Chapter 3 Spatial Filtering

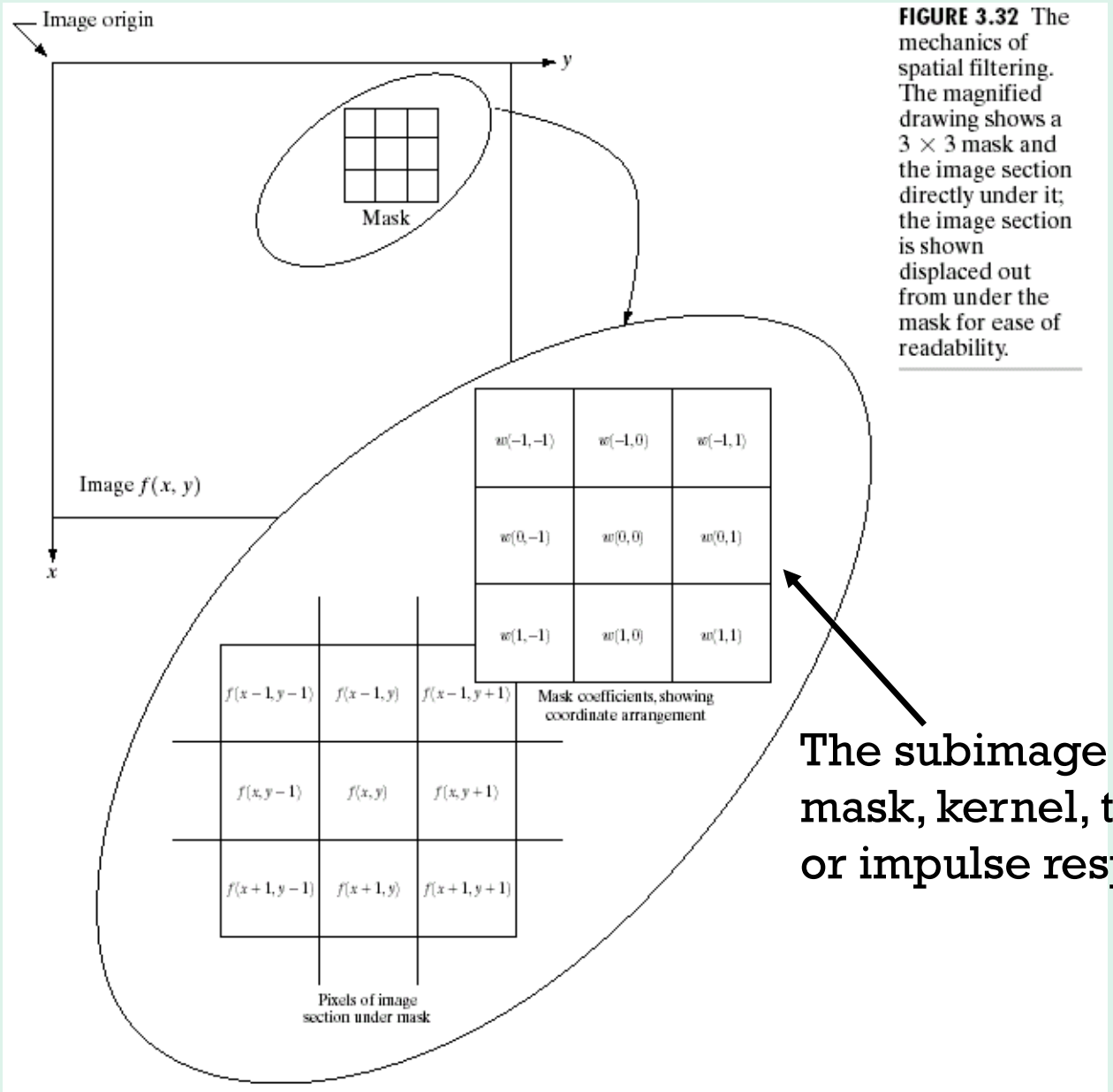


FIGURE 3.32 The mechanics of spatial filtering. The magnified drawing shows a 3×3 mask and the image section directly under it; the image section is shown displaced out from under the mask for ease of readability.

The subimage is called a filter, mask, kernel, template, window, or impulse response.

Linear Filtering (Convolution)

Linear filtering computes the sum of products repeatedly over an entire image. In other words, linear filtering is performed by convolving a mask with an image.

$$R = w_1 z_1 + w_2 z_2 + \cdots + w_9 z_9 = \sum_{i=1}^{9} w_i z_i$$

FIGURE 3.33 Another representation of a general 3×3 spatial filter mask.

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

Linear Smoothing Filters

Smoothing filters are also referred to as averaging filters or low pass filters. The general form of low pass filtering is weighted averaging.

$$g(x,y) = \frac{\sum_{s=-a}^{a}\sum_{t=-b}^{b} w(s,t)\, f(x+s, y+t)}{\sum_{s=-a}^{a}\sum_{t=-b}^{b} w(s,t)}$$

(x, y: image coordinates, s, t: mask coordinates)

Masks for Low Pass Filters

$$R = \frac{1}{9}\sum_{i=1}^{9} z_i \qquad R = \frac{1}{16}\sum_{i=1}^{9} w_i z_i$$

