\,] \\ [\,] & [\,] & [\,] & [\,] \end{bmatrix}$$

Performing calculations for each pixel, the resultant image is:

$$\begin{bmatrix} 48.7478 & 59.2645 & 79.7865 & 100.2444 \\ 57.1176 & 69.7512 & 94.9296 & 121.1870 \\ 57.1740 & 68.9526 & 92.2220 & 119.6981 \\ 47.7534 & 59.9750 & 74.1254 & 96.7113 \end{bmatrix}$$

Edge Detection

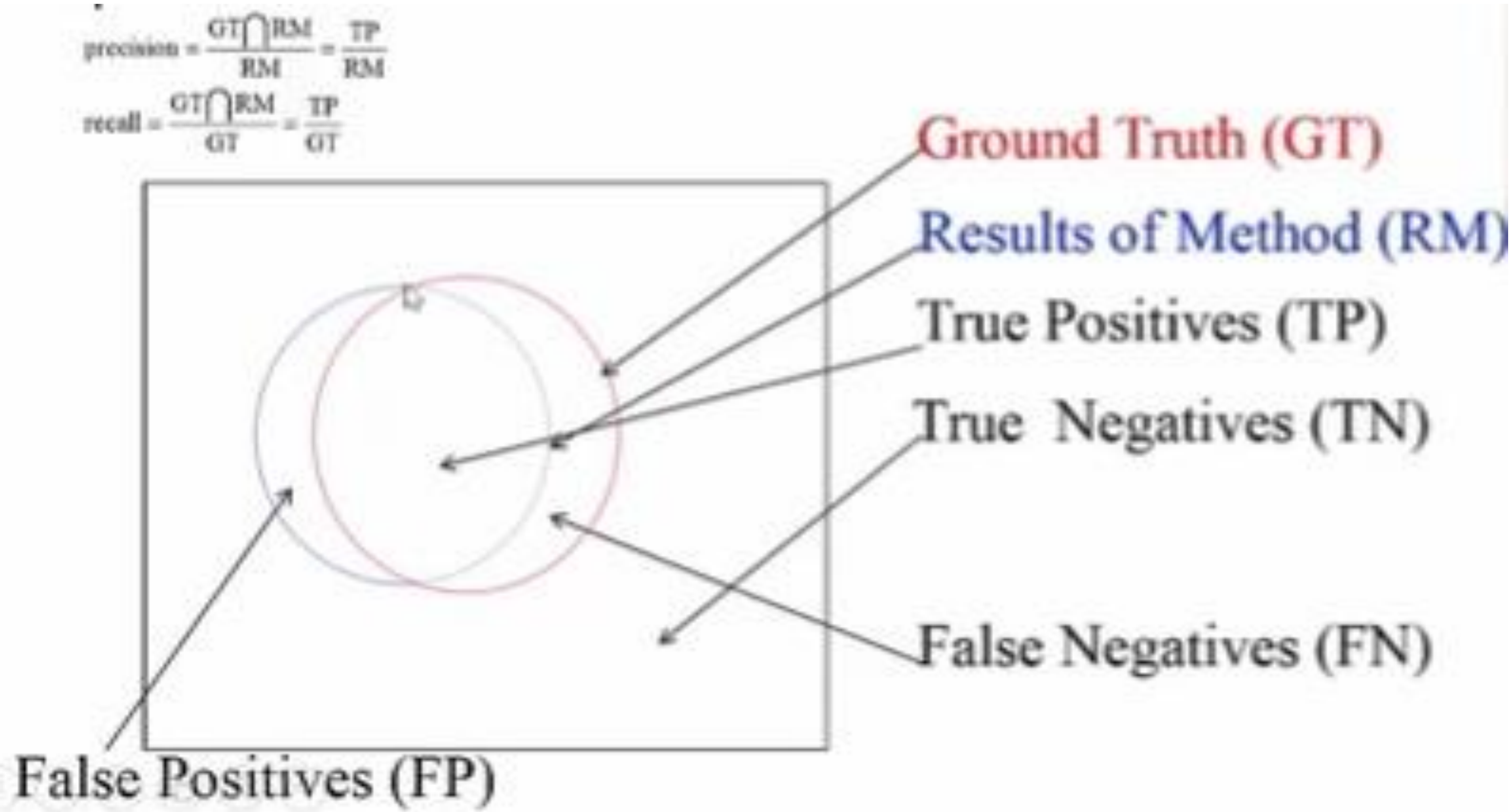
- **Edge detection** is an image processing technique for finding the boundaries of objects within images. It works by **detecting** discontinuities in brightness.

