Chapter 1 Introduction

Objectives

1. Enhance and improve the information of the picture for better interpretation and understanding
2. Processing for efficient data storage, transmission over network and extraction of right information from the picture.

1.1 Origin

The first industry to use digital images was the newspaper industry. The pictures were sent by submarine cable between London and New York.

1.2 Digital Image Processing

The human eye-brain mechanism produces the best imaging system. An image is an object or visual which one sees. It is a 2-dimensional function of a 3-dimensional world that surrounds us. Basically, images are 2-D light intensity function f(x, y) where x and y are spatial or plane co-ordinates and the amplitude at any co-ordinates pair (x, y) is defined as the intensity or gray level of the image at that point.
If x, y and the intensity values are all finite and discrete, then the image is known as a digital image.
The digital image is composed of a finite number of elements which has a particular location and value. These elements are called picture elements or pixels or pels.

Digital Image Processing is modification and enhancement of images by applying various filtering and enhancement techniques for perceiving better visual information and perform various analysis on the images.

Image processing accepts images as inputs and generates a modified image as output for human perception or the modified image may provide useful information.

1.3 Applications and Examples of Digital Image Processing

To develop a basic understanding of the breadth and length of the applications, the image processing applications are categorized according to their sources.

The principal source of energy for images is the electromagnetic energy spectrum. It also includes acoustic, ultrasonic, electronic etc.
Synthetic images used for modelling and visualization are generated in computer.
The most common applications of digital image processing are

1. Gamma-ray Imaging
   Imaging based on gamma rays are mostly for nuclear medicine, astronomical observations. In this type of imaging, images are produced from the emissions collected from gamma-ray detectors.
2. X-ray Imaging
   X-ray imaging is used mainly for medical diagnostics and industrial imaging. It is also used for astronomical applications.
3. Ultraviolet Imaging
   It is used for lithography, industrial inspection, microscopy, lasers, biological imaging and astronomical observations.
4. Wide applications include
   a. Remote Sensing
   b. Light microscopy
   c. Astronomy
   d. Weather observation and prediction
   e. Visual inspection of manufactured goods
   f. Traffic monitoring and surveillance
   g. License plate character recognition
   h. Currency recognition
   i. Finger-print and face recognition
   j. Radar imaging to explore inaccessible regions of the Earth’s surface.
   k. Mineral and oil exploration
   l. Ultrasound imaging of foetus

The list of applications is limited only for writing here, the applications of digital image processing is wide and the scope is large.

1.4 Fundamental Steps in Digital Image Processing

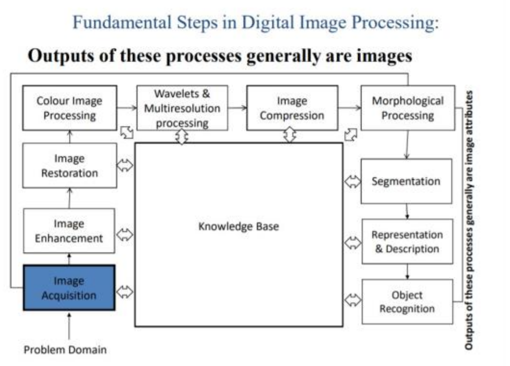

Fundamental Steps in Digital Image Processing:
Outputs of these processes generally are images
Colour Image Processing
Wavelets & Multiresolution processing
Image Compression
Morphological Processing
Image Restoration
Segmentation
Knowledge Base
Image Enhancement
Representation & Description
Image Acquisition
Object Recognition
Problem Domain
Outputs of these processes generally are image attributes

Fig:1.1 Image Courtesy://Digital Image Processing-Gonzalez; adapted from http://bharathuniv.ac.in

1. Image acquisition

   The image is captured by a sensor like camera or any analog device and digitized if the output of the camera or sensor is not already in digital form, using analogue-to-digital converter. It involves pre-processing of images.
2. Image Enhancement

   The process of manipulating an image so that the result is more suitable than the original for specific applications. The idea behind enhancement techniques is to bring out details that are hidden, or simple to highlight certain features of interest in an image

3. Image Restoration
   Deals with improving the appearance of an image.
   Based on mathematical or probabilistic models of image degradation.
4. Colour Image Processing
   Use the colour of the image to extract features of interest in an image.
   Understand the basics concepts of color models and color processing in digital domain.
5. Wavelets
   Wavelets are the foundation of representing images in various degrees of resolution. It is used for image data compression and for representation of images in smaller regions.
6. Compression
   Deals with various techniques used for reducing the storage required to save an image in digital form or the bandwidth required to transmit the images.
7. Morphological Processing
   Deals with tools for extracting image components that are useful in the representation and description of shape. In this step, there would be a transition from processes that output images, to processes that output image attributes.
8. Segmentation

Segmentation partitions an image into its constituent parts or objects. Autonomous segmentation is the most difficult tasks in digital image processing.
The more accurate the segmentation, the better automated object classification.

9. Feature extraction
   It consists of feature detection and feature description. Feature detection refers to finding a feature in an image, region or boundary. Feature description assigns quantitative attributes to the detected features.
10. Image pattern classification
   The process that assigns a label to an object based on its feature descriptors. There are various classification algorithms like correlation, Bayes classifiers to identify and predict the class label for the object.

1.5 Components of an Image Processing Systems

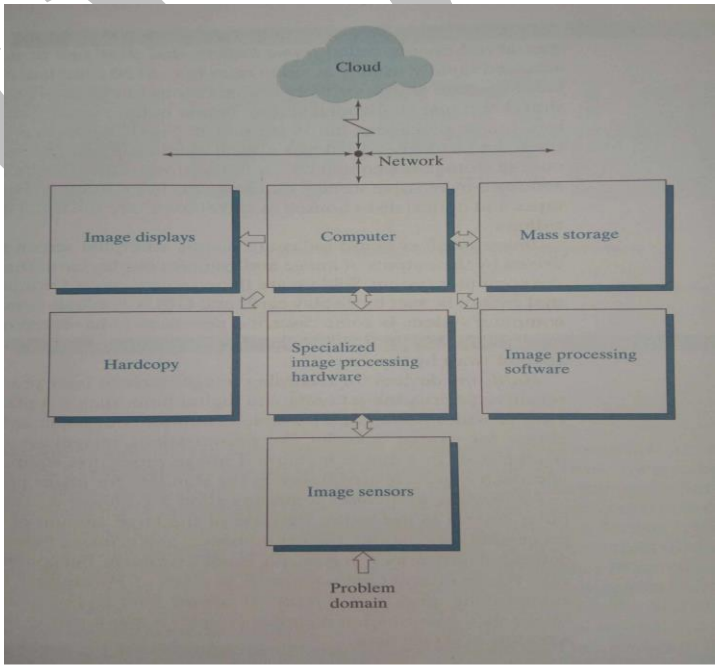


Fig 1.2: Components of a general-purpose image processing systems (Image courtesy:// Digital Image Processing -Fourth Edition)

1. Image Sensors
   Two subsystems are required to acquire digital images.
   The first is the physical device that is sensitive to the energy radiated by the object we wish to image (Sensor). The second, called a digitizer, is a device for converting the output of the physical sensing device into digital form. The digitizer converts the electrical outputs to digital data.
2. Specialized Image Processing Hardware
   It usually consists of the digitizer, and the hardware that performs other primitive operations, such as an arithmetic logic unit (ALU), which performs arithmetic and logical operations in parallel on entire images. This type of hardware is called as frontend subsystem, and its most distinguishing characteristic is speed. In other words, this unit performs functions that require fast data throughputs that the typical main computer cannot handle.
3. Computer
   The computer in an image processing system is a general-purpose computer and can range from a PC to a supercomputer.
4. Image Processing Software
   The software performs tasks with the help of specialized modules. There are many softwares available commercially.
5. Mass Storage
   Digital storage for image processing applications basically is divided into three principal categories.
   a) Short-term storage for use during processing
   b) On-line storage for fast recall
   c) Archival storage which is characterized by infrequent access.

   Storage is measured in bytes (Kbytes, Mbytes, Gbytes and Tbytes)

6. Image Displays

   Monitors are driven by the outputs of the image and graphics display cards that are an integral part of a computer system. There are variants in monitor displays.

7. Hardcopy devices
   Used for recording images, include laser printers, film cameras, heat-sensitive devices, inkjet units and digital units, such as optical and CD-ROM disks.
8. Networking and cloud communication
   Transmission bandwidth has improved due to optical fibre and other cloud technologies.

1.6 Elements of Visual Perception

Image processing applications produce images that are to be viewed by human observers. The Elements of Visual Perception are

1) Structure of the Human eye
2) Image formation in the eye
3) Brightness adaptation and discrimination

1) Structure of the Human eye

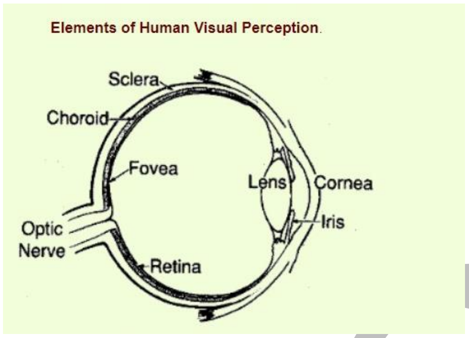


Fig 1.3: Simplified diagram of the human eye. (Image Courtesy: nptel)

The first part of the visual system is the eye. Its form is nearly spherical and its diameter is approximately 20 mm. Its outer cover consists of the ‘cornea' and ‘sclera'

The cornea is a tough transparent tissue in the front part of the eye. The sclera is an opaque membrane, which is continuous with cornea and covers the remainder of the eye. Directly below the sclera lies the “choroids”, which has many blood vessels. At its anterior extreme lies the iris diaphragm. The light enters in the eye through the central opening of the iris, whose diameter varies from 2mm to 8mm, according to the illumination conditions. Behind the iris is the “lens” which consists of concentric layers of fibrous cells and contains up to 60 to 70% of water. Its operation is similar to that of the man-made optical lenses. It focuses the light on the “retina” which is the innermost membrane of the eye.
Retina has two kinds of photoreceptors: cones and rods. The cones are highly sensitive to color. Their number is 6-7 million and they are mainly located at the central part of the retina. Each cone is connected to one nerve end.

Cone vision is the photopic or bright light vision. Rods serve to view the general picture of the vision field. They are sensitive to low levels of illumination and cannot discriminate colors. This is the scotopic or dim-light vision. Their number is 75 to 150 million and they are distributed over the retinal surface. Several rods are connected to a single nerve end. This fact and their large spatial distribution explain their low resolution.

Both cones and rods transform light to electric stimulus, which is carried through the optical nerve to the human brain for the high-level image processing and perception.

2) Image formation in the eye

- In the human eye, the distance between the centre of the lens and the imaging sensor is fixed
- Lens in the eye is flexible
- Shape controlled by muscles
- To focus on distance objects – Muscles flatten lens
- To focus on close objects – Muscles allow lens to thicken

3) Brightness Adaptation and Discrimination

• Digital Images are displayed as a discrete set of intensity

• Eye's ability to discriminate intensities at a given adaptation level is an important consideration when displaying images.

- Range of brightness's that can be discriminated simultaneously is small in comparison to total adaptation range.
- For a given set of conditions the current sensitivity level of the visual system is called the brightness adaptation level.

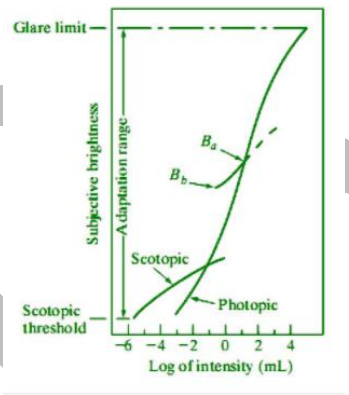


Fig 1.4: Range of subjective brightness adaptation level

Weber Ratio The ability of the eye to discriminate between changes in light intensity at any specific adaptation level.

The quantity ΔIc/I, ΔIc is the increment of illumination discriminable 50% of the time with background illumination I which is constant, **is called the Weber Ratio.**

Small weber ratio- good brightness discrimination

Large weber ratio-poor brightness discrimination

Visual Perception

Perceived brightness is not a simple function of intensity

Mach bands- Visual system tends to undershoot or overshoot around the boundaries of regions with different intensities.

Simultaneous Contrast- regions perceived interest does not depend on intensity.

Optical illusions- Eyes fill in non-existing information or perceives geometry properties of objects in an incorrect manner.