$\frac{1}{9} \times$

1	1	1
1	1	1
1	1	1

$\frac{1}{16} \times$

1	2	1
2	4	2
1	2	1

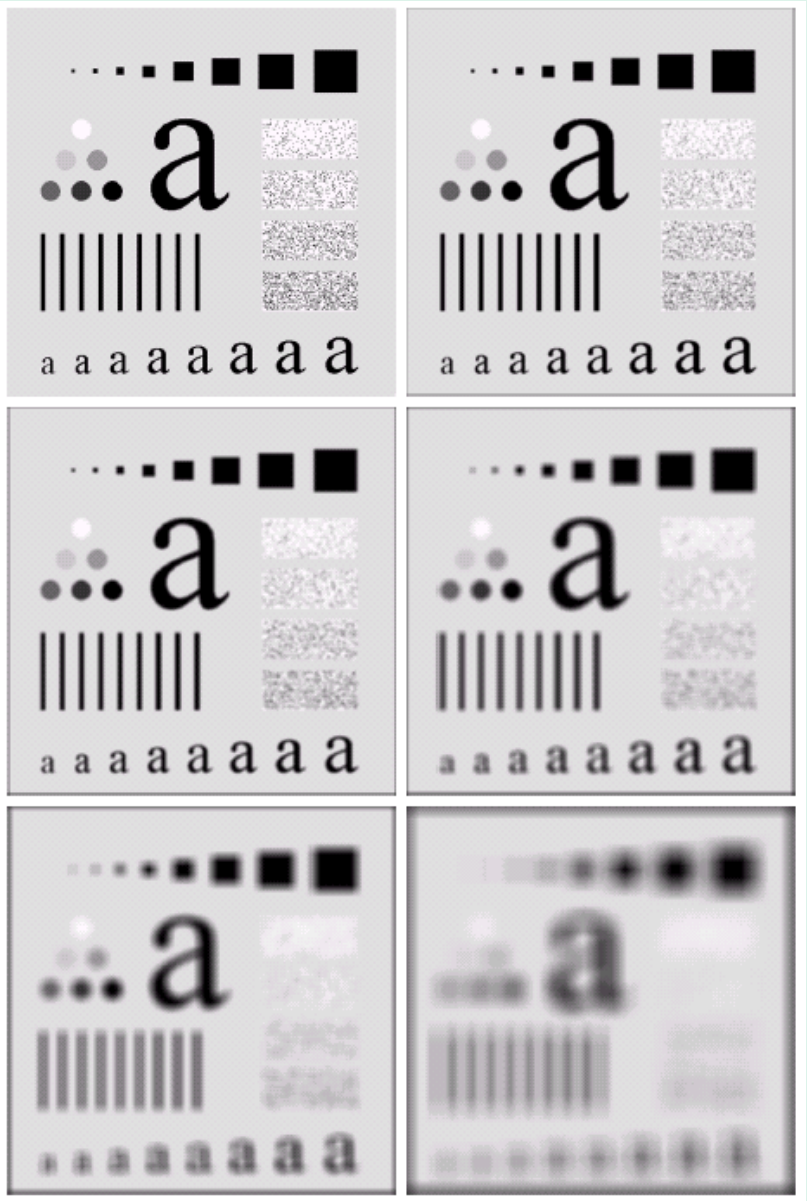
a b

FIGURE 3.34 Two 3×3 smoothing (averaging) filter masks. The constant multiplier in front of each mask is equal to the sum of the values of its coefficients, as is required to compute an average.

Mean filter mask (Averaging filter)

Weighted averaging filter

Low Pass Filtering Results



Original Image (500×500)	3×3 mean filter
5×5	9×9
15× 15	35×35

Exercise 1: Mean Filtering

Perform the low pass filtering below manually.

6	6	6	6	6	6	6
5	5	5	5	5	5	5
4	4	4	4	4	4	4
3	3	3	10	3	3	3
2	2	2	2	2	2	2
1	1	1	1	1	1	1
0	0	0	0	0	0	0

A Digital Image

*

1	1	1
1	1	1
1	1	1

/ 9

A Convolution Mask

Exercise 2: Weighted Averaging Filtering

Perform the low pass filtering below manually.

6	6	6	6	6	6	6
5	5	5	5	5	5	5
4	4	4	4	4	4	4
3	3	3	10	3	3	3
2	2	2	2	2	2	2
1	1	1	1	1	1	1
0	0	0	0	0	0	0

A Digital Image

*

1	2	1
2	4	2
1	2	1

/ 16

A Convolution Mask

MATLAB Commands for Filtering

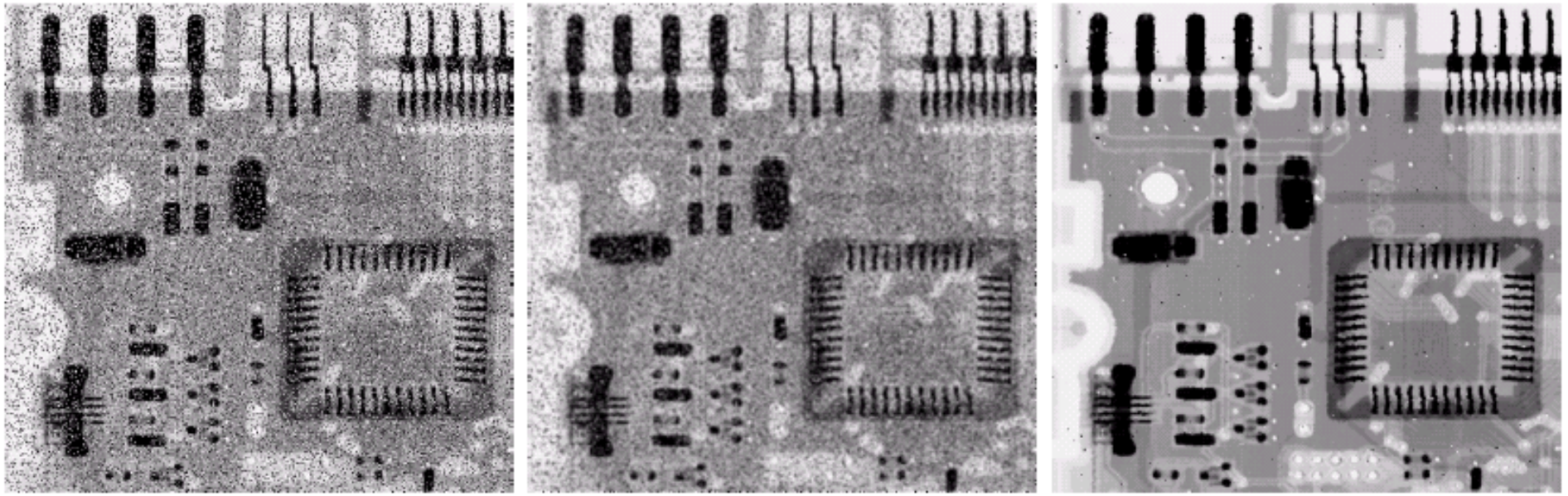
```
%Define an image
>> img=[6 6 6 6 6 6 6; 5 5 5 5 5 5 5; 4 4 4 4 4 4 4; 3 3 3
    10 3 3 3; 2 2 2 2 2 2 2; 1 1 1 1 1 1 1;0 0 0 0 0 0 0]

%Define a mask for filtering in two ways
>> mask=fspecial('average',3)
>> mask=[1 1 1; 1 1 1; 1 1 1]/9

%Filtering in two ways
>> output=filter2(mask,img,'same')
>> output=conv2(img,mask,'same')
```