- **Edge detection** is used for image segmentation and data extraction in areas such as image processing, computer vision, and machine vision.

Edge Detectors

- Gradient Operators
  - Robert Cross
  - Prewit
  - Sobel
- Gradient of Gaussian (Canny)
- Laplacian of Gaussian (LoG) (Marr-Hildreth)

Computing Efficiency of Edge Detection Algorithms



Spatial Filtering for Image Sharpening

Background: to highlight fine detail in an image or to enhance blurred detail

Applications: electronic printing, medical imaging, industrial inspection, autonomous target detection (smart weapons)......

Foundation (Blurring vs Sharpening):

- Blurring/smoothing is performed by spatial averaging (equivalent to integration)
- Sharpening is performed by noting only the gray level changes in the image that is the differentiation

Spatial Filtering for Image Sharpening

Operation of Image Differentiation

- Enhance edges and discontinuities (magnitude of output gray level >>0)
- De-emphasize areas with slowly varying gray-level values (output gray level: 0)

Mathematical Basis of Filtering for Image Sharpening

- First-order and second-order derivatives
- Gradients
- Implementation by mask filtering

Derivatives

First Order Derivative

- A basic definition of the first-order derivative of a one-dimensional function f(x) is the difference

$$\frac{\partial f}{\partial x} = f(x+1) - f(x)$$

Second Order Derivative

- Ssimilarly, we define the second-order derivative of a one-dimensional function f(x) is the difference

$$\frac{\partial^2 f}{\partial x^2} = f(x+1) + f(x-1) - 2f(x)$$

First and Second Order Derivatives

$$\frac{\partial f}{\partial x} = f(x+1) - f(x)$$

$f(x)$	$f(x+1)$

Position for the output pixel

$$\frac{\partial^2 f}{\partial x^2} = f'(x+1) - f'(x)$$

$$= [f(x+2) - f(x+1)] - [f(x+1) - f(x)]$$

$$= f(x+2) - 2f(x+1) + f(x)$$

$f(x)$	$f(x+1)$	$f(x+2)$

$$\frac{\partial^2 f}{\partial x^2} = f(x+1) + f(x-1) - 2f(x)$$

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

Comparison between f'' and f'

- f' generally produces thicker edges in an image
- f'' has a stronger response to fine detail
- f' generally has a stronger response to a gray-level step
- f'' produces a double response at step changes in gray level
- For image enhancement, f'' is generally better suited than f'
- Major application of f' is for edge extraction; f' used together with f'' results in impressive enhancement effect

Matlab Functions for Filters

- special: Create predefined 2-D filters
  - *H = fspecial(TYPE)* creates a two-dimensional filter H of the specified type. Possible values for TYPE are:
  - 'average' averaging filter;
  - 'gaussian' Gaussian lowpass filter
  - 'laplacian' filter approximating the 2-D Laplacian operator
  - 'log' Laplacian of Gaussian filter
  - 'prewitt' Prewitt horizontal edge-emphasizing filter
  - 'sobel' Sobel horizontal edge-emphasizing filter
  - Example: H=fspecial('gaussian',7,1) creates a 7x7 Gaussian filter with variance 1.