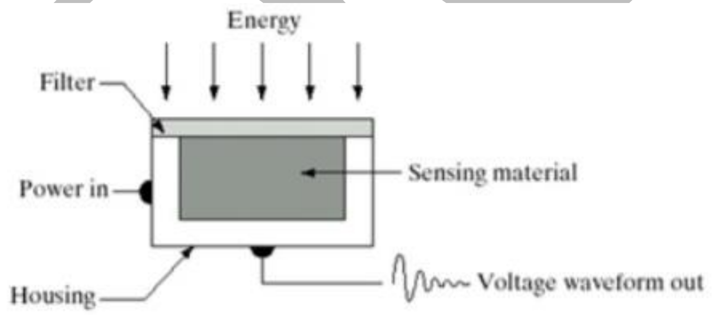
1.7 Image Sensing and Acquisition

Images are generated by the combination of illumination of source and the reflection or absorption of energy from that source by element of the scene being imaged.

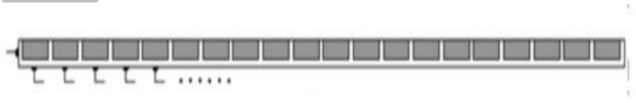
There are three principal sensor arrangements that can be used to transform incident energy into digital images.

Incoming energy is transformed into a voltage pulses by input electric power and sensor response where a digital quantity is obtained by digitizing the response.

1.7.1 Image Acquisition using a single sensing element



1.7.2 Image Acquisition using Line sensor



1.7.3 Image Acquisition using Array sensor

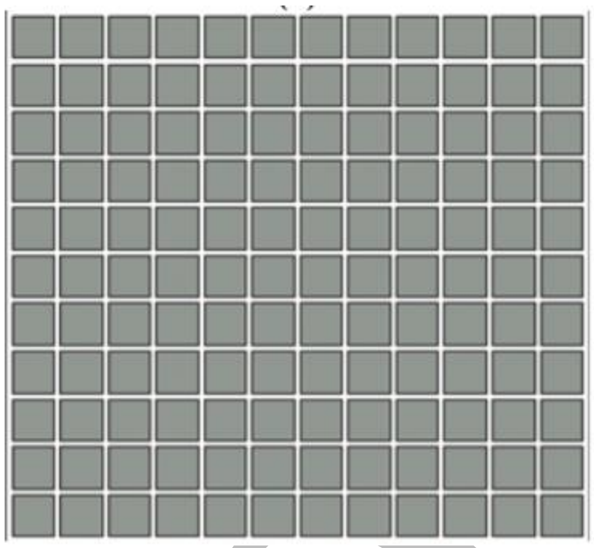


Image Acquisition using sensor strips

The most common sensor is the photodiode constructed of silicon materials and output voltage waveform proportional to light. Using a filter in front of the sensor improves its selectively.

In order to generate a 2-D image using a single sensor, there have to be relative displacements in both the x- and y-directions between the sensor and the area to be imaged. An arrangement used in high precision scanning, where a film negative is mounted onto a drum whose mechanical rotation provides displacement in one dimension. The single sensor is mounted on a lead screw that provides motion in the perpendicular direction. Since mechanical motion can be controlled with high precision, this method is an inexpensive (but slow) way to obtain high-resolution images.

1.8 Image Sampling and Quantization

The output of most sensors is a continuous voltage waveform whose amplitude and spatial behaviour are related to the physical phenomenon being sensed. To create a digital image, it is essential to convert the continuous sensed data into digital form. This involves two processes: sampling and quantization.

1.8.1 Image Sampling

An image may be continuous with respect to the x and y coordinates and also in amplitude. To convert it into digital form we have to sample the function in both coordinates and in amplitudes.

Digitalizing the coordinate values is called sampling.

Digitalizing the amplitude values is called quantization.
There is a continuous image along the line segment AB. To sample this function, we take equally spaced samples along line AB. The location of each sample is given by a vertical tick back (mark) in the bottom part. The samples are shown as block squares superimposed on function the set of these discrete locations gives the sampled function.

In order to form a digital image, the gray level values must also be converted (quantized) into discrete quantities. So, we divide the gray level scale into eight discrete levels ranging from black to white. The vertical tick mark assigns the specific value assigned to each of the eight level values. The continuous gray levels are quantized simply by assigning one of the eight discrete gray levels to each sample. The assignment it made depending on the vertical proximity of a simple to a vertical tick mark.

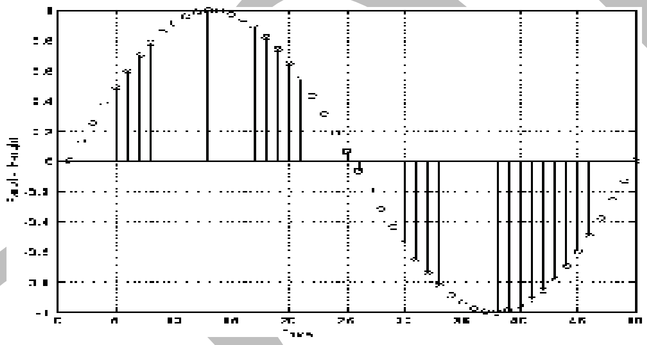


Fig 1.5: Sampling

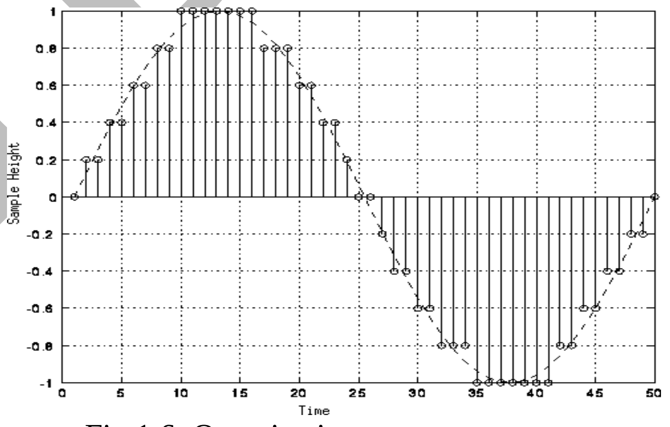


Fig 1.6: Quantization

Example

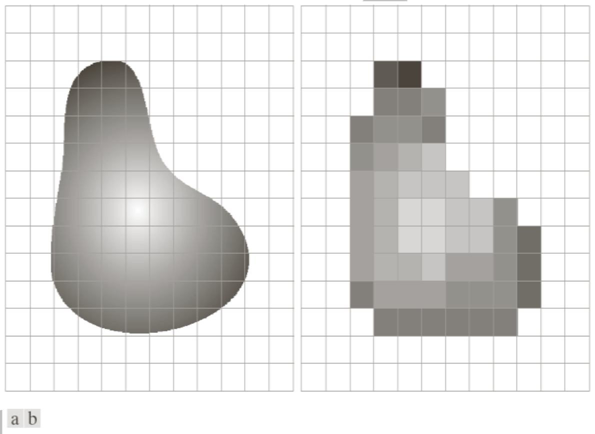

a b

FIGURE 2.17 (a) Continuous image projected onto a sensor array. (b) Result of image sampling and quantization.

Digital Image

A digital image f [m, n] described in a 2D discrete space is derived from an analog image f (x.y) in a 2D continuous space through a sampling process that is frequently referred to as digitization.

The 2D continuous image f (x, y) is divided into N rows and M columns. The intersection of a row and a column is termed a pixel. The value assigned to the integer coordinates [m, n] with {m=0,1, 2…, M-1}and {n=0,1, 2..., N-1}is F [m, n].

1.10 Pixel **Relationships**

Neighbours of a pixel

A pixel p at coordinate (x, y) has four horizontal and vertical neighbour whose coordinate can be given by

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

This set of pixel is called the 4-neighbours of p and is denoted by n4(p), Each pixel is at a unit distance from (x,y) and some of the neighbours of P lie outside the digital image or (x,y) is on the border of the image .

The four-diagonal neighbour of P have co-ordinates

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

And are denoted by nd(p) these points, together with the 4-neighbours are called 8 – neighbours of P denoted by n8(p)

Adjacency

Let V be the set of gray–level values used to define adjacency in a binary image, if V= {1} we are referencing to adjacency of pixel with value. Three types of adjacencies occur

4- Adjacency – two-pixel P and Q with value from V are 4–adjacency if A is in the set n4(P) 8- Adjacency – two-pixel P and Q with value from V are 8–adjacency if A is in the set n8(P) M-adjacency –two-pixel P and Q with value from V are m– adjacency if

- Q is in n4 (p) or
- Q is in nd (q) and the set N4(p) È N4(q) has no pixel whose values

are from V Distance measures

For pixel p, q and z with coordinate (x,y), (s,t) and (v,w) respectively D is a distance function or metric if

D [p.q] ≥ O {D [p.q] = O iff p=q} D

[p.q] = D [p.q] and

D [p.q] ≥ O {D [p.q] +D(q,z)

The Euclidean Distance between p and is
defined as De (p, q) = I y – t I

The D4 Education Distance between p and
is defined as De (p, q) = I y – t I

Review Questions

1. Explain Digital Image Processing briefly?
2. Describe various methods of image sensing and acquisition.
3. Explain pixel and its relationships with its neighbourhood pixels.
4. Write a short note on Image Sampling and Quantization.

References for in detail study

1. www.ImageProcessingplace.com
2. https://nptel.ac.in/courses/106/105/106105032/
3. https://nptel.ac.in/courses/117/105/117105079/

Chapter 2 Intensity Transformations and Spatial filtering

Objective

1. To understand the meaning of spatial domain and apply techniques for intensity-based transformations
2. To know the concept of histogram and use it for image enhancement
3. To understand the mechanics of spatial filtering and use combination methods or enhancing image
4. Use fuzzy techniques to perform spatial filtering methods

2.1 Introduction

The principal objective of enhancement is to process an image so that the result is more suitable than the original image for a specific application. Image enhancement approaches fall into two broad categories

1. Spatial domain methods
2. Frequency domain methods

The chapter mainly focusses on Spatial domain methods. All methods and techniques covered here are for spatial domain.

The term spatial domain refers to the image plane itself and approaches in these categories are based on direct manipulation of pixel in an image.

Spatial domain process is denoted by the expression

g (x, y) =T [f (x, y)]

f (x, y)- input image, T- operator on f, defined over some neighbourhood of f (x, y)

g (x, y)-processed image

The neighbourhood of a point (x, y) can be explained by using as square or rectangular sub image area centred at (x, y).

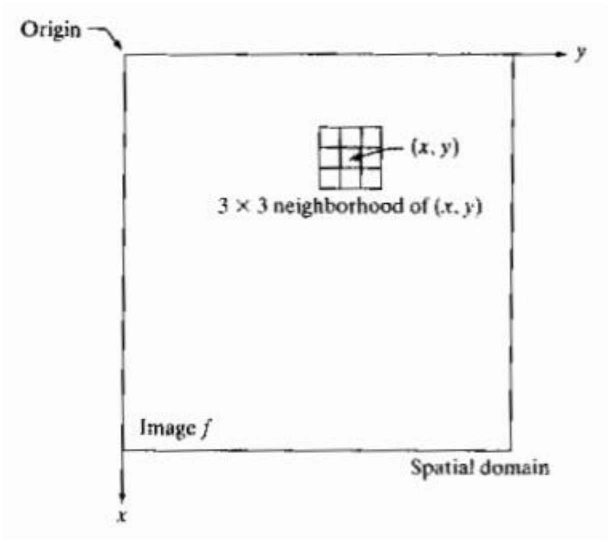


Fig 2.1 A 3 x 3 neighbourhood about a point (x0, y0) in an image. The neighbourhood is moved from pixel to pixel in the image to generate an output image

The center of sub image is moved from pixel to pixel starting at the top left corner. The operator T is applied to each location (x, y) to find the output g at that location. The process utilizes only the pixel in the area of the image spanned by the neighbourhood.

2.2 Basic Intensity Transformation Functions

It is the simplest form of the transformation when the neighbourhood is of size 1 x 1. In this case g depends only on the value of f at (x, y) and T becomes an intensity or gray level transformation function of the form

S=T(r)

r-Denotes the gray level of f (x, y)

s-Denotes the gray level of g (x, y) at any point (x, y)

Because enhancement at any point in an image deepens only on the gray level at that point, technique in this category is referred to as point processing.

2.2.1 Point Processing

Contract stretching -It produces an image of higher contrast than the original one.