Nonlinear Smoothing Filters (Median Filters)

The median filter replaces the value of a pixel by the median of the gray levels in the neighborhood.

$$\hat{f}(x,y) = \underset{(s,t \in S_{xy})}{\text{median}}\{g(s,t)\}$$

Effect of Median Filtering



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.)

Exercise 3: Median Filter

Apply the 3 × 3 Median filter to the images below.

5	5	5	5	5
4	4	4	4	4
3	3	10	3	3
2	2	2	2	2
1	1	1	1	1

(a) Image A

0	0	0	0	0
0	0	0	0	0
0	0	10	10	10
0	0	10	10	10
0	0	10	10	10

(b) Image B

Exercise 4: Median Filter

Apply the 3 × 3 Median filter to the images below.

10	10	0	10	10
10	10	0	10	10
10	10	0	10	10
10	10	0	10	10
10	10	0	10	10

(a) Image A

0	0	10	0	0
0	0	10	0	0
0	0	10	0	0
0	0	10	0	0
0	0	10	0	0

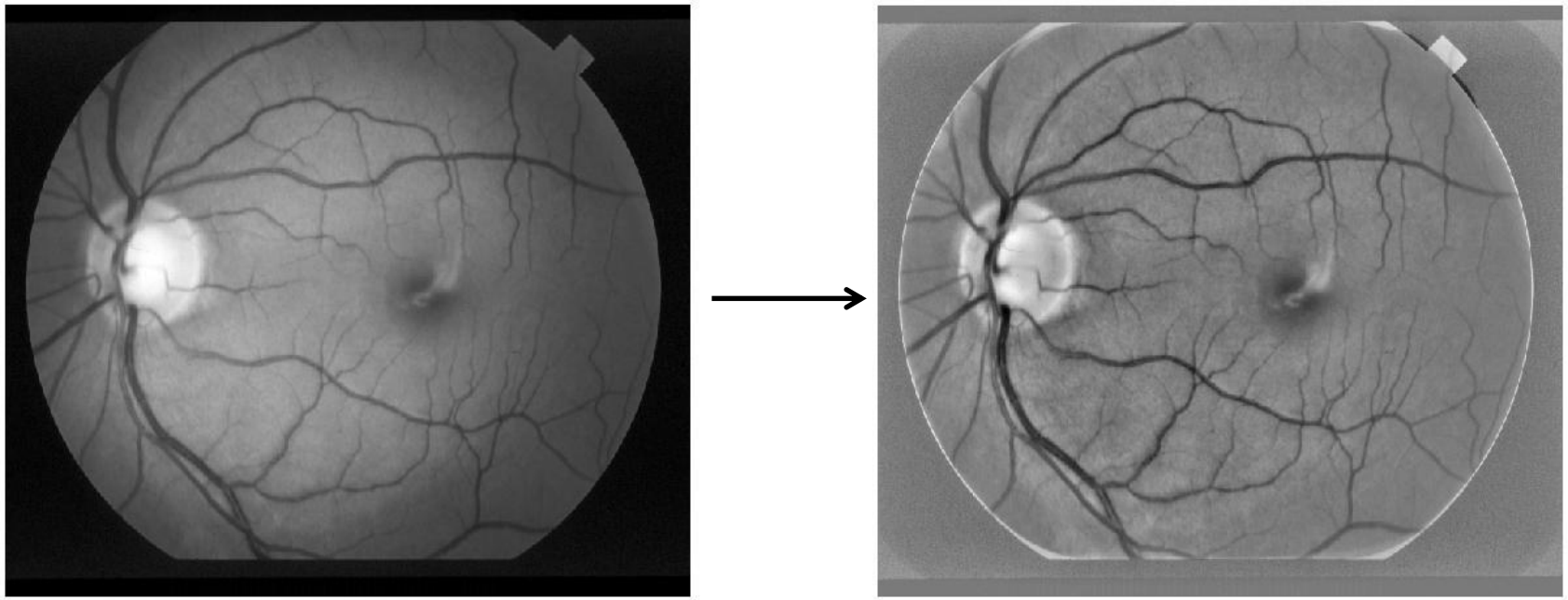
(b) Image B

MATLAB Commands for Median Filtering

```
%Define an image
>> img=[5 5 5 5 5; 4 4 4 4 4; 3 3 10 3 3; 2 2 2 2 2; 1 1 1 1
    1;0 0 0 0 0]

%Median filtering
>> output=medfilt2(img,[3 3],'symmetric')
```

Exercise 5: Median Filter

The left image is an image subject to non-uniform illumination. Devise a method to correct this irregular lighting (right image) using the Median filter.



Linear Sharpening Filters

Sharpening filters are equivalent to high pass filters and are based on first- and second-order derivatives.