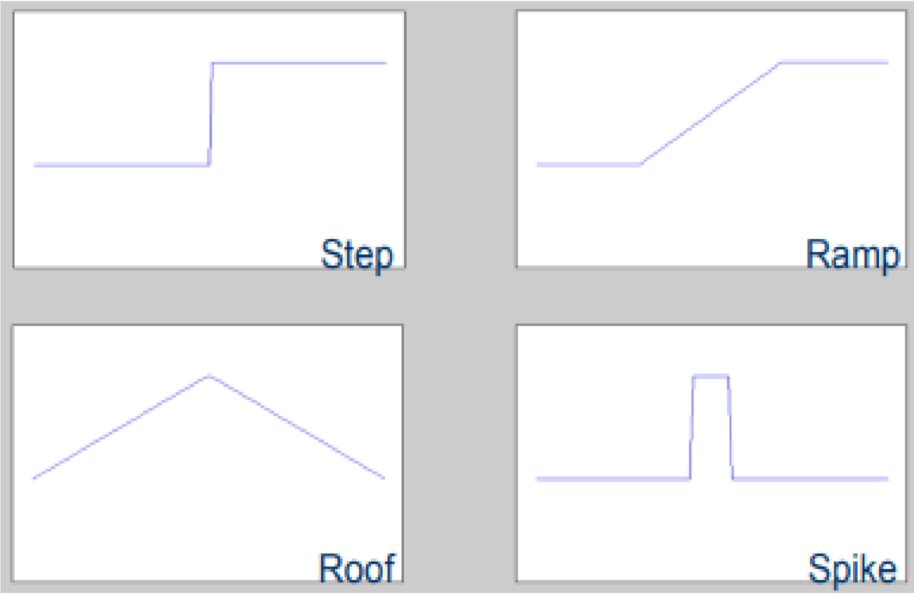
Image Gradient

- An **image gradient** is a directional change in the intensity or color in an **image**.
- The **gradient** of the **image** is one of the fundamental building blocks in **image processing**.
- For example the Canny edge detector uses **image gradient** for edge detection.