The operation is performed by darkening the levels below m and brightening the levels above m in the original image.

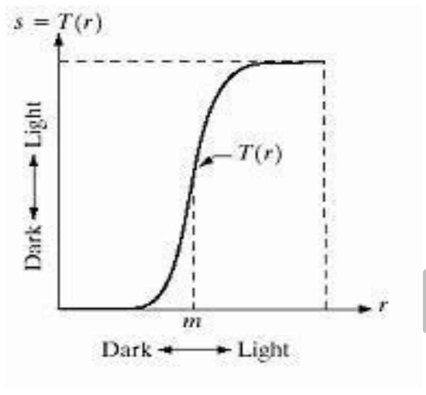


Fig: 2.2 Contrast Stretching

In this technique the value of r below m are compressed by the transformation function into a narrow range of s towards black. The opposite effect takes place for the values of r above m.

Thresholding function – It is a limiting case where T(r) produces a two levels binary image. The values below m are transformed as black and above m are transformed as white.

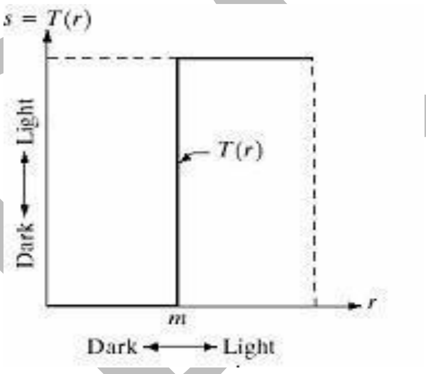


Fig 2.3 Thresholding function

2.2.2 Basic Intensity Level Transformation

These are the simplest image enhancement techniques

2.2.2.1. Image Negative – The negative of in image with gray level in the range [0, L-1] is obtained by using the negative transformation.
The expression of the transformation is

$$s= L-1-r$$

Reverting the intensity levels of an image in this manner produces the equivalent of a photographic negative. This type of processing is practically suited for enhancing white or gray details embedded in dark regions of an image especially when the black areas are dominant in size.

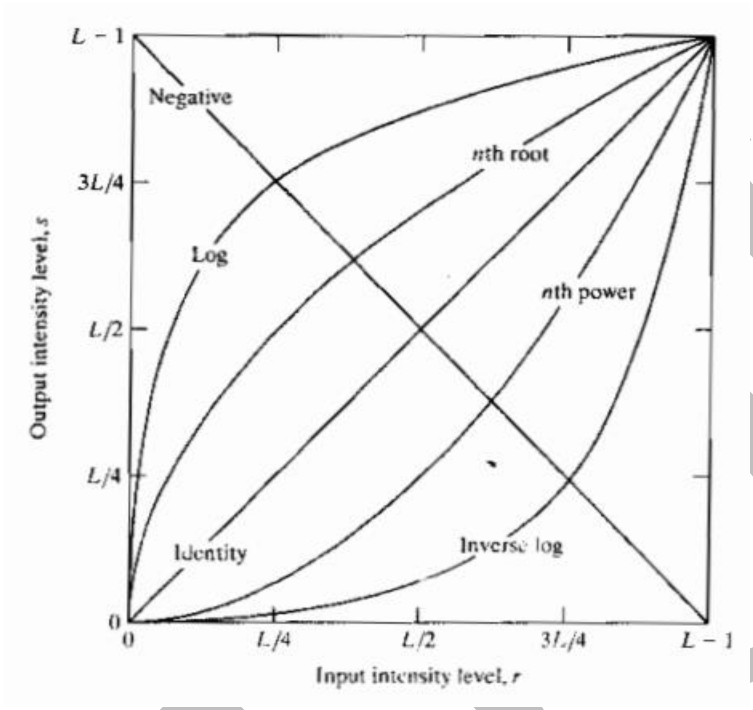


Fig 2.4 Basic intensity transformation functions

2.2.2.2 Log transformations-

The general form of log transform is S=c log(1+R)

Where R >= 0

This transformation maps a narrow range of gray level values in the input image into a wider range of output gray levels. The opposite is true for higher values of input levels. These transformations are used to expand the values of dark pixels in an image while compressing the higher-level values. The opposite is true for inverse log transformation.

The log transformation function has an important characteristic that it compresses the dynamic range of images with large variations in pixel values.

E.g.- Fourier spectrum

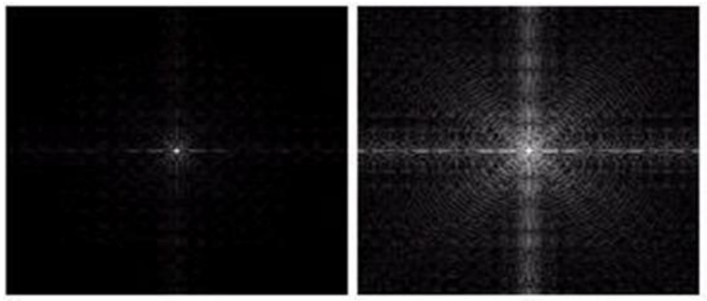


Fig 2.5 a) Fourier Spectrum b) Log transformed Fourier spectrum

2.2.2.3 Power law transformation

Power law transformation has the basic function

S= c r$^{\gamma}$

Where c and γ are positive constants.

Power law curves with fractional values of γ map a narrow range of dark input values into a wider range of output values, with the opposite being true for higher values of input gray levels. We may get various curves by varying values of γ.

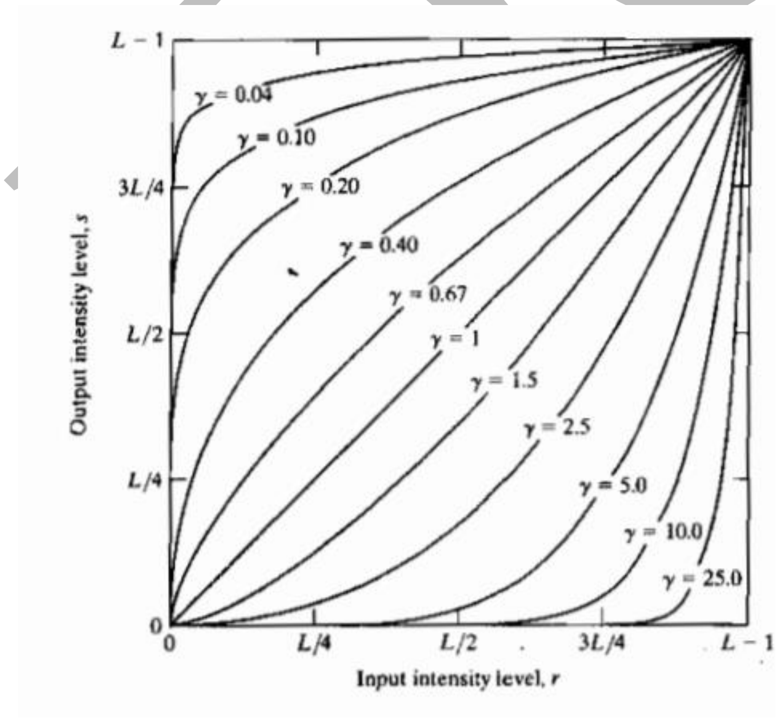


Fig: 2.6 Shape of the curve formed by the gamma equation

A variety of devices used for image capture, printing and display respond according to a power

law. The process used to correct this power law response phenomenon is called gamma correction.

For E.g.-CRT devices have intensity to voltage response that is a power function

Gamma correction is important if displaying an image accurately on a computer screen is of concern. Images that are not corrected properly can look either bleached out or too dark.

Color phenomenon also uses this concept of gamma correction. It is becoming more popular due to use of images over the internet.

It is important in general purpose contract manipulation.

To make an image black we use $\gamma > 1$ and $\gamma < 1$ for white image.

2.2.2.4 Piece wise Linear transformation functions-

The principal advantage of piecewise linear functions is that these functions can be arbitrarily complex. But their specification requires considerably more user input

Contrast Stretching-

It is the simplest piecewise linear transformation function.

We may have various low contrast images and that might result due to various reasons such as lack of illumination, problem in imaging sensor or wrong setting of lens aperture during image acquisition.

The idea behind contrast stretching is to increase the dynamic range of gray levels in the image being processed.

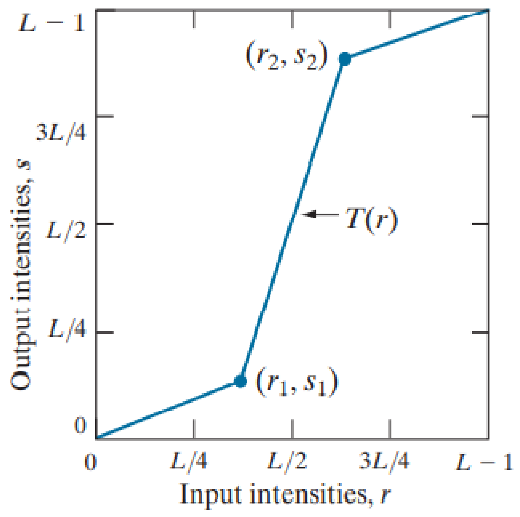


Fig 2.7 Piecewise linear transformation function

The location of points (r1, s1) and (r_2, s_2) control the shape of the curve

a) If $r_1=r_2$ and $s_1=s_2$, the transformation is a linear function that deduces no change in gray levels.

b) If r1=s1, s1=0 , and s2=L-1, then the transformation become a thresholding function that creates a binary image

c) Intermediate values of (r1, s1) and (r2, s2) produce various degrees of spread in the gray value of the output image thus effecting its contract.

Generally, r1≤ r2 and s1 ≤ s2 so that the function is single valued and monotonically increasing

Gray Level Slicing- Highlighting a specific range of gray levels in an image is often desirable
For example, when enhancing features such as masses of water in satellite image and enhancing flaws in x- ray images.

There are two ways

1) This method is to display a high value for all gray level in the range of interest and a low value for all other gray level
2) Second method is to brighten the desired ranges of gray levels but preserve the background and gray level tonalities in the image

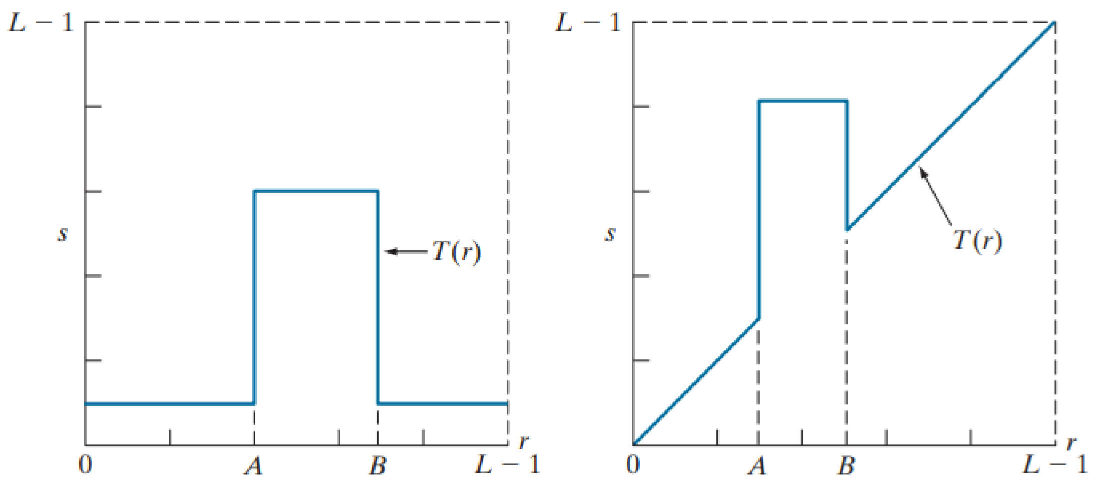


Fig 2.8 a) Highlights the range [A, B] b) Highlights the range [A, B] and preserves background details

2.2.2.5 Bit Plane Slicing

It is important to highlight the contribution made to the total image appearance by specific bits. Suppose that each pixel is represented by 8 bits.
Imagine that an image is composed of eight 1-bit planes ranging from bit plane 0 for the least significant bit to bit plane 7 for the most significant bit. In terms of 8-bit bytes, plane 0 contains all the lowest order bits in the image and plane 7 contains all the high order bits.