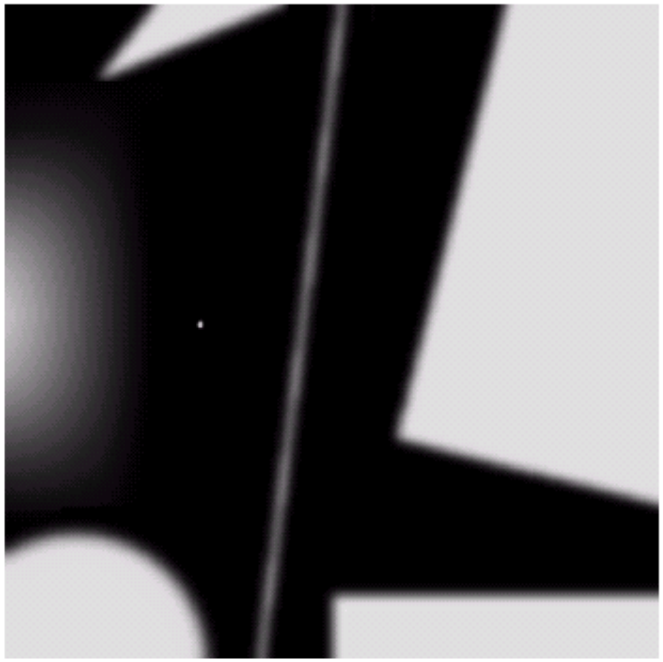
$$\frac{\partial f}{\partial x} = f(x+1) - f(x).$$

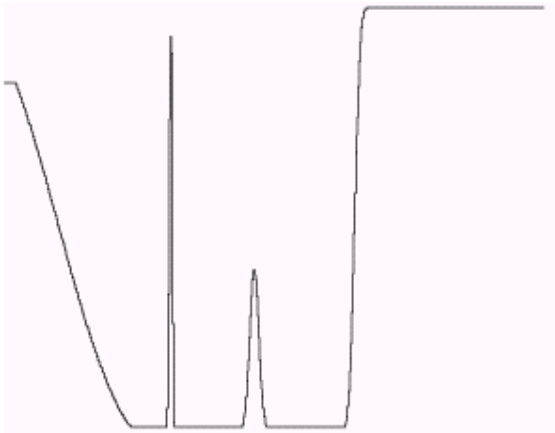
$$\frac{\partial^2 f}{\partial x^2} = f(x-1) - 2f(x) + f(x+1).$$

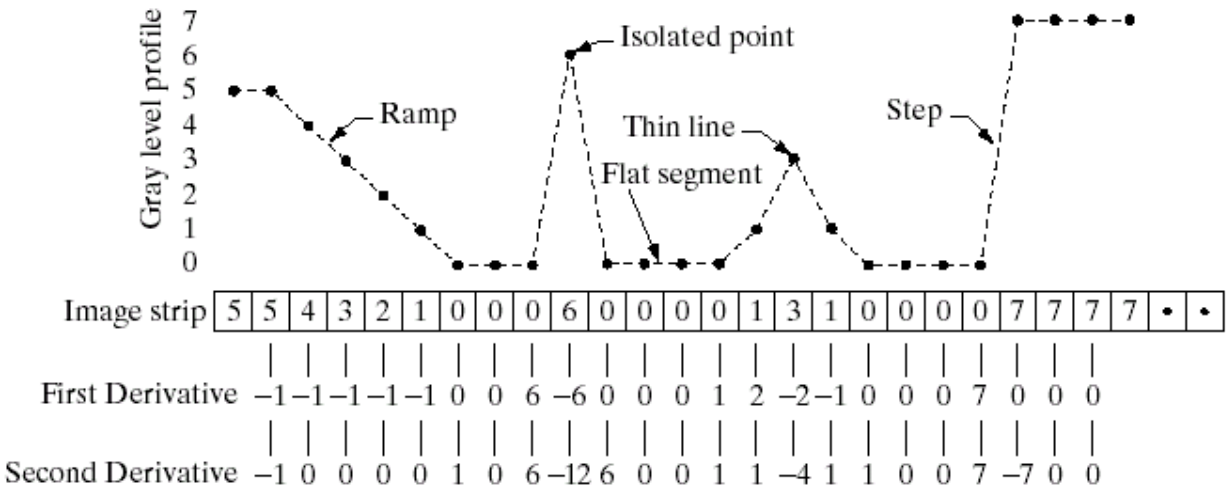
First- and Second-order Derivatives

a b
c

FIGURE 3.38
(a) A simple image. (b) 1-D horizontal gray-level profile along the center of the image and including the isolated noise point.
(c) Simplified profile (the points are joined by dashed lines to simplify interpretation).







First Derivatives (Gradient)

The gradient of an image *f* is defined by

$$\nabla f = \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \partial f / \partial x \\ \partial f / \partial y \end{bmatrix}$$

The magnitude of the gradient is given by

$$|\nabla f| = \sqrt{g_x^2 + g_y^2} = \sqrt{(\partial f / \partial x)^2 + (\partial f / \partial y)^2}$$

The direction of the gradient is given by

$$\theta(x, y) = \tan^{-1}(g_y / g_x)$$

Discrete Approximation of Gradient 1

Using two-point approximation,

$$\frac{\partial f(x,y)}{\partial x} \approx \frac{f(x+\Delta x, y) - f(x,y)}{\Delta x}$$

$$\frac{\partial f(x,y)}{\partial y} \approx \frac{f(x, y+\Delta y) - f(x,y)}{\Delta y}$$

which correspond to mask operations with

$$\begin{bmatrix} -1 \\ 1 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} -1 & 1 \end{bmatrix}$$

Discrete Approximation of Gradient 2

Using a centered difference approximation,

$$\frac{\partial f(x,y)}{\partial x} \approx \frac{f(x+\Delta x, y) - f(x-\Delta x, y)}{2\Delta x}$$