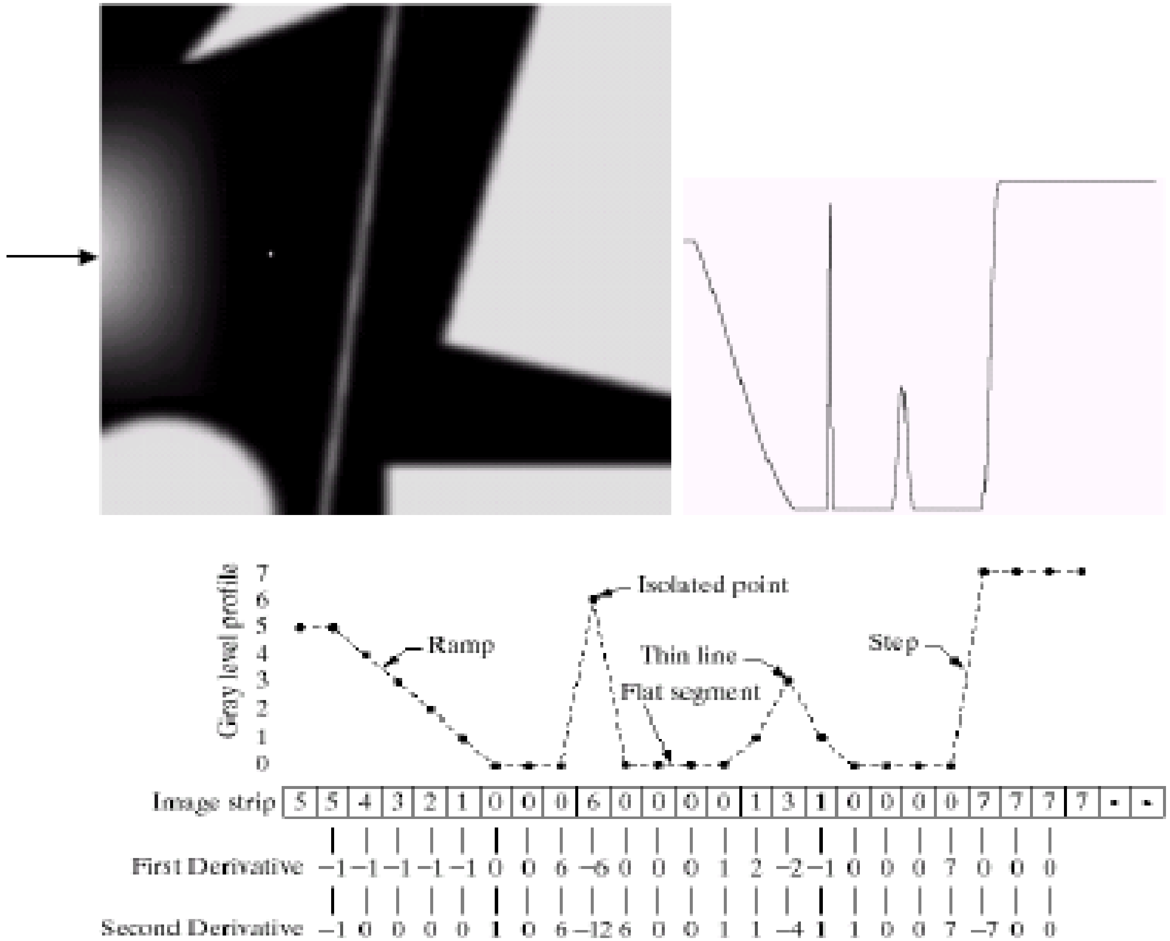
Edge Detection

- What is an edge?
  - An abrupt or sudden change in intensity value
  - Discontinuity of intensities in the image

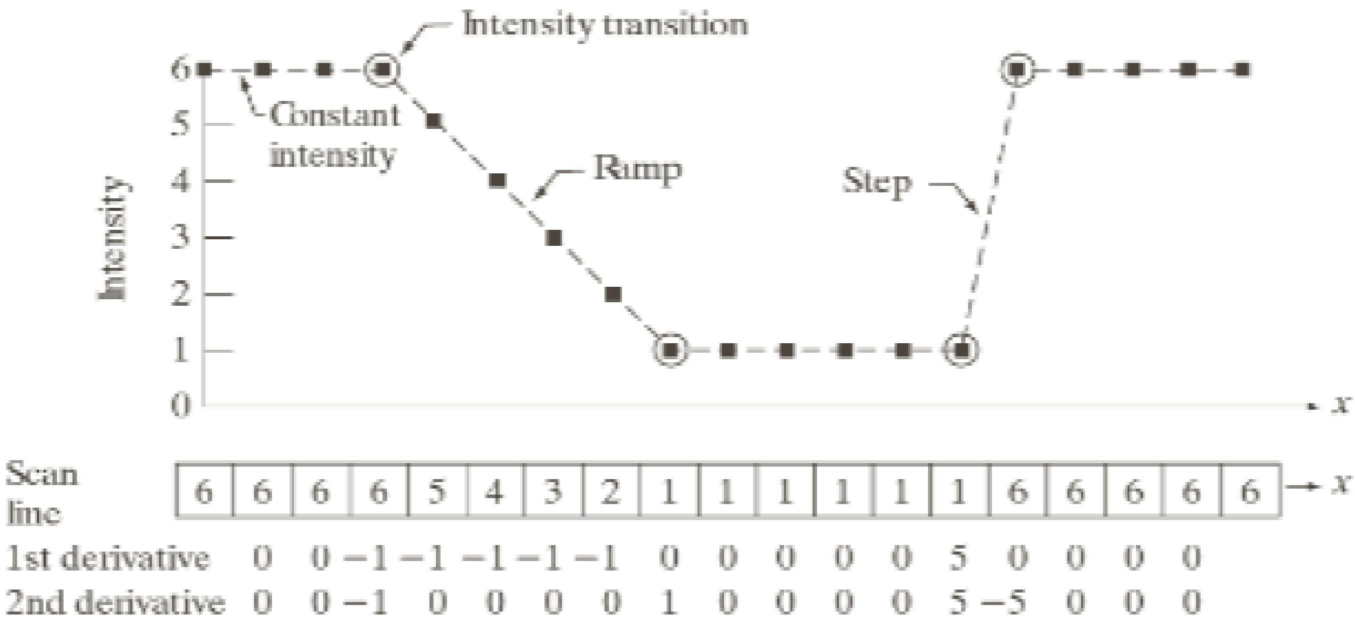
- Edge Models
  - Step
  - Roof
  - Ramp
  - Spike



Example for Discrete Derivatives



Example for Discrete Derivatives



1^{st} and 2^{nd} Order Derivatives

First Order Derivative

- Must be zero in area of constant gray levels
- Non zero along the ramps
- Non zero at the onset of the gray level step or ramp

Second Order Derivative

- Zero in flat areas
- Zero along the ramps of constant slope
- Non zero at the onset and end of the gray level step or ramp