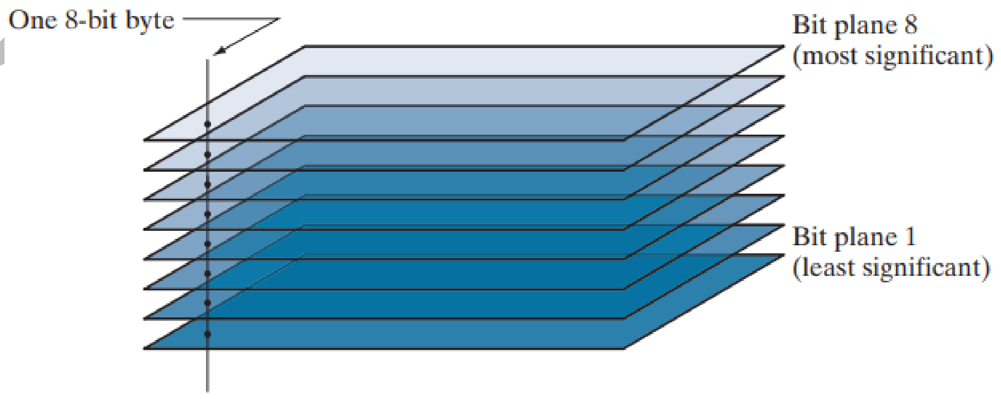


Fig 2.9 Bit-planes of an image

2.3 Histogram Processing

The histogram of a digital image with gray levels in the range [0, L-1] is a discrete function of the form H(rk)=nk

where rk is the kth gray level and nk is the number of pixels in the image having the level rk.
A normalized histogram is given by the equation
P(rk)=nk/n for k=0,1, 2, . . ., L-1
P(rk) gives the estimate of the probability of occurrence of gray level rk. The sum of all components of a normalized histogram is equal to 1.

The histogram plots are simple plots of H(rk)=nk versus rk.

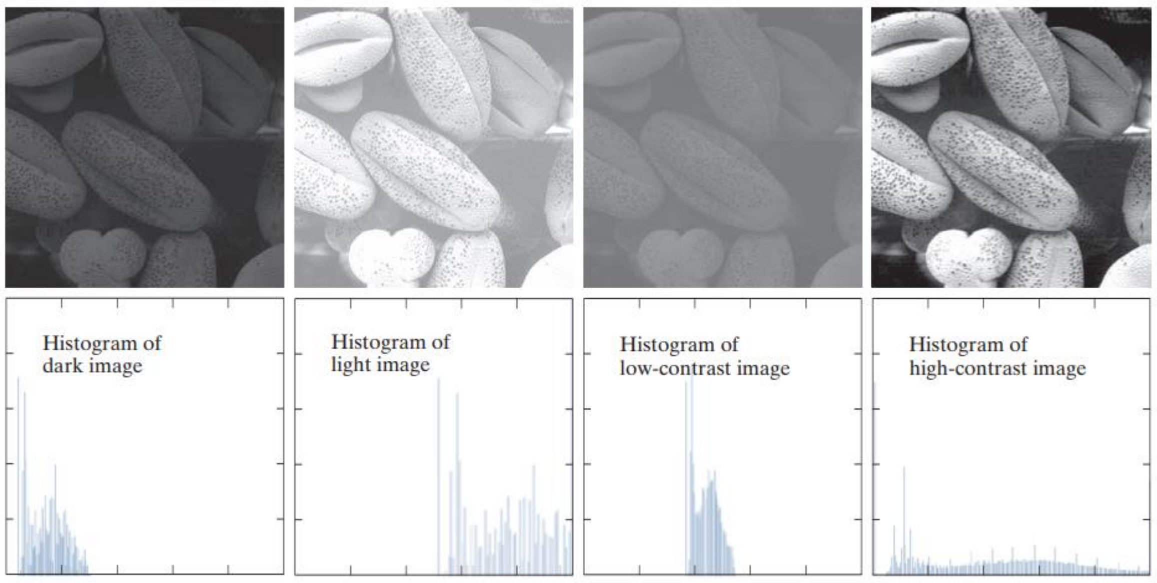


Fig 2.10 Four image types and their corresponding histograms. (a) dark; (b) light; (c) low contrast; d) high contrast

In the dark image the components of the histogram are concentrated on the low (dark) side of the gray scale. In case of bright image, the histogram components are biased towards the high side of the gray scale.
The histogram of a low contrast image will be narrow and will be centered towards the middle of the gray scale.
The components of the histogram in the high contrast image cover a broad range of the gray scale. The net effect of this will be an image that shows a great deal of gray levels details and has high dynamic range.

2.3.1 Histogram Equalization

Histogram equalization is a common technique for enhancing the appearance of images. Suppose we have an image which is predominantly dark. Then its histogram would be skewed

towards the lower end of the grey scale and all the image detail are compressed into the dark end of the histogram. If we could ‘stretch out’ the grey levels at the dark end to produce a more uniformly distributed histogram then the image would become much clearer.
Let there be a continuous function with r being gray levels of the image to be enhanced.
The range of r is [0, 1] with r=0 repressing black and r=1 representing white.
The transformation function is of the form.

$$S=T(r) \qquad \text{where } 0<r<1$$

It produces a level s for every pixel value r in the original image. the transformation function is assumed to fulfill two condition T(r)) is single valued and monotonically increasing in the interval 0<T(r)<1 for 0<r<1

The transformation function should be single valued so that the inverse transformations should exist. Monotonically increasing condition preserves the increasing order from black to white in the output image. The second conditions guarantee that the output gray levels will be in the same range as the input levels.
The gray levels of the image may be viewed as random variables in the interval [0.1]
The most fundamental descriptor of a random variable is its probability density function (PDF) Pr(r) and Ps(s) denote the probability density functions of random variables r and s respectively. Basic results from an elementary probability theory states that if Pr(r) and Tr are known and T-1 (s) satisfies conditions (a), then the probability density function Ps(s) of the transformed variable s is given by the formula-
Thus, the PDF of the transformed variable s is the determined by the gray levels PDF of the input image
and by the chosen transformations function

$$P_s(s) = P_r(r)\frac{dr}{ds},$$

A transformation function of a particular importance in image processing

$$s = T(r) = \int_0^r P_r(w)dw.$$

This is the cumulative distribution function of r
Using this definition of T we see that the derivative of s with respect to r is

$$\frac{ds}{dr} = P_r(r).$$

Substituting it back in the expression for Ps we may get

$$P_s(s) = P_r(r)\frac{1}{P_r(r)} = 1\frac{1}{\cdot(r)} = 1$$

An important point here is that Tr depends on Pr(r) but the resulting Ps(s) always is uniform, and independent of the form of P(r).
For discrete values we deal with probability and summations instead of probability density functions and integrals.

The probability of occurrence of gray levels rk in an image as approximated

Pr(r)=nk/N

N is the total number of the pixels in an image. nk is the number of the pixels that have gray level rk. L is the total number of possible gray levels in the image. The discrete transformation function is given by

$$s_k = T(r_k) = \sum_{i=0}^{k} \frac{n_i}{N}$$
$$= \sum_{i=0}^{k} P_r(r_i).$$

Thus, a processed image is obtained by mapping each pixel with levels rk in the input image into a corresponding pixel with level sk in the output image.

A plot of Pr (rk) versus rk is called a histogram. The transformation function given by the above equation is the called histogram equalization or linearization. Given an image the process of histogram equalization consists of implementing the transformation function which is based on information that can be extracted directly from the given image, without the need for further parameter specification.

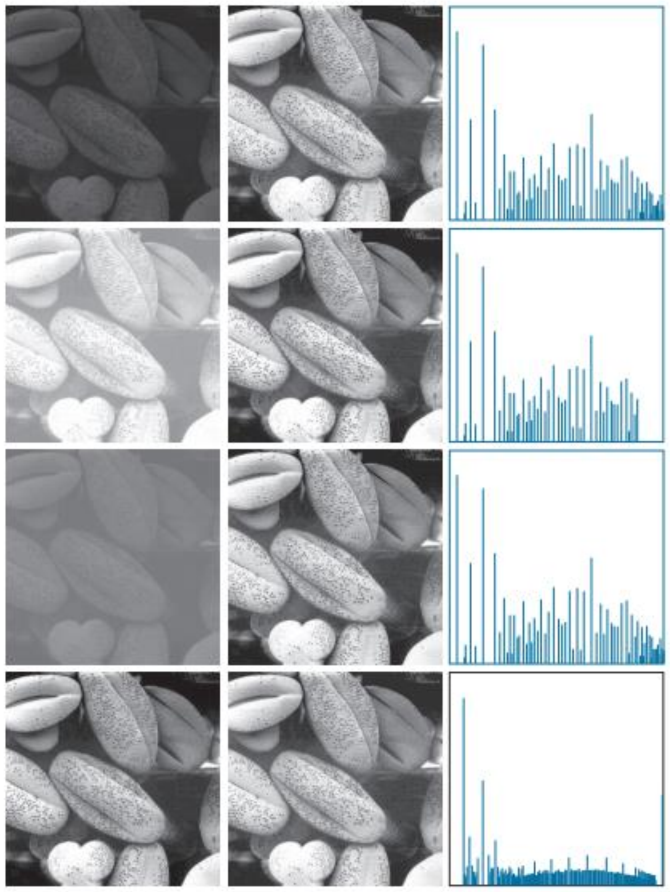


Fig 2.11 Left column- original images, center column- histogram equalized images and last column- histogram of histogram equalized images.

Equalization automatically determines a transformation function that seeks to produce an output image that has a uniform histogram. It is a good approach when automatic enhancement is needed.

2.3.2 Histogram Matching (Specification)

In some cases, it may be desirable to specify the shape of the histogram that we wish the processed image to have.
Histogram equalization does not allow interactive image enhancement and generates only one result: an approximation to a uniform histogram. Sometimes we need to be able to specify particular histogram shapes capable of highlighting certain gray-level ranges. The method used to generate a processed image that has a specified histogram is called histogram matching or histogram specification.
Algorithm

1. Compute sk=Pf (k), k = 0, …, L-1, the cumulative normalized histogram of f .
2. Compute G(k), k = 0, …, L-1, the transformation function, from the given histogram hz

3. Compute G-1(sk) for each k = 0, …, L-1 using an iterative method (iterate on z), or in effect, directly compute G-1(Pf (k))
4. Transform f using G-1(Pf (k))

Global and Local enhancement

In earlier methods pixels were modified by a transformation function based on the gray level of an entire image. It is not suitable when enhancement is to be done in some small areas of the image. This problem can be solved by local enhancement where a transformation function is applied only in the neighborhood of pixels in the interested region.
Define square or rectangular neighborhood (mask) and move the center from pixel to pixel. For each neighborhood

1) Calculate histogram of the points in the neighbourhood
2) Obtain histogram equalization/specification function
3) Map gray level of pixel centered in neighbourhood
4) The center of the neighbourhood region is then moved to an adjacent pixel location and the procedure is repeated.

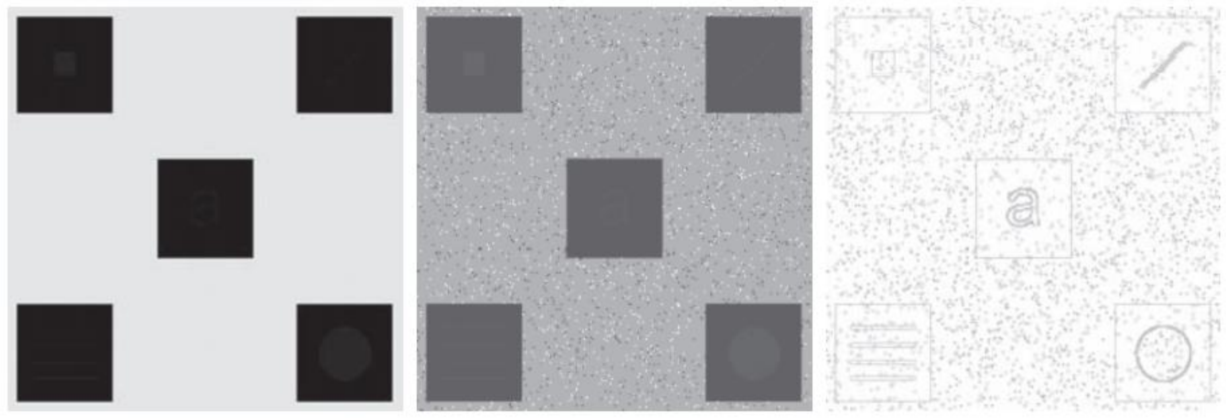
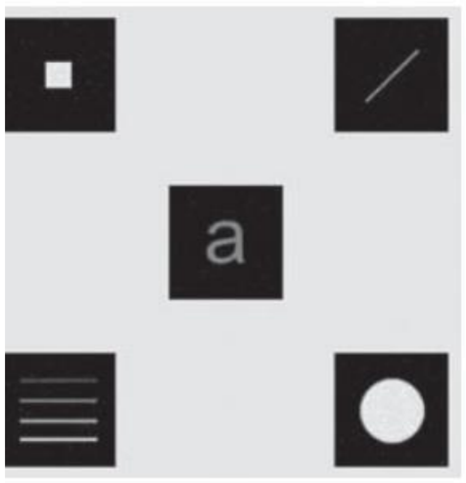


Fig 2.12a) Original Image b) Image obtained after global histogram equalization and c) after local histogram equalization.



d) Result of local enhancement based on local histogram

2.4 Fundamentals of Spatial filtering

1. Spatial filtering modifies an image by replacing the value of each pixel by a function of the values of the pixel and its neighbours.
2. If the operation performed on the image pixels is linear, then the filter is called linear spatial filter. Otherwise, the filter is a non-linear filter.