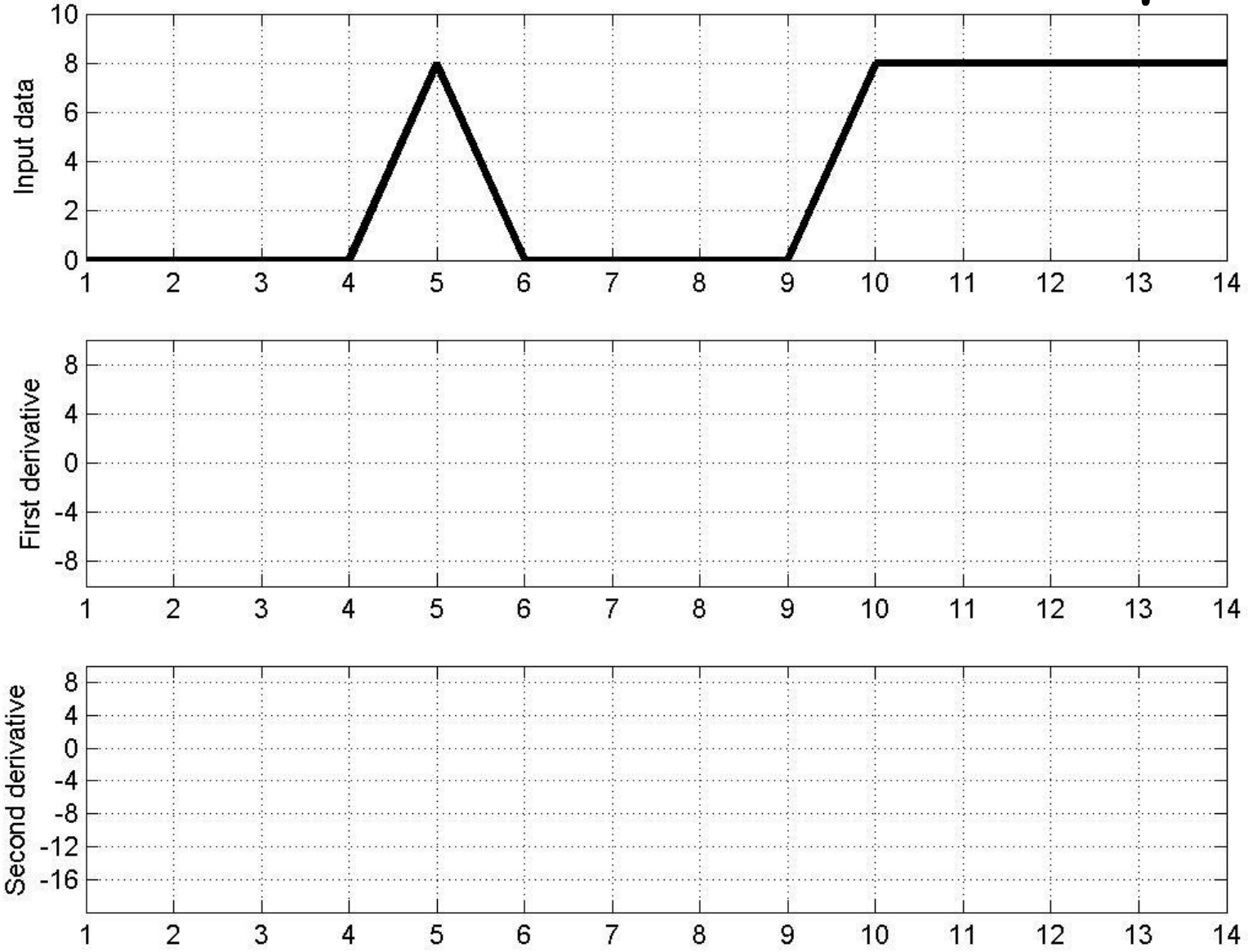
$$\frac{\partial f(x,y)}{\partial y} \approx \frac{f(x, y+\Delta y) - f(x, y-\Delta y)}{2\Delta y}$$

which is equivalent to masking operations with

$$\begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} \quad \textbf{and} \quad \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$$

Exercise 6: First & Second Derivatives

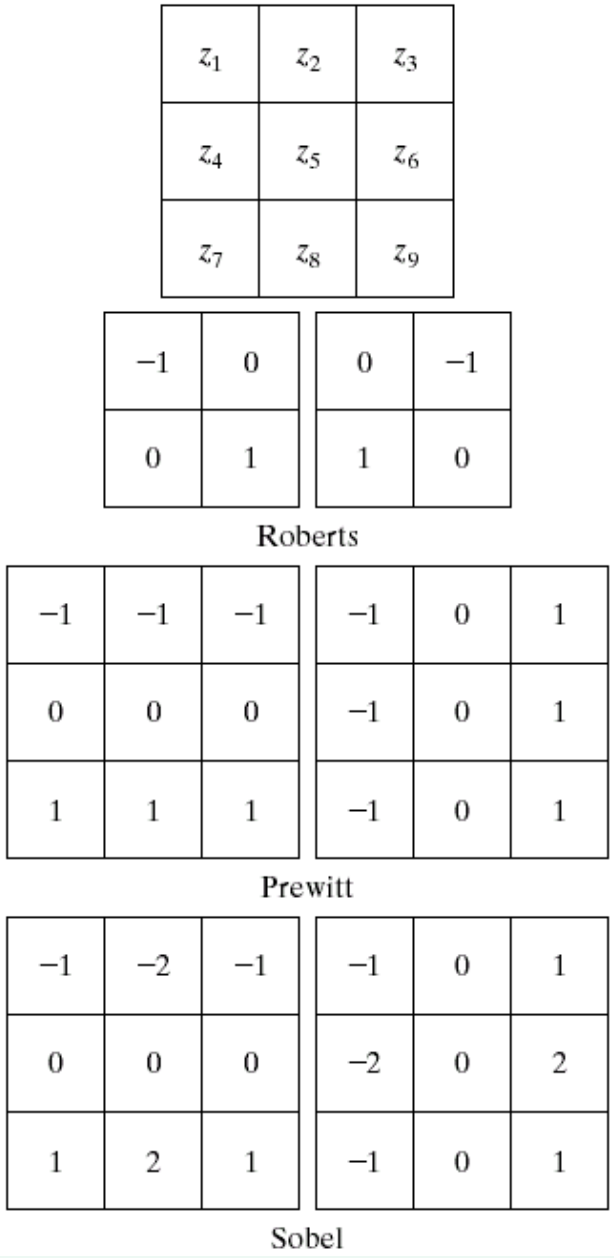
Plot the first and second-order derivatives of the input signal.