Laplacian for Image Enhancement

Laplacian operator $$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

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

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

$$\nabla^2 f = f(x+1, y) + f(x-1, y) + f(x, y+1) + f(x, y-1) - 4f(x, y)$$

y

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

x

0	1	0
1	-4	1
0	1	0

Laplacian for Image Enhancement

(a)

0	1	0
1	−4	1
0	1	0

(b)

1	1	1
1	−8	1
1	1	1

(c)

0	−1	0
−1	4	−1
0	−1	0

(d)

−1	−1	−1
−1	8	−1
−1	−1	−1

a	b
c	d

FIGURE 3.39 (a) Filter mask used to implement the digital Laplacian, as defined in Eq. (3.7-4). (b) Mask used to implement an extension of this equation that includes the diagonal neighbors. (c) and (d) Two other implementations of the Laplacian.

Laplacian for Image Enhancement

To obtain the enhanced image

$$g(x,y)=\begin{cases} f(x,y)-\nabla^2 f(x,y), w_5<0 \\ f(x,y)+\nabla^2 f(x,y), w_5>0 \end{cases}$$

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

In this way, background tonality can be perfectly preserved
while details are enhanced

Laplacian for Image Enhancement (Example)

a b
c d

FIGURE 3.40
(a) Image of the North Pole of the moon.
(b) Laplacian-filtered image.
(c) Laplacian image scaled for display purposes.
(d) Image enhanced by using Eq. (3.7-5).
(Original image courtesy of NASA.)