Spatial filtering is an example of neighborhood operations, in this the operations are done on the values of the image pixels in the neighborhood and the corresponding value of a sub image that has the same dimensions as of the neighborhood.
This sub image is called a **filter, mask, kernel, template or window**; the values in the filter sub image are referred to as coefficients rather than pixel. Spatial filtering operations are performed directly on the pixel values (amplitude/gray scale) of the image. The process consists of moving the filter mask from point to point in the image. At each point (x, y) the response is calculated using a predefined relationship.

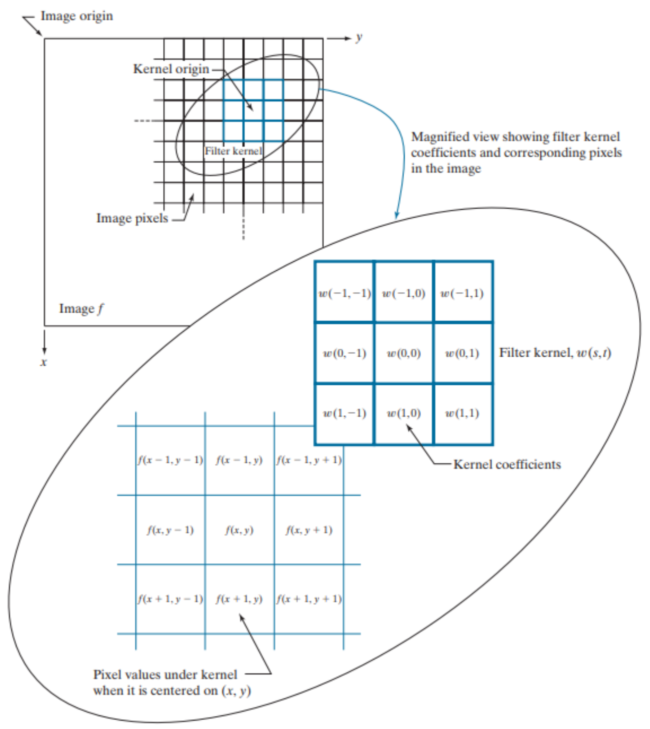


Fig 2.13 The mechanics of spatial filtering using a 3 x 3 kernel

For linear spatial filtering the response is given by a sum of products of the filter coefficient and the corresponding image pixels in the area spanned by the filter mask.

The results F (x, y) of linear filtering with the filter mask at point (x, y) in the image is given by

w (-1, -1) f (x-1, y-1) + w (-1,0) f (x-1, y) + ………...+ w (1, 1) f (x + 1, y +1)

The coefficient w (0,0) coincides with image value f (x, y) indicating that mask it centered at (x, y) when the computation of sum of products takes place.
For a mask of size M x N, we assume **m=2a+1 and n=2b+1**, where a and b are nonnegative integers. It shows that all the masks are **of odd size**.
In the general linear filtering of an image of size f of size M*N with a filter mask of size m*m is given by the expression.

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

Where a= (m-1)/2 and b = (n-1)/2
To generate a complete filtered image this equation must be applied for x=0, 1, 2, -----M-1 and y=0,1,2---, N-1. Thus, the mask processes all the pixels in the image.
The process of linear filtering is similar to frequency domain concept called convolution. For this reason, linear spatial filtering often is referred to as convolving a mask with an image. Filter mask are sometimes called convolution mask.

Padding the image by adding rows and columns of o's & of padding by replicating rows and columns when the centre of the filter approaches the border of the image.

2.4.1Smoothing Spatial Filters(lowpass)

These filters are used for blurring and noise reduction blurring is used in preprocessing steps such as removal of small details from an image prior to object extraction and bridging of small gaps in lines or curves.

Smoothing Linear Filters

The output of a smoothing liner spatial filter is simply the average of the pixel contained in the neighborhood of the filter mask. These filters are also called averaging filters or low pass filters. Smoothing filters are used in combination with other image enhancement techniques.

The operation is performed by replacing the value of every pixel in the image by the average of the gray levels in the neighborhood defined by the filter mask. This process reduces sharp transitions in gray levels in the image.

A major application of smoothing is noise reduction but because edges are also provided using sharp transitions so smoothing filters have the undesirable side effect that they blur edges. It also removes an effect false contouring which is caused by using insufficient number of gray levels in the image.
Irrelevant details can also be removed by filters, irrelevant means which are not of interest.

A spatial averaging filter in which all coefficients are equal is sometimes referred to as a "box filter".

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

A weighted average filter is the one in which pixel are multiplied by different coefficients.

Order Statistics Filter

These are nonlinear spatial filter whose response is based on ordering of the pixels contained in the image area compressed by the filter and the replacing the value of the center pixel with value determined by the ranking result.

The best example of this category is median filter. In this filter the values of the center pixel are replaced by median of gray levels in the neighborhood of that pixel. Median filters are popular because, for certain types of random noise, they provide excellent noise-reduction capabilities, with considerably less blurring than linear smoothing filters.

These filters are particularly effective in the case of impulse or salt and pepper noise. It is called so because of its appearance as white and black dots superimposed on an image.

In order to perform median filtering at a point in an image, we first sort the values of the pixel in the question and its neighbors, determine their median and assign this value to that pixel.

Low pass filtering is used for shadow correction, region extraction if used with thresholding.

2.4.2 Sharpening Spatial Filters (High pass)

The principal objective of sharpening is to highlight fine details in an image or to enhance details that have been blurred either in error or as a natural effect of particular method for image acquisition.

The applications of image sharpening range from electronic printing and medical imaging to industrial inspection and autonomous guidance in military systems.

As smoothing can be achieved by integration, sharpening can be achieved by spatial differentiation. The strength of response of derivative operator is proportional to the degree of discontinuity of the image at that point at which the operator is applied. Thus, image differentiation enhances edges and other discontinuities and deemphasizes the areas with slow varying grey levels.

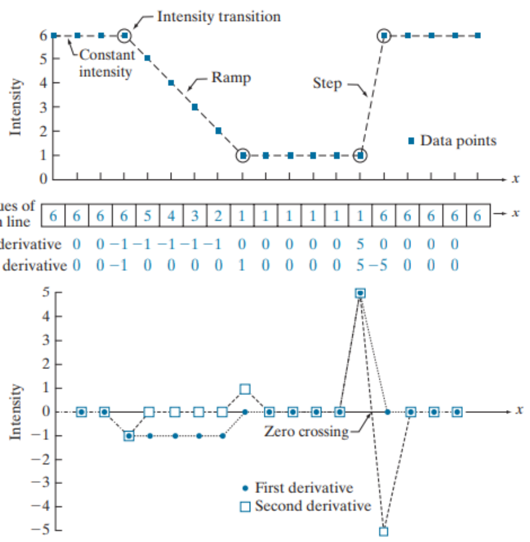
It is a common practice to approximate the magnitude of the gradient by using absolute values instead of square and square roots.

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

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

A second order derivative as the difference

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



Values of scan line	6	6	6	6	5	4	3	2	1	1	1	1	1	1	6	6	6	6	6
1st derivative		0	0	−1	−1	−1	−1	−1	0	0	0	0	0	5	0	0	0	0	
2nd derivative		0	0	−1	0	0	0	0	1	0	0	0	0	5	−5	0	0	0	

Fig 2.14 A scan line from an image showing ramp, step and constant segments with its first and second order derivative

2.4.3.Using LAPLACIAN- Second derivative for image sharpening

The second order derivative is calculated using Laplacian. It is simplest isotropic filter. Isotropic filters are the ones whose response is independent of the direction of the image to which the operator is applied.

The Laplacian for a two-dimensional function f (x, y) is defined as

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

In the x-direction

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

In the y-direction

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

From the above equation, the discrete Laplacian of two variable is

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

The equation can be represented using any one of the following masks

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

Fig 2.15 Laplacian kernels

Laplacian highlights gray-level discontinuities in an image and deemphasize the regions of slow varying gray levels. This makes the background a black image. The background texture can be recovered by adding the original and Laplacian images.

$$g(x, y) = \begin{cases} f(x, y) - \nabla^2 f(x, y) & \text{if the center coefficient of the Laplacian mask is negative} \\ f(x, y) + \nabla^2 f(x, y) & \text{if the center coefficient of the Laplacian mask is positive.} \end{cases}$$

$$\begin{aligned} g(x, y) &= f(x, y) - [f(x+1, y) + f(x-1, y) \\ &\quad + f(x, y+1) + f(x, y-1)] + 4f(x, y) \\ &= 5f(x, y) - [f(x+1, y) + f(x-1, y) \\ &\quad + f(x, y+1) + f(x, y-1)]. \end{aligned}$$

The strength of the response of a derivative operator is propositional to the degree of discontinuity of the image at that point at which the operator is applied. Thus, image differentiation enhances eddies and other discontinuities and deemphasizes areas with slowly varying gray levels.

The derivative of a digital function is defined in terms of differences. Any first derivative definition

(1) Must be zero in flat segments (areas of constant gray level values)

(2) Must be nonzero at the onset of a gray level step or ramp

(3) Must be nonzero along ramps.

Any second derivative definition

(1) Must be zero in flat areas

(2) Must be nonzero at the onset and end of a gray level step or ramp

(3) Must be zero along ramps of constant slope.

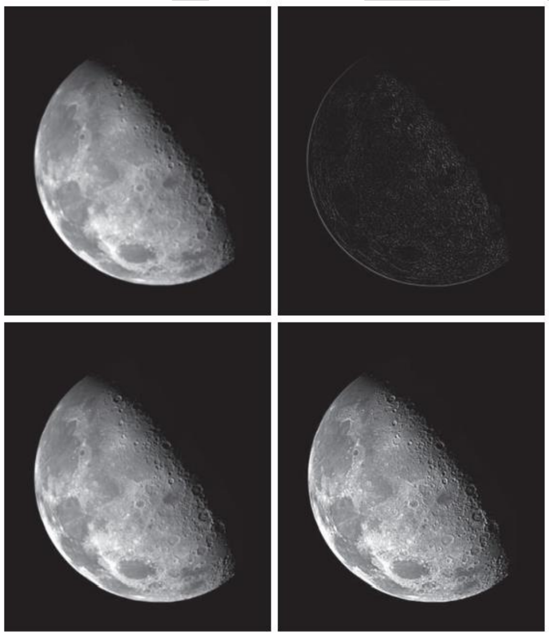


Fig 2.16 Example of sharpening from blurred image to sharpened image

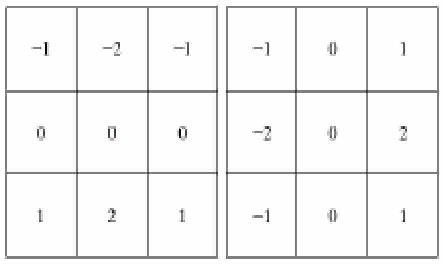
It is common practice to approximate the magnitude of the gradient by using absolute values instead of squares and square roots.

Roberts Goss gradient operators
For implementing the gradient operators

Z1	Z2	Z3
Z4	Z5	Z6
Z7	Z8	Z9

-1	0
0	1

The smallest filter mask is size 3x3 mask



−1	−2	−1
0	0	0
1	2	1

−1	0	1
−2	0	2
−1	0	1

The difference between third and first row a 3x3 mask approximates the derivate in the x-direction and difference between the third and first column approximates the derivative in y-direction. These masks are called sobel operators.