Gradient Operators

a
b c
d e
f g

FIGURE 10.8
A 3×3 region of an image (the z's are gray-level values) and various masks used to compute the gradient at point labeled z_5.

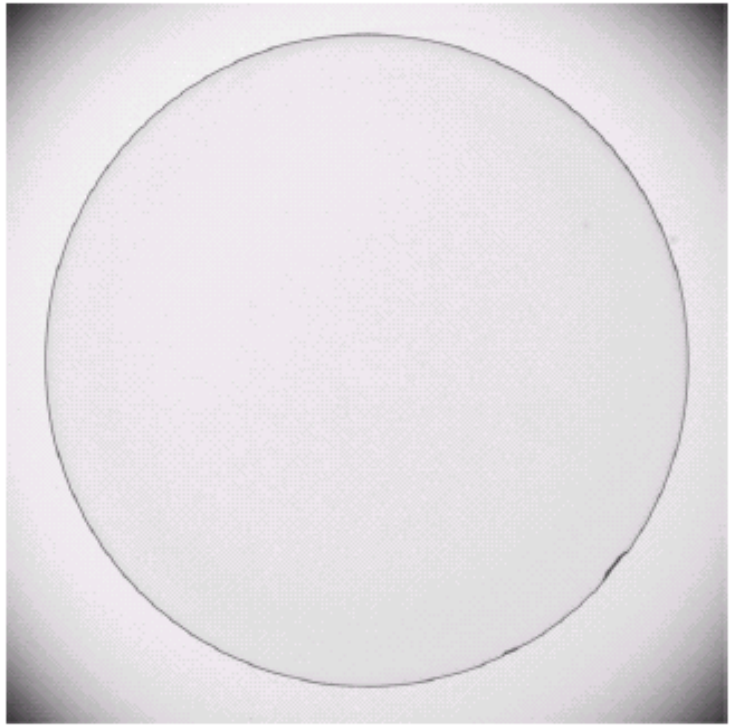


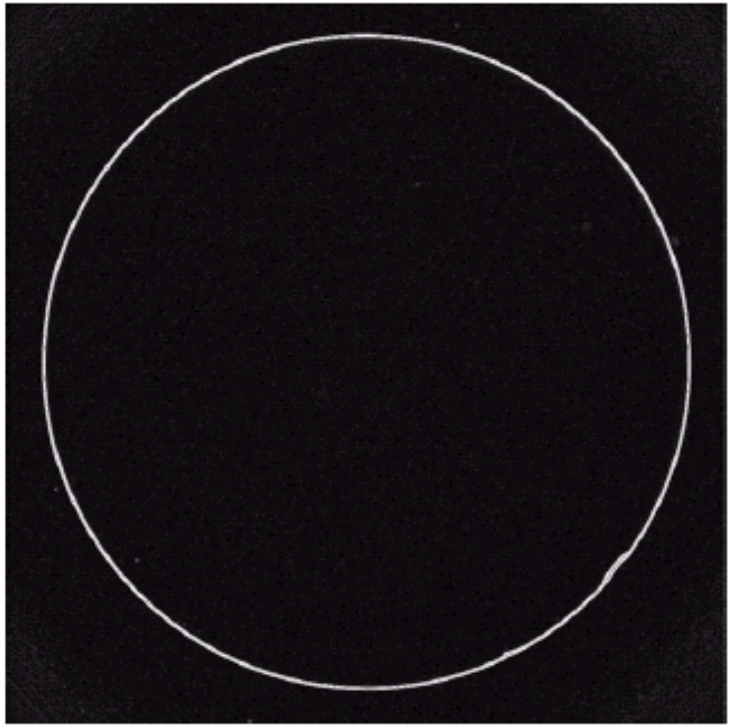
Roberts Operators

Prewitt Operators

Sobel Operators

Edge Detection by Sobel Operators





a b

FIGURE 3.45 Optical image of contact lens (note defects on the boundary at 4 and 5 o'clock). (b) Sobel gradient. (Original image courtesy of Mr. Pete Sites, Perceptics Corporation.)

$$|\nabla f|$$

Exercise 7: Prewitt Operators

Apply the Prewitt operators to the image below and calculate the magnitude of the gradient.

0	0	0	0	0	0	0
0	0	0	0	0	0	1
0	0	0	0	0	1	1
0	0	0	0	1	1	1
0	0	0	1	1	1	1
0	0	1	1	1	1	1
0	1	1	1	1	1	1

MATLAB Commands for Edge Detection

```
%Call the Prewitt operators
>> mask1=fspecial('Prewitt')
>> mask2=mask1'

%Filtering (i.e., covolution)
>> tmp1=conv2(img,mask1,'same');
>> tmp2=conv2(img,mask2,'same');

%Calculate the magnitude
>> mag=sqrt(tmp1.^2+tmp2.^2);

%Display it as an image
>> imshow(mat2gray(mag))
```

Second Derivatives (Laplacian)