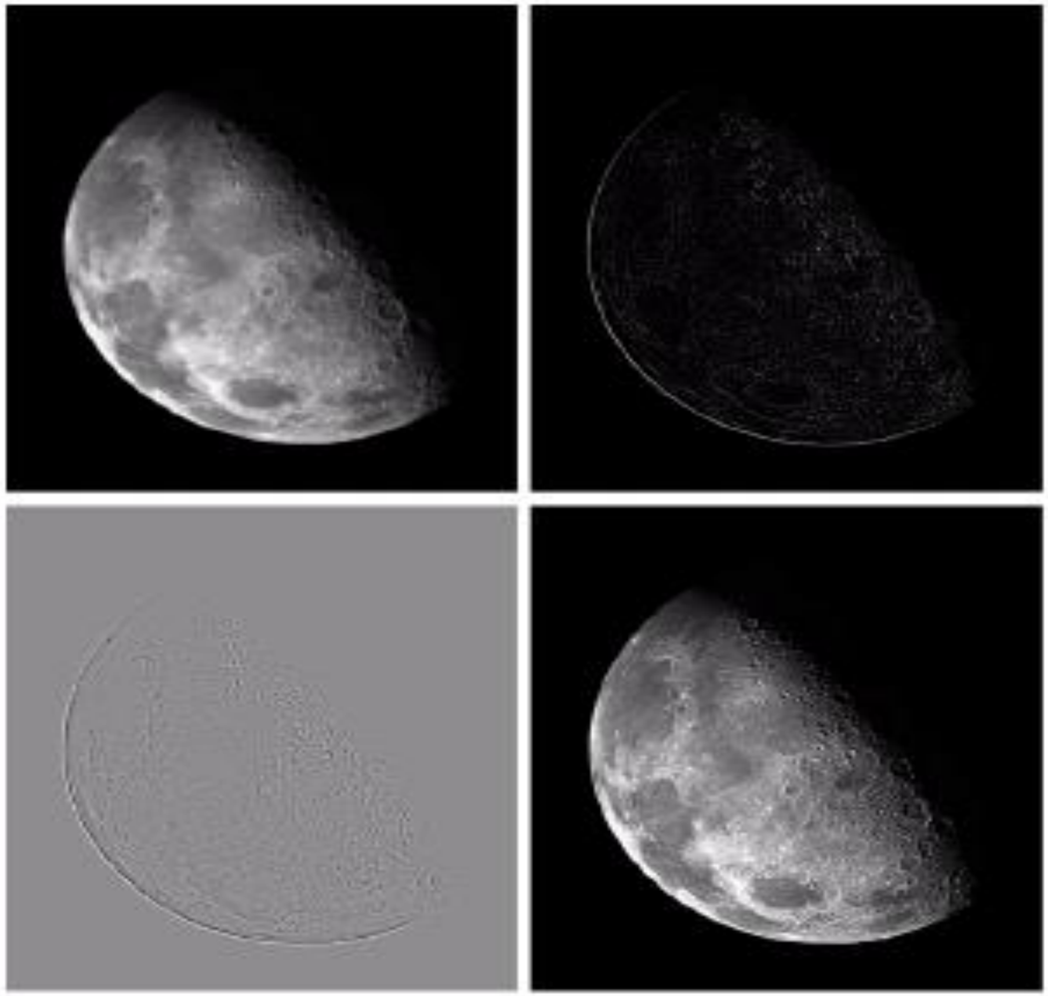


Image Sharpening Based on Unsharp Masking

Subtract the blurred image from the original image

$$f_s(x,y) = f(x,y) - \bar{f}(x,y)$$

$f_s(x,y) \Rightarrow$ sharpened image

$\bar{f}(x,y) \Rightarrow$ blurred image of $f(x,y)$

Generalization of *unsharp masking* ☞ high-boost filtering

$$f_{hb}(x,y) = Af(x,y) - \bar{f}(x,y) \quad A \geq 1$$

1st Derivative Filtering

- Implementing 1st derivative filters is difficult in practice

- For a function f (x, y) the gradient of f at coordinates (x, y) is given as the column vector:

$$\nabla \mathrm{f} = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \dfrac{\partial f}{\partial x} \\ \dfrac{\partial f}{\partial y} \end{bmatrix}$$

1st Derivative Filtering (cont…)

- The magnitude of this vector is given by:

$$\begin{aligned}\nabla f &= mag(\nabla \mathrm{f}) \\ &= \left[G_x^2 + G_y^2\right]^{1/2} \\ &= \left[\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2\right]^{1/2}\end{aligned}$$

- For practical reasons this can be simplified as:

$$\nabla f \approx \left|G_x\right| + \left|G_y\right|$$

1st Derivative Filtering (cont…)

- Now we want to define digital approximations and their Filter Masks
- For simplicity we use a 3x3 region
- For example z_5 denotes f(x,y), z_1 denotes f(x-1,y-1)
- A simple approximation for First Derivative is

$$G_x = (z_8 - z_5) \text{ and } G_y = (z_6 - z_5).$$

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

1st Derivative Filtering (cont…)

A simple approximation for First Derivative is

$$G_x = (z_8 - z_5) \text{ and } G_y = (z_6 - z_5).$$

Two other definitions proposed by Roberts use cross- difference

$$G_x = (z_9 - z_5) \quad \text{and} \quad G_y = (z_8 - z_6).$$

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

If we use

$$\nabla f = \left[\left(\frac{\partial f}{\partial x} \right)^2 + \left(\frac{\partial f}{\partial y} \right)^2 \right]^{1/2}$$

$$\nabla f = \left[(z_9 - z_5)^2 + (z_8 - z_6)^2 \right]^{1/2}$$

Gradient Operators

- Based on *grey scale gradient* at a pixel

$$g_x(x,y) \approx f(x+1,y) - f(x-1,y)$$

$$g_y(x,y) \approx f(x,y+1) - f(x,y-1)$$

x

y

98	98	98	106
10	103	110	116
51	38	60	110
95	68	60	29
110	116	95	65
103	110	116	120

100	13
9	72
-35	39
-15	51

-60	-38
-35	-50
78	35
42	56

What is

$$\sqrt{g_x^2 + g_y^2}\ ?$$

Answer:

117	40
36	87
85	52
44	76

Assuming threshold T = 50,
a pixel is selected if >= T

1st Derivative Filtering (cont…)

If we use absolute values then

$$\nabla f \approx |G_x| + |G_y|.$$

$$\nabla f \approx |z_9 - z_5| + |z_8 - z_6|.$$

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

The Masks corresponding to these equations are:

−1	0
0	1

0	−1
1	0

Roberts Cross-Gradient Operators

Gradient Operators

Normally the smallest mask used is of size 3 x 3

Based on the concept of approximating the gradient several spatial masks have been proposed:

Prewitt operator

−1	−1	−1
0	0	0
1	1	1

Extract horizontal edges

−1	0	1
−1	0	1
−1	0	1

Extract vertical edges

Pixel arrangement

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

Prewitt operators (equation):

$$\nabla f \approx \left|(z_7 + z_8 + z_9) - (z_1 + z_2 + z_3)\right| + \left|(z_3 + z_6 + z_9) - (z_1 + z_4 + z_7)\right|$$

Gradient Operators

Sobel operators

−1	−2	−1
0	0	0
1	2	1

Extract horizontal edges

−1	0	1
−2	0	2
−1	0	1

Extract vertical edges

Sobel operators (equation):

$$\nabla f \approx \left|(z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3)\right| + \left|(z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7)\right|$$

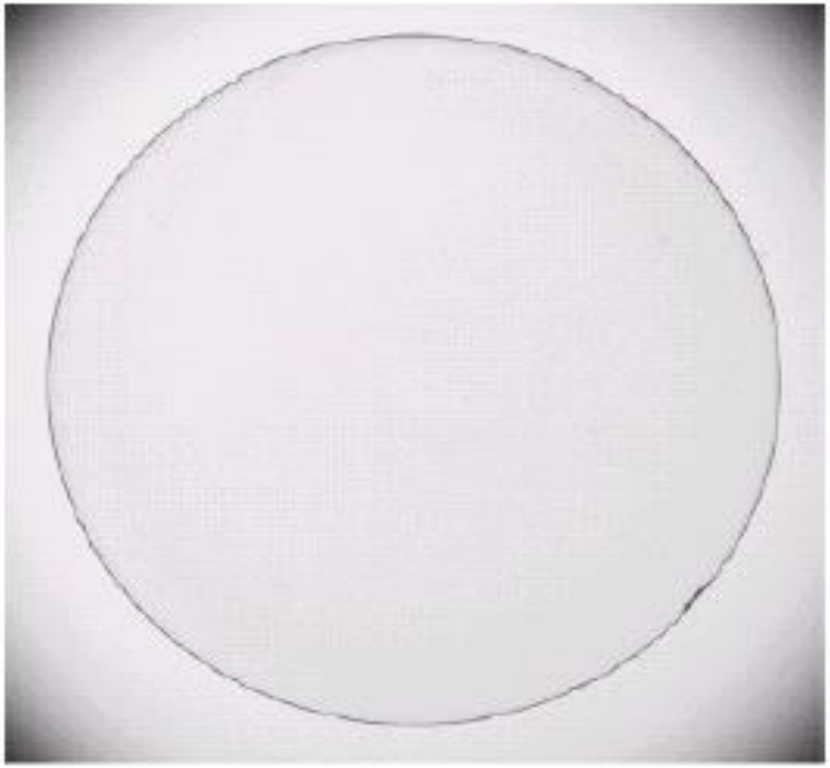
Emphasize more the current position (y)

Emphasize more the current position (x)