2.4.4 Unsharp masking and highboost filtering

Unsharp masking means subtracting a blurred version of an image form the image itself.
It consists of following steps

1. Blur the original image
2. Subtract the blurred image from the original
3. Add the mask to the original

Letting $\bar{f}(x, y)$ denote the blurred image, the mask in equation form is

$$g_{\text{mask}}(x, y) = f(x, y) - \bar{f}(x, y)$$

Then we add a weighted portion of the mask back to the original image

$$g(x, y) = f(x, y) + k\, g_{\text{mask}}(x, y)$$

where we included a weight, k, k () ≥ 0, for generality. When k = 1 we have unsharp masking, as defined above. When k > 1, the process is referred to as highboost filtering. Choosing k < 1 reduces the contribution of the unsharp mask

2.5 Highpass, Band reject and Band pass filters from Lowpass filters

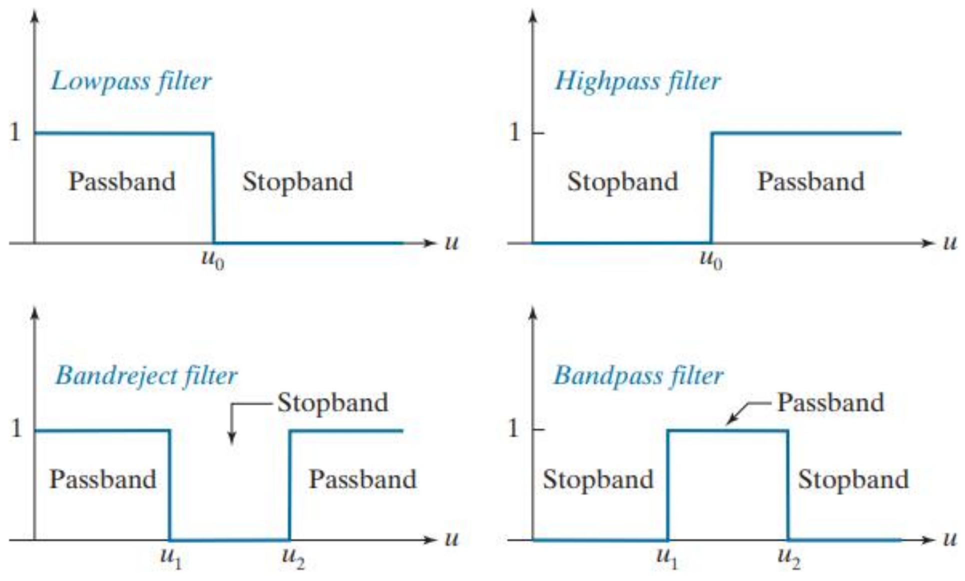


Fig 2.17 Low pass, Highpass ,Bandreject and Bandpass filters

Spatial and frequency-domain linear filters are classified into four broad categories: lowpass and highpass filters, bandpass and bandreject filters.

Lowpass filter - A lowpass filters attenuate or delete high frequencies, while passing low frequencies.

Highpass filter - A highpass filter deletes or attenuates all frequencies below a cut-off value, u_0, and passes all frequencies above this value. A highpass filter transfer function is obtained by subtracting a lowpass function from 1. This operation is in the frequency domain.

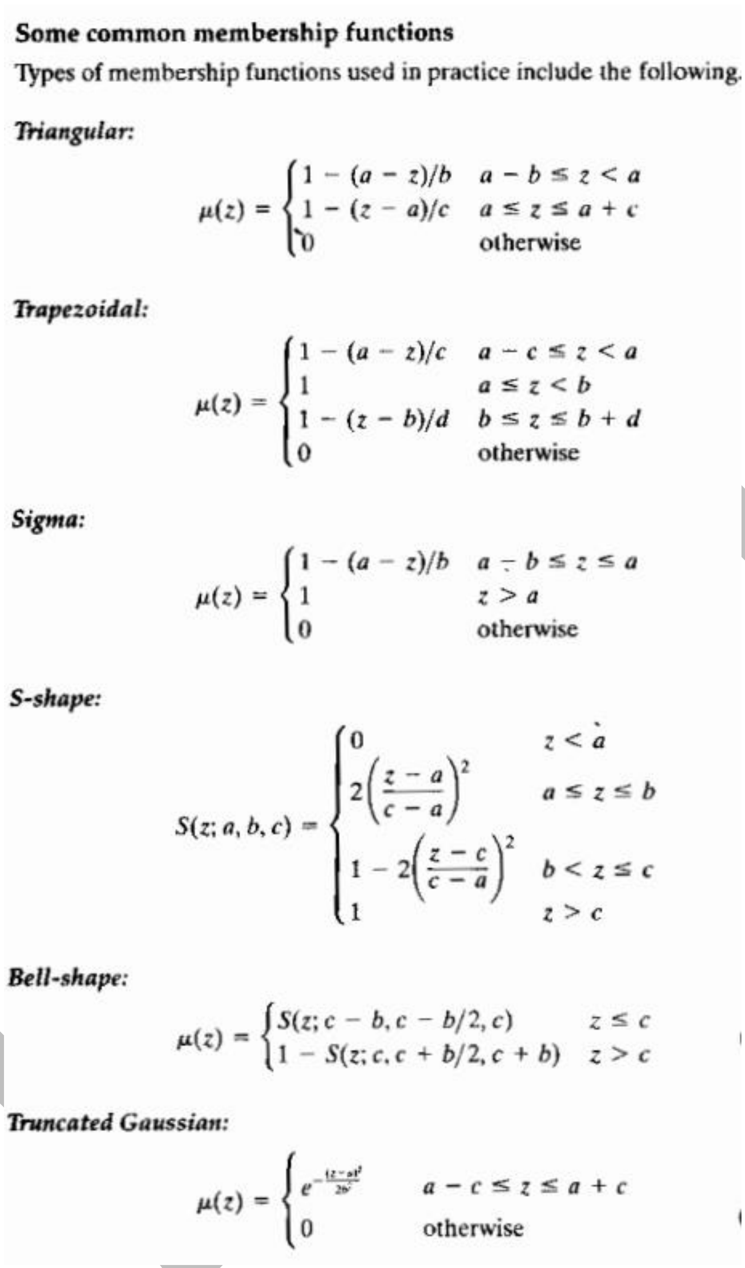
Bandreject filter- These filters can be constructed from the sum of a lowpass and a highpass function with different cut-off frequencies. They are known as notch filters.

Bandpass filter- The bandpass filter transfer function can be obtained by subtracting the bandreject function from 1 (a unit impulse in the spatial domain).

2.6 Spatial Intensity transformation and filtering methods using fuzzy techniques

Fuzzy sets and its applications are very useful when it comes to image processing specially to spatial intensity transformation and filtering. Fuzzy logic proves more beneficial and flexible than classical set theory.

Fuzzy logic deals with degree of membership in a set and never probabilistic. They find application in situation characterized by vagueness and imprecision.



Some common membership functions

Types of membership functions used in practice include the following.

Triangular:

$$\mu(z) = \begin{cases} 1 - (a - z)/b & a - b \le z < a \\ 1 - (z - a)/c & a \le z \le a + c \\ 0 & \text{otherwise} \end{cases}$$

Trapezoidal:

$$\mu(z) = \begin{cases} 1 - (a - z)/c & a - c \le z < a \\ 1 & a \le z < b \\ 1 - (z - b)/d & b \le z \le b + d \\ 0 & \text{otherwise} \end{cases}$$

Sigma:

$$\mu(z) = \begin{cases} 1 - (a - z)/b & a - b \le z \le a \\ 1 & z > a \\ 0 & \text{otherwise} \end{cases}$$

S-shape:

$$S(z; a, b, c) = \begin{cases} 0 & z < a \\ 2\left(\frac{z - a}{c - a}\right)^2 & a \le z \le b \\ 1 - 2\left(\frac{z - c}{c - a}\right)^2 & b < z \le c \\ 1 & z > c \end{cases}$$

Bell-shape:

$$\mu(z) = \begin{cases} S(z; c - b, c - b/2, c) & z \le c \\ 1 - S(z; c, c + b/2, c + b) & z > c \end{cases}$$

Truncated Gaussian:

$$\mu(z) = \begin{cases} e^{-\frac{(z-a)^2}{2b^2}} & a - c \le z \le a + c \\ 0 & \text{otherwise} \end{cases}$$

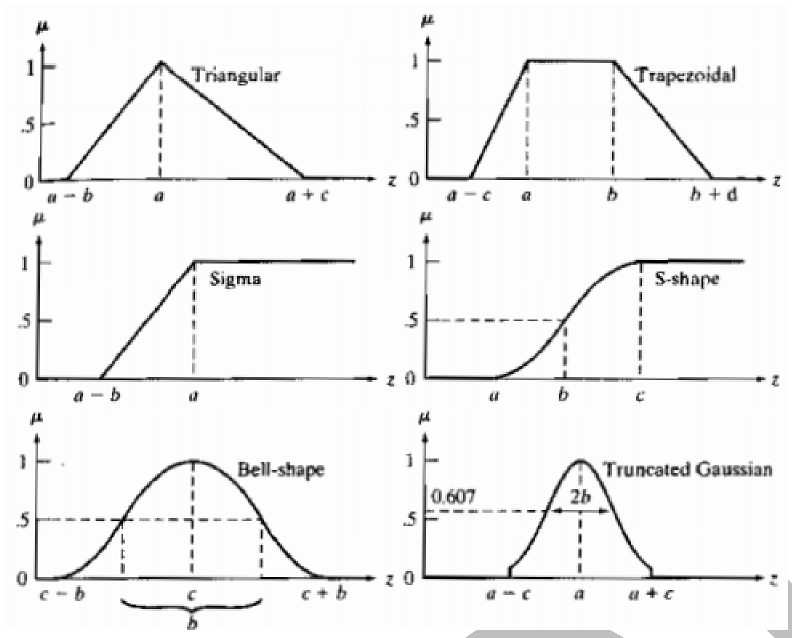


Fig: 2.18 Membership functions with respect to their equations

The principal steps followed in the application of rule-based fuzzy logic

1. Fuzzify the inputs
2. Perform any required fuzzy logical operations
3. Apply an implication method
4. Apply an aggregation method to the fuzzy sets from step 3
5. Defuzzify the final output fuzzy set

The problem specific knowledge can be formalized with the following IF-THEN fuzzy rules. Consider the following rules for a fruit.

R1: IF the color is green, THEN the fruit is verdant
OR
R2: IF the color is yellow, THEN the fruit is half-mature
OR
R3: IF the color is red, THEN the fruit is mature