The Laplacian is a second-order derivative operator defined as

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

Discrete approximation:

$$\frac{\partial^2 f}{\partial x^2} \approx \big(f(x+\Delta x, y) - f(x,y)\big) - \big(f(x,y) - f(x-\Delta x, y)\big)$$

$$\approx f(x+\Delta x, y) - 2f(x,y) + f(x-\Delta x, y)$$

which may be represented by masking operations with

$$\begin{bmatrix} 1 & -2 & 1 \end{bmatrix}$$

Laplacian Operators

0	1	0
1	−4	1
0	1	0

1	1	1
1	−8	1
1	1	1

0	−1	0
−1	4	−1
0	−1	0

−1	−1	−1
−1	8	−1
−1	−1	−1

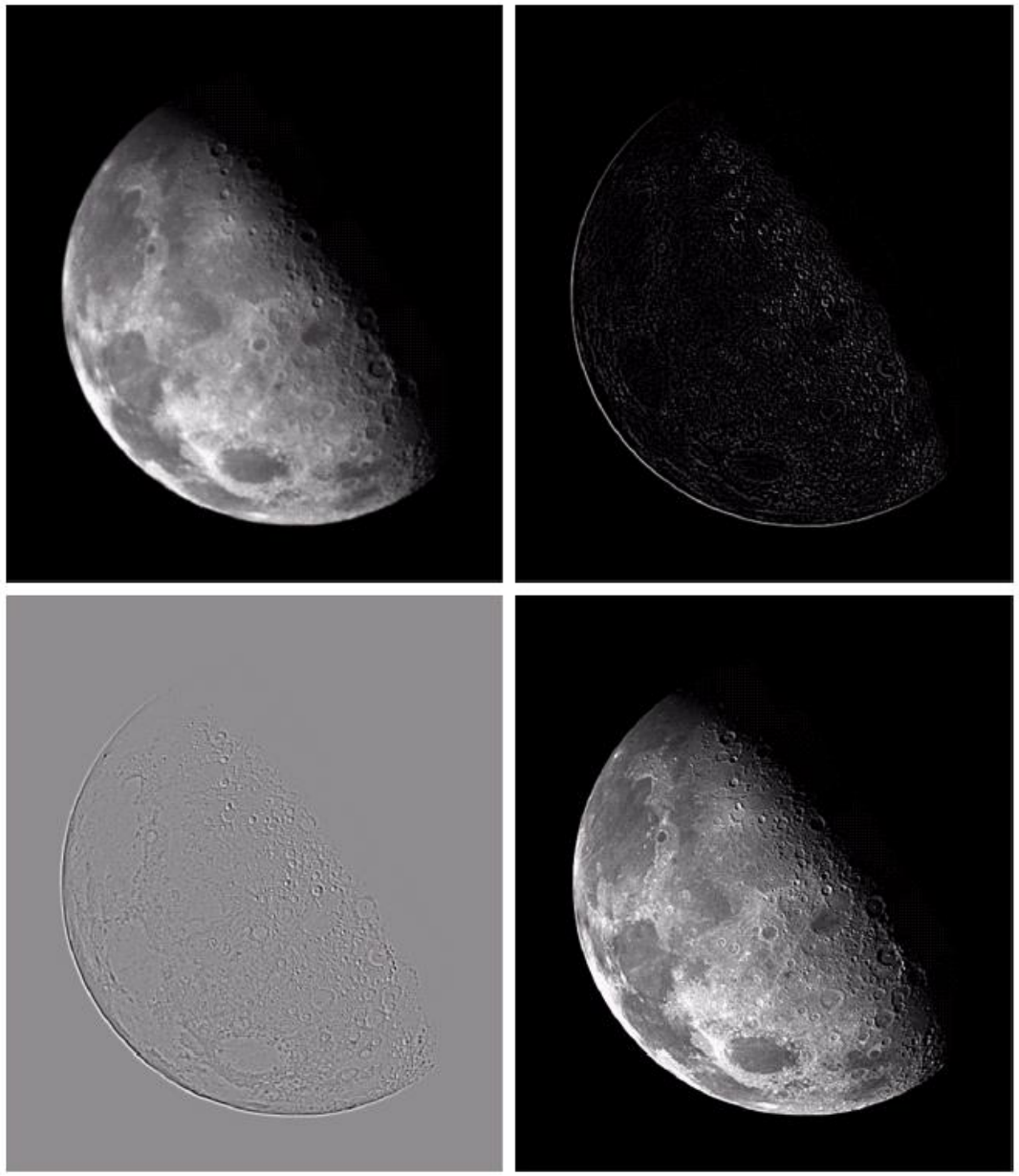
a b
c d

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.

Enhancement by the Laplacian

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.)



Exercise 8: Laplacian Operator

Apply the Laplacian operator to the image below.

0	0	0	0	0	0	0
0	0	0	0	0	0	1
0	0	0	0	0	1	1
0	0	0	0	1	1	1
0	0	0	1	1	1	1
0	0	1	1	1	1	1
0	1	1	1	1	1	1

$*$

1	1	1
1	-8	1
1	1	1

MATLAB Commands for Laplacian

```
%Call the Laplacian operator
>> mask=fspecial('Laplacian')

%Filtering (i.e., covolution)
>> tmp=conv2(img,mask,'same');

%Display it as an image
>> imshow(mat2gray(tmp))
```

MATLAB Commands for Enhancement

```
%Enhancement
>> tmp=img-tmp*0.1

%Display the enhanced image
>> imshow(mat2gray(tmp))
```

Assignment from Chapter 3

1. Add ‘Gaussian’ and ‘Salt&Pepper’ noise to your image separately. Apply averaging filters and Median filters to the noisy images. (See Chap3_1.m) (>>nimg=imnoise(img,’salt & pepper’,0.05);)

2. Apply the Sobel and Laplacian filters to your original facial image separately. Also attempt to enhance your image using the Laplacian filter. (See Chap3_2.m)

Write a concise report in one week, including your resulting images and comments on them.

Assignment 1: Sample Results

Original Image

Noisy Image (Gaussian)

Averaging Filtered

Median Filtered

Assignment 1: Sample Results

Original Image

Noisy Image (Salt&Pepper)

Averaging Filtered

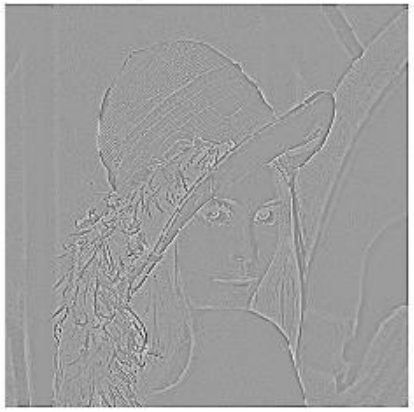
Median Filtered

Assignment 2: Sample Results

Original Image

Sobel Filtered

Laplacian Filtered



Enhanced Image

Image Thresholding

Definition

The simplest thresholding methods replace each pixel in an image with a black pixel if the image intensity $f(x, y)$ is less than a fixed value called the threshold (T), or a white pixel if the pixel intensity is greater than that threshold. In the example image on the right, this results in the dark tree becoming completely black, and the bright snow becoming completely white.