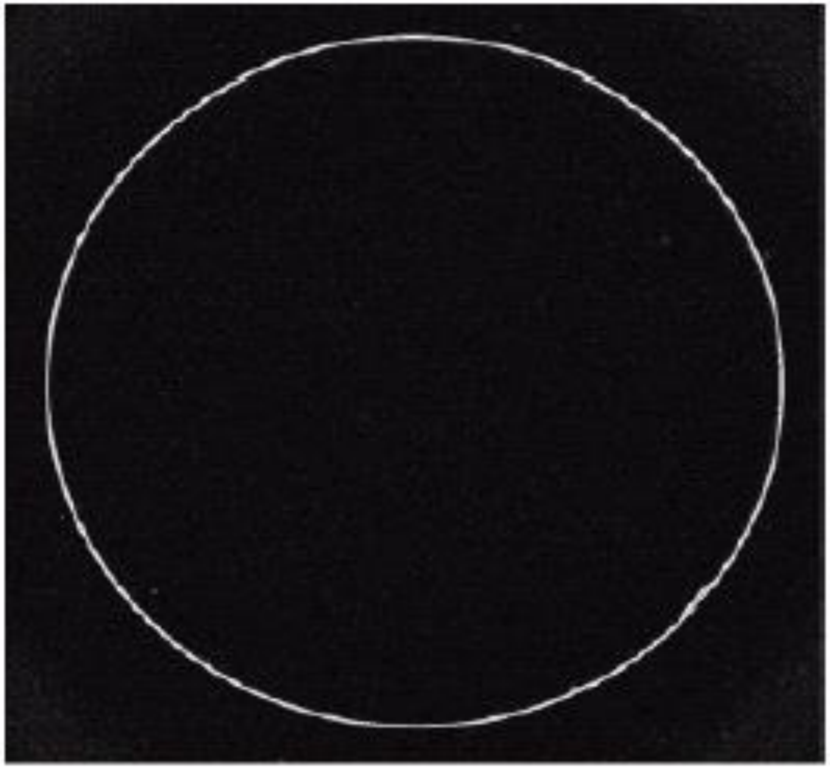
Pixel arrangement

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

Gradient Processing (Example)



Optical image of contact lens
(note defects at 4 and 5 o' clock



Sobel gradient

NOTE

- The summation of coefficients in all masks equals 0, indicating that they would give a response of 0 in an area of constant gray level.

Mask used to estimate the Gradient

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

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

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

Canny Edge Detection

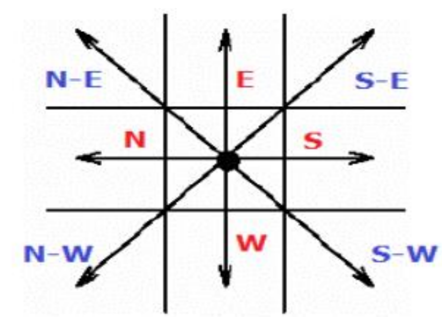
- The **Canny edge detector** is an edge detection operator that uses a multi-stage algorithm to detect a wide range of edges in images. It was developed by John F. Canny in 1986.

Canny Edge Detection

- The Process of Canny edge detection algorithm can be broken down to 5 different steps:
  - Apply Gaussian filter to smooth the image in order to remove the noise
  - Find the intensity gradients of the image
  - Apply non-maximum suppression to get rid of spurious response to edge detection
  - Apply double threshold to determine potential edges
  - Track edge by hysteresis: Finalize the detection of edges by suppressing all the other edges that are weak and not connected to strong edges.

Non-Maximum Suppression

- Non-maximum suppression is an edge thinning technique.
- Non-Maximum suppression is applied to "thin" the edge.
- Compare the edge strength of the current pixel with the edge strength of the pixel in the positive and negative gradient directions.
- If the edge strength of the current pixel is the largest compared to the other pixels in the mask with the same direction (i.e., the pixel that is pointing in the y direction, it will be compared to the pixel above and below it in the vertical axis), the value will be preserved. Otherwise, the value will be suppressed.



Double Threshold

- After application of non-maximum suppression, remaining edge pixels provide a more accurate representation of real edges in an image.
- However, some edge pixels remain that are caused by noise and color variation.
- Filter out edge pixels with a weak gradient value and preserve edge pixels with a high gradient value. This is accomplished by selecting high and low threshold values.
- If an edge pixel's gradient value is higher than the high threshold value, it is marked as a strong edge pixel.
- If an edge pixel's gradient value is smaller than the high threshold value and larger than the low threshold value, it is marked as a weak edge pixel.
- If an edge pixel's value is smaller than the low threshold value, it will be suppressed.
- The two threshold values will depend on the content of a given input image.

Edge tracking by hysteresis

- Usually a weak edge pixel caused from true edges will be connected to a strong edge pixel while noise responses are unconnected.

- To track the edge connection, blob analysis is applied by looking at a weak edge pixel and its 8-connected neighborhood pixels. As long as there is one strong edge pixel that is involved in the blob, that weak edge point can be identified as one that should be preserved.

Matlab command

```
BW = edge(I)
BW = edge(I,'Sobel')
BW = edge(I,'Sobel',threshold)
```

```
BW = edge(I,'Prewitt')
BW = edge(I,'Prewitt',threshold)
```

```
BW = edge(I,'Roberts')
BW = edge(I,'Roberts',threshold)
```

```
W = edge(I,'Canny')
BW = edge(I,'Canny',threshold)
```

End of Lecture