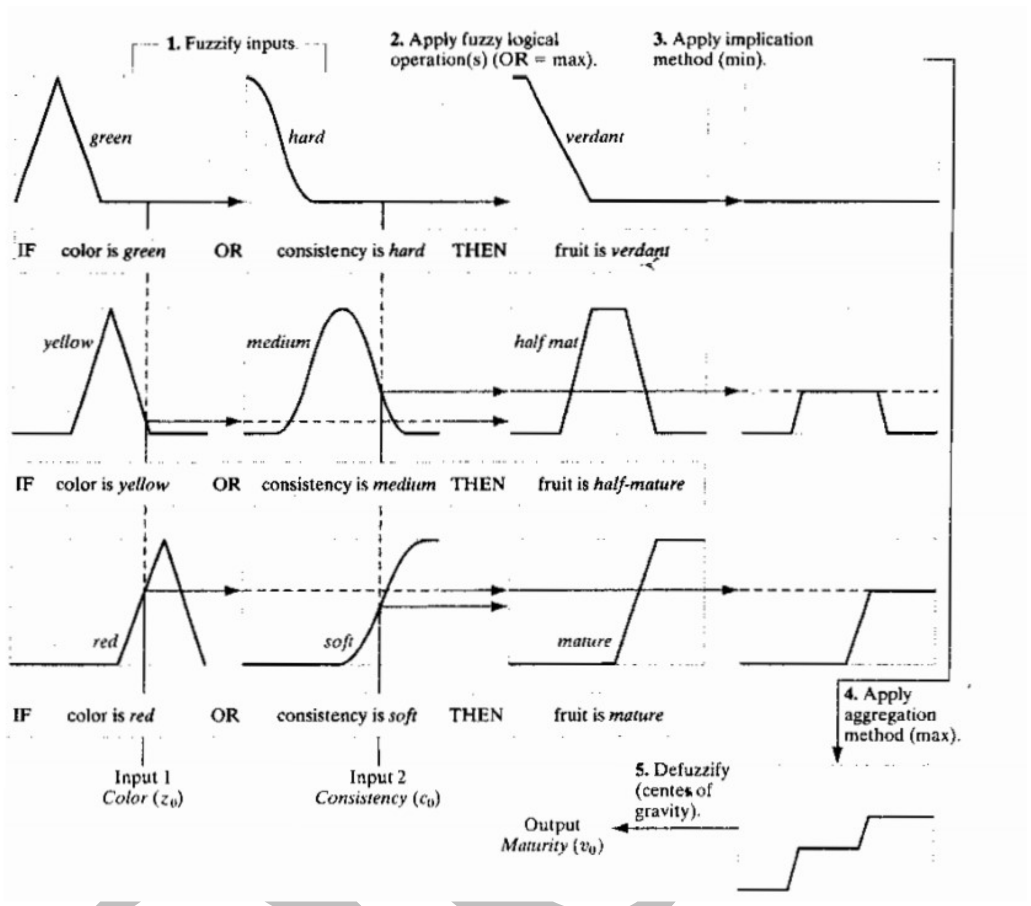

1. Fuzzify inputs.
2. Apply fuzzy logical operation(s) (OR = max).
3. Apply implication method (min).
green
hard
verdant
IF color is green OR consistency is hard THEN fruit is verdant
yellow
medium
half mat
IF color is yellow OR consistency is medium THEN fruit is half-mature
red
soft
mature
IF color is red OR consistency is soft THEN fruit is mature
4. Apply aggregation method (max).
Input 1 Color (z_0)
Input 2 Consistency (c_0)
5. Defuzzify (center of gravity).
Output Maturity (v_0)

Fig 2.19 Five basic steps used to implement the fuzzy rule-based system

Using Fuzzy sets for Intensity Transformation

Consider a general problem of contrast enhancement. The following rules can be defined for enhancing the contrast in a gray-scale image.

R1: IF a pixel is dark, THEN make it darker

R2: IF a pixel is gray, THEN make it gray

R3: IF a pixel is bright, THEN make it brighter

The concept of dark, gray and bright can be expressed and evaluated using membership functions. Fuzzy image processing is computationally intensive because the entire process of fuzzification, processing the antecedents of all rules, implication, aggregation and defuzzification must be applied to every pixel in the input image.

The output of these fuzzy rules whose output is interpreted as constant intensities are considered as singletons which means their membership functions are constant. Singletons greatly reduce the computational requirements in fuzzy image processing. Similarly fuzzy rules can be applied for spatial filtering. When applying fuzzy sets to spatial filtering, the basic approach is to define the neighbourhood properties, that capture the essence of filter are to detect.

Review Questions

1. Explain spatial domain processing.
2. Describe point processing, contrast enhancement.
3. What are the basic intensity transformations?
4. What is filtering? What do you understand by the term spatial filtering?
5. Explain histogram equalization.
6. Write a short note on smoothing filters.
7. Write a short note on sharpening filters.
8. Explain the use of fuzzy techniques in image processing.

References

1. Digital Image Processing, third edition by Gonzalez and Woods.
2. www.Imageprocessingplace.com
3. Refer nptel courses on digital image processing for further detail study.

UNIT II

Chapter 3 Filtering in the Frequency Domain

3.1 Introduction

Fourier transform named after the French mathematician Jean Baptiste Joseph Fourier is one of the most prominent transforms used in Image processing.

Any function that periodically repeats itself can be expressed as a sum of sines and cosines of different frequencies each multiplied by a different coefficient, this sum is called Fourier series. Even the functions which are non-periodic but whose area under the curve if finite can also be represented in such form; this is now called Fourier transform.

A function represented in either of these forms and can be completely reconstructed via an inverse process with no loss of information.

3.2 Sampling and Fourier transform of sampled functions

Continuous function is converted into a sequence of discrete values before they can be processed in a computer, which requires sampling and quantization.

Consider a continuous function f(t) to be sampled at uniform intervals, ΔT of the independent variable t.
The simplest way to sampling is to multiply f(t) by a sampling function equal to the train of impulses ΔT units apart. The equation of the sampled function is given by

$$\tilde{f}(t) = f(t)s_{\Delta T}(t) = \sum_{n=-\infty}^{\infty} f(t)\delta(t - n\Delta T)$$

The Fourier Transform of sampled function

Let F(u) denote the Fourier transform of a continuous function f(t). The sampled function is a product of f(t) and an impulse train.
The Fourier transform of the product of the two functions in the spatial domain is the convolution of the transforms of the two functions in the frequency domain.

The Fourier transform of the sampled function is given by

$$\begin{aligned}\tilde{F}(\mu) &= \Im\left\{\tilde{f}(t)\right\} = \Im\{f(t)s_{\Delta T}(t)\} \\ &= (F \star S)(\mu)\end{aligned}$$

where, the given equation below is the Fourier transform of the impulse train.

$$S(\mu) = \frac{1}{\Delta T}\sum_{n=-\infty}^{\infty} \delta\left(\mu - \frac{n}{\Delta T}\right)$$

3.3 Discrete Fourier Transform of one variable

1-D Fourier Transformation and its Inverse

Let f(x) be a continuous function of real variable x. The Fourier transform of f(x) is

$$\Im\{f(x)\} = F(u) = \int_{-\infty}^{\infty} f(x)\exp[-j2\pi ux]dx$$

Where j = $\sqrt{-1}$

F(u) is composed of an infinite sum of sine and cosine terms and each value of u determines the frequency of its corresponding sine-cosine pair where u is a frequency variable.

Given F(u), f(x) can be obtained by the inverse Fourier transform

$$\Im^{-1}\{F(u)\} = f(x)$$

$$= \int_{-\infty}^{\infty} F(u)\exp[j2\pi ux]du$$

3.4 Discrete Fourier Transform of two variables

2-Dimensional Fourier transform

Images being 2-dimensional functions, we need to define a 2-D Fourier transform.

Fourier and Inverse Fourier transform of a two variable continuous function is given by

$$\Im\{f(x,y)\} = F(u,v) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)\exp[-j2\pi(ux+vy)]dxdy$$

$$\Im^{-1}\{F(u,v)\} = f(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} F(u,v)\exp[j2\pi(ux+vy)]dudv$$

Where u, v are frequency variables.

Since the work is on digital images, rather than continuous variables discrete Fourier transform is more appropriate

1-D Discrete Fourier Transform

The discrete Fourier transform of one variable f(x) is given by

$$F(u) = \frac{1}{M}\sum_{x=0}^{M-1} f(x)\exp[-j2\pi ux/M]$$

where u = 0,1, 2……, M-1 and

$$f(x) = \sum_{u=0}^{M-1} f(u)\exp[j2\pi ux/M]$$

Where x = 0,1, 2……, M – 1

To compute F(u) we substitute u=0 in the exponential term and sum for all values of x.

Repeating for all M values of u, it takes M*M summations and multiplications to compute discrete fourier transform

$$F(u) = \frac{1}{M}\sum_{x=0}^{M-1} f(x)\exp[-j2\pi ux/M]$$

Where u = = 0,1, 2……, M-1

For discrete functions, the fourier transform and its inverse exist always.

F(u) is complex, F(u) = R(u) + j I (u)

Where R is real and I is imaginary

The Fourier transform of a real function is generally complex and polar coordinates are used.

Magnitude or spectrum of the Fourier transform is given by

$$|F(u)| = \left[R^2(u) + I^2(u)\right]^{1/2}$$

Magnitude specifies how much of a certain frequency component is present

Phase angle or phase spectrum of the Fourier transform is given by

$$\phi(u) = \tan^{-1}\left[\frac{I(u)}{R(u)}\right]$$

The phase specifies where the frequency component is in the images

The square of the spectrum is referred to as the power spectrum of f(x) (spectral density).

2-D Discrete Fourier transform

The Fourier transform of a 2D discrete function (image) f (x, y) of size Mx N is given by:

$$F(u,v) = \frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)e^{-j2\pi(ux/M+vy/N)}$$

where u = 0,1, 2……, M-1 and v = 0,1, 2……, N-1

The inverse 2-D Discrete Fourier transform is given by

$$f(x,y) = \sum_{u=0}^{M-1}\sum_{v=0}^{N-1} F(u,v) e^{j2\pi(ux/M+vy/N)}$$

where x = = 0,1, 2……, M – 1 and y = 0,1, 2……, N – 1

The 2D Fourier Spectrum, Phase Spectrum and Power Spectrum can be respectively denoted by

$$|F(u,v)| = \left[R^2(u,v) + I^2(u,v)\right]^{1/2}$$

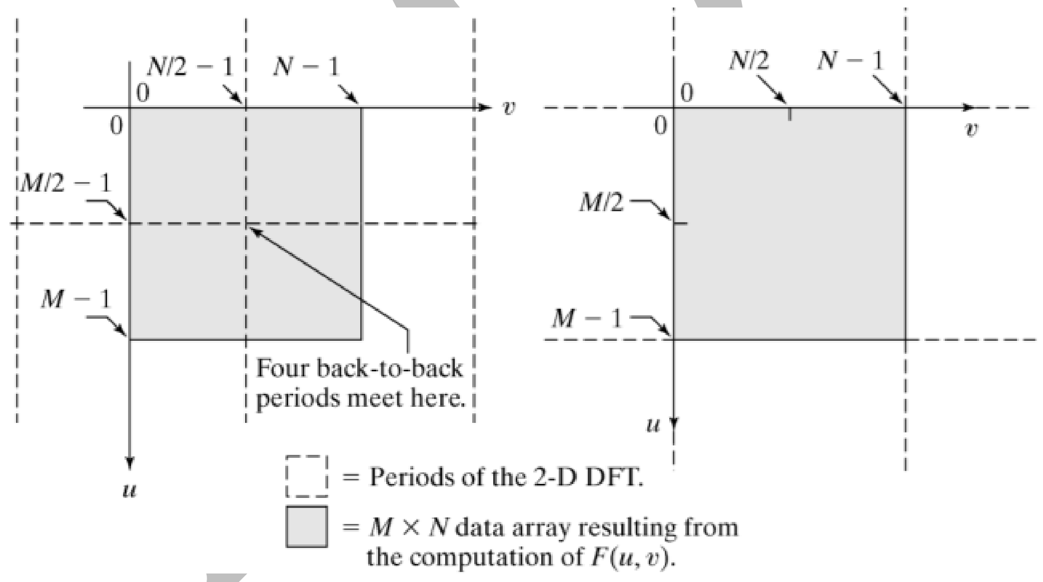
$$\phi(u,v) = \tan^{-1}\left[\frac{I(u,v)}{R(u,v)}\right]$$

$$P(u,v) = |F(u,v)|^2 = R^2(u,v) + I^2(u,v)$$

Properties of 2D Discrete Fourier Transform

The Periodicity property: F (u, v) in 2D DFT has a period of N in horizontal and M in vertical directions.

The Symmetry property: The magnitude of the transform is centered on the origin



N/2 − 1
N − 1
0
v
M/2 − 1
M − 1
u
Four back-to-back periods meet here.
N/2
M/2
= Periods of the 2-D DFT.
= M × N data array resulting from the computation of F(u, v).