Automatic thresholding

Definition

While in some cases, the threshold *T* can be selected manually by the user, there are many cases where the user wants the threshold to be automatically set by an algorithm. In those cases, the threshold should be the "best" threshold in the sense that the partition of the pixels above and below the threshold should match as closely as possible the actual partition between the two classes of objects represented by those pixels (e.g., pixels below the threshold should correspond to the background and those above to some objects of interest in the image).

Automatic thresholding

Threshold Model

Many types of automatic thresholding methods exist, the most famous and widely used being Otsu's method.

- Histogram shape-based methods, where, for example, the peaks, valleys and curvatures of the smoothed histogram are analyzed.[2] Note that these methods, more than others, make certain assumptions about the image intensity probability distribution
- Clustering-based methods, where the gray-level samples are clustered in two parts as background and foreground
- Entropy-based methods result in algorithms that use the entropy of the foreground and background regions, the cross-entropy between the original and binarized image, etc.,
- Object Attribute-based methods search a measure of similarity between the gray-level and the binarized images, such as fuzzy shape similarity, edge coincidence, etc.,

Otsu's Algorithm

Otsu's method [edit]

The algorithm exhaustively searches for the threshold that minimizes the intra-class variance, defined as a weighted sum of variances of the two classes:

$$\sigma_w^2(t) = \omega_0(t)\sigma_0^2(t) + \omega_1(t)\sigma_1^2(t)$$

Weights ω_0 and ω_1 are the probabilities of the two classes separated by a threshold t ,and σ_0^2 and σ_1^2 are variances of these two classes.

The class probability $\omega_{0,1}(t)$ is computed from the L bins of the histogram:

$$\omega_0(t) = \sum_{i=0}^{t-1} p(i)$$

$$\omega_1(t) = \sum_{i=t}^{L-1} p(i)$$

For 2 classes, minimizing the intra-class variance is equivalent to maximizing inter-class variance:[2]

$$\begin{aligned}\sigma_b^2(t) &= \sigma^2 - \sigma_w^2(t) = \omega_0(t)(\mu_0 - \mu_T)^2 + \omega_1(t)(\mu_1 - \mu_T)^2 \\ &= \omega_0(t)\omega_1(t)[\mu_0(t) - \mu_1(t)]^2\end{aligned}$$

which is expressed in terms of class probabilities ω and class means μ, where the class means

$$\mu_0(t) = \frac{\sum_{i=0}^{t-1} ip(i)}{\omega_0(t)}$$

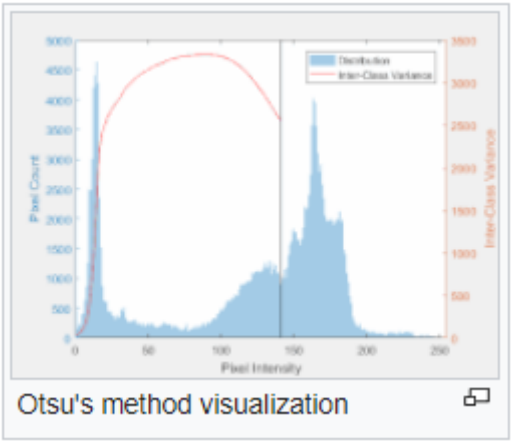
$$\mu_1(t) = \frac{\sum_{i=t}^{L-1} ip(i)}{\omega_1(t)}$$

$$\mu_T = \sum_{i=0}^{L-1} ip(i)$$

The following relations can be easily verified:

$$\omega_0\mu_0 + \omega_1\mu_1 = \mu_T$$

$$\omega_0 + \omega_1 = 1$$



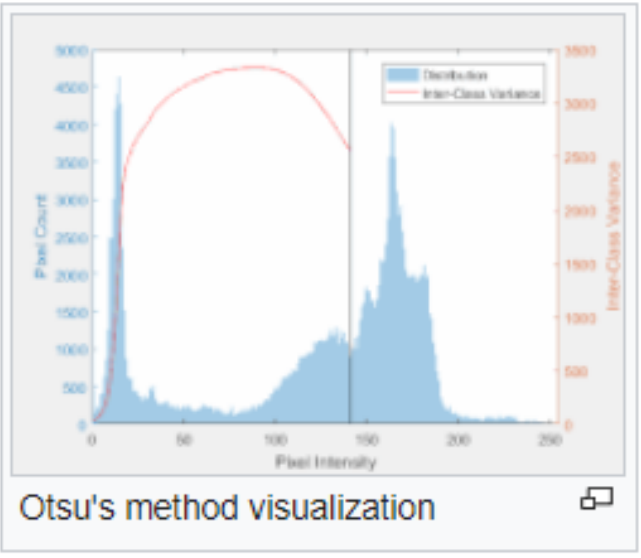
Otsu's method visualization

The class probabilities and class means can be computed iteratively. This idea yields an effective algorithm.

Otsu's Algorithm

Algorithm

1. Compute histogram and probabilities of each intensity level
2. Set up initial $w_i(0)$ and $\mu_i(0)$
3. Step through all possible thresholds $t = 1, \ldots$ maximum intensity
   - 3.1 Update $w_i(t)$ and $\mu_i(t)$
   - 3.2 Compute $\sigma_b^2(t)$
4. Desired threshold corresponds to the maximum $\sigma_b^2(t)$



Otsu's method visualization