Fig 3.1 M x N Fourier Spectrum and Spectrum obtained by multiplying f(x,y) by $(-1)^{x+y}$

Origin = 0,0

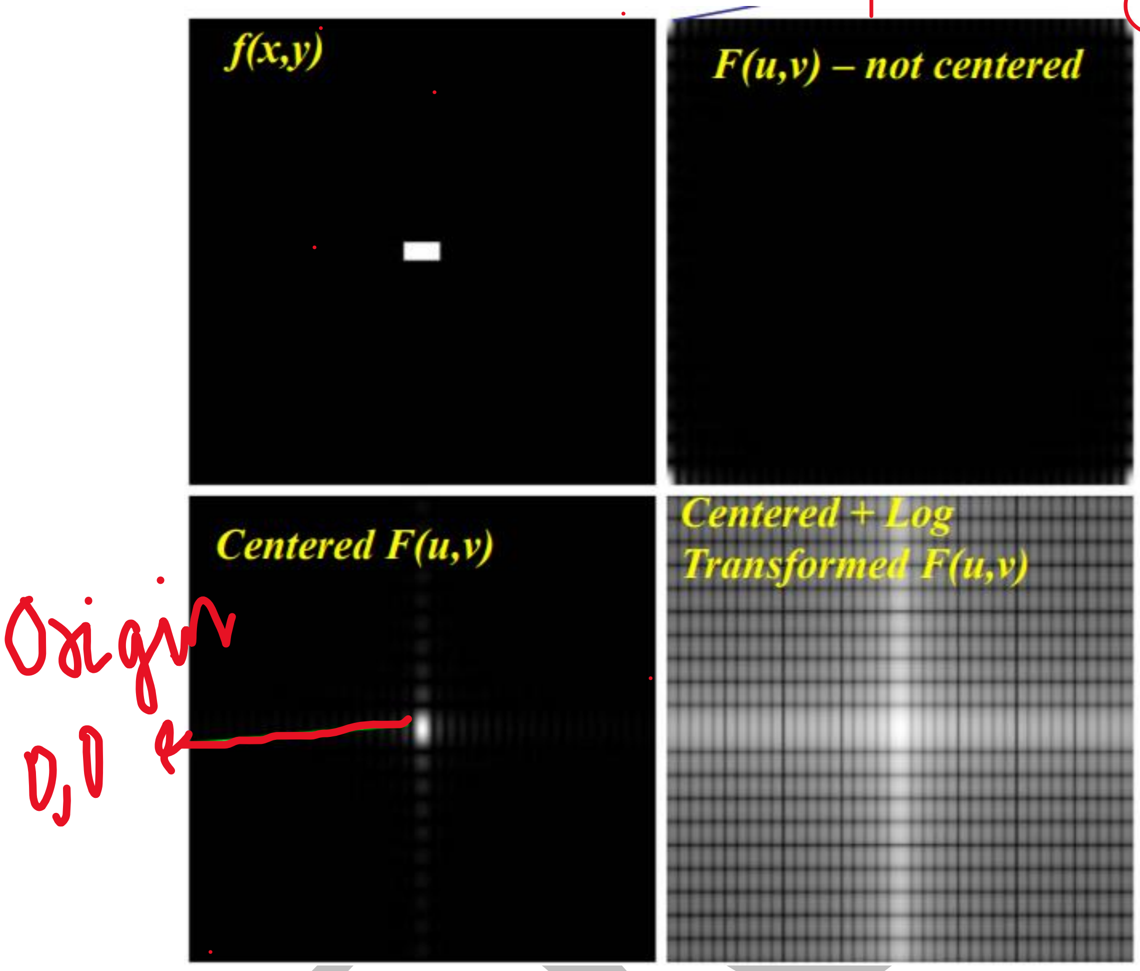


Fig 3.2 a) Image b) Spectrum across four corners c) Centred Spectrum d) Log Transformed image

The center value (at the origin) of the Frequency Spectrum corresponds to the ZERO frequency component which also referred to as the DC component in an image: Substituting 0,0 to the origin, the Fourier transform function yields to the average/DC component value as follows:

$$F(0,0) = \frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)$$

F (0,0) is called "dc" component of the spectrum (current of zero frequency)

The above equation implies that the value of the Fourier transform at the origin is equal to the average gray level of the image.

3.5 Properties of 2D DFT

1) Fourier transform is conjugate symmetric.

F (u, v) =F*(-u, -v)

so |F (u, v) |=|F (-u, -v) |

which means that the Fourier spectrum is symmetric.

2) The relationship between the samples in the spatial and frequency domain

$\Delta u = \frac{1}{M\Delta x}$ and $\Delta v = \frac{1}{N\Delta y}$

3) Translation and Rotation

Fourier transform pairs are given by

$$f(x,y)e^{j2\pi(u_0x/M + v_0y/N)} \Leftrightarrow F(u-u_0, v-v_0)$$

$$f(x-x_0, y-y_0) \Leftrightarrow F(u,v)e^{-j2\pi(x_0u/M + y_0v/N)}$$

Using the polar coordinates

x= r cosθ , y = r sinθ u = ω cosφ v = ω sinφ

It results in the following transform pair

$$f(r, \theta+\theta_0) \Leftrightarrow F(\omega, \varphi+\theta_0)$$

which indicates that rotating f(x, y) by an angle θ_0 rotates F(u, v) by the same angle.

4) Periodicity

The 2D Fourier transform and its inverse are infinitely periodic in the u and v directions

$$F(u,v) = F(u+k_1M, v) = F(u, v+k_2N) = F(u+k_1M, v+k_2N)$$

and

$$f(x,y) = f(x+k_1M, y) = f(x, y+k_2N) = f(x+k_1M, y+k_2N)$$

*Summary of the DFT pairs

Name	DFT pairs
Linearity	$af_1(x,y)+bf_2(x,y) \Leftrightarrow aF_1(u,v)+bF_2(u,v)$
Translation (general)	$f(x,y)e^{j2\pi(u_0x/M+v_0y/N)} \Leftrightarrow F(u-u_0,v-v_0)$ $f(x-x_0,y-y_0) \Leftrightarrow F(u,v)e^{-j2\pi(ux_0/M+vy_0/N)}$
Translation to center of the frequency rectangle, $(M/2,N/2)$	$f(x,y)(-1)^{x+y} \Leftrightarrow F(u-M/2,v-N/2)$ $f(x-M/2,y-N/2) \Leftrightarrow F(u,v)(-1)^{u+v}$
Rotation	$f(r,\theta+\theta_0) \Leftrightarrow F(\omega,\varphi+\theta_0)$ $r=\sqrt{x^2+y^2}\quad \theta=\tan^{-1}(y/x)\quad \omega=\sqrt{u^2+v^2}\quad \varphi=\tan^{-1}(v/u)$
Convolution theorem[†]	$f\star h)(x,y) \Leftrightarrow (F\bullet H)(u,v)$ $(f\bullet h)(x,y) \Leftrightarrow (1/MN)[(F\star H)(u,v)]$
Correlation theorem[†]	$(f\,☆\,h)(x,y) \Leftrightarrow (F^*\bullet H)(u,v)$ $(f^*\bullet h)(x,y) \Leftrightarrow (1/MN)[(F\,☆\,H)(u,v)]$
Discrete unit impulse	$\delta(x,y) \Leftrightarrow 1$ $1 \Leftrightarrow MN\delta(u,v)$
Rectangle	$\mathrm{rec}[a,b] \Leftrightarrow ab\dfrac{\sin(\pi ua)}{(\pi ua)}\dfrac{\sin(\pi vb)}{(\pi vb)}e^{-j\pi(ua+vb)}$
Sine	$\sin(2\pi u_0x/M+2\pi v_0y/N) \Leftrightarrow \dfrac{jMN}{2}[\delta(u+u_0,v+v_0)-\delta(u-u_0,v-v_0)]$
Cosine	$\cos(2\pi u_0x/M+2\pi v_0y/N) \Leftrightarrow \dfrac{1}{2}[\delta(u+u_0,v+v_0)+\delta(u-u_0,v-v_0)]$

3.6 Basics of filtering in the frequency domain

Filtering techniques in the frequency domain are based on modifying the Fourier transform to achieve a specific objective, and then computing the inverse DFT to get us back to the spatial domain.

Basic Steps for Filtering in the Frequency Domain:

1. Multiply the input image by (-1) $^{x+y}$ to center the transform.
2. Compute F (u, v), the DFT of the image from (1).
3. Multiply F (u, v) by a filter function H (u, v).
4. Compute the inverse DFT of the result in (3).

5. Obtain the real part of the result in (4).
6. Multiply the result in (5) by $(-1)^{x+y}$
.
Given the filter H (u, v) (filter transfer function) in the frequency domain, the Fourier transform of the output image (filtered image) is given by:

G (u, v) = H (u, v) F (u, v)
The filtered image g (x, y) is simply the inverse Fourier transform of G (u, v).

$$g(x, y) = \Im^{-1}[G(u,v)]$$

3.7 Smoothing in the Frequency Domain

Smoothing is a low pass operation in the frequency domain. Smoothing is achieved in the frequency domain by high frequency attenuation of a specified range of high frequency components in the transform of a given image.

There are three types of lowpass filters: Ideal, Butterworth, and Gaussian Low pass filters

1. Ideal Low Pass filter

A lowpass filter is a filter that "cuts off" all high frequency components of the Fourier transform that are at a distance greater than a specified distance D_0 from the origin of the transform. However, ILPF is not used practically.

The transfer function of an Ideal Low pass filter is given by

$$H(u,v) = \begin{cases} 1 & \text{if } D(u,v) \le D_0 \\ 0 & \text{if } D(u,v) > D_0 \end{cases}$$

Where D (u, v) is the distance from point (u, v) to the center of frequency rectangle and D_0 is the nonnegative constant.

$$D(u,v) = \left[(u - P/2)^2 + (v - Q/2)^2\right]^{1/2}$$

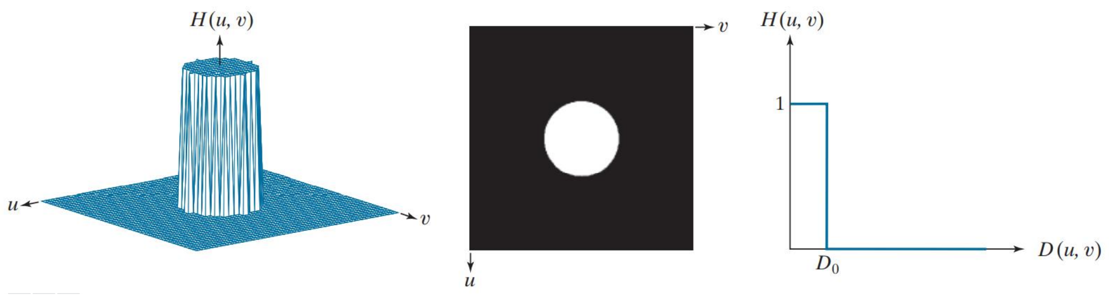


Fig 3.3 a) Perspective plot of ILPF b) Function as an image c) Radial cross section

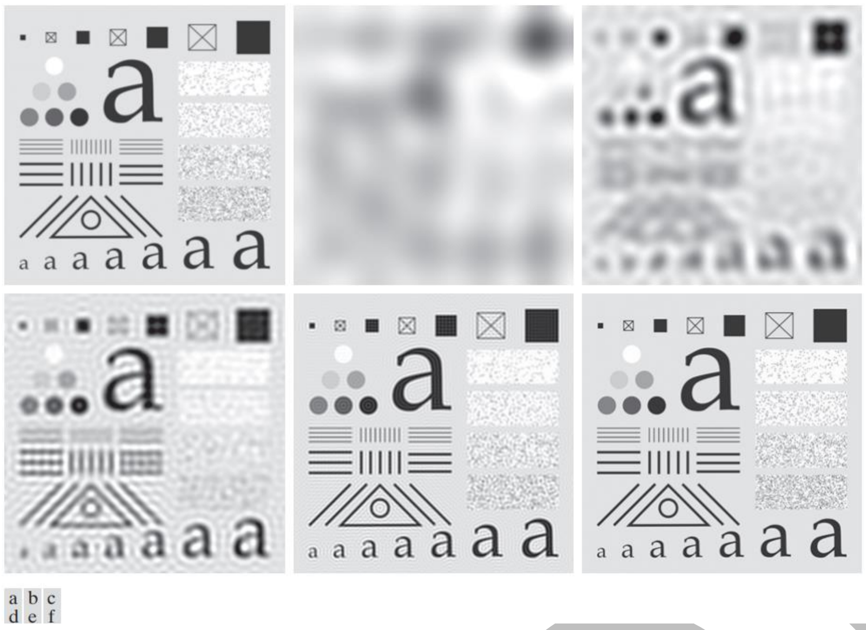


Fig 3.4 a) Original image b- f) Results of filtering using ILPFs with cut-off frequencies set at radii values 10, 30, 60, 160, and 460

2) Butterworth Low pass filter

It has a parameter called the filter order. For high values of filter order, it approaches the form of the ideal filter whereas for low filter order values it acts as a Gaussian filter. It may be viewed as a transition between two extremes. The transfer function of a Butterworth low pass filter (BLPF) of order n with cut off frequency at distance Do from the origin is defined as

$$H(u,v) = \frac{1}{1+\left[D(u,v)/D_0\right]^{2n}}$$

A BLPF approaches the sharpness of an ILPF function with considerably less ringing.

The kernel corresponding to the BLPF of order 1 has neither ringing nor negative values. The kernel corresponding to a BLPF of order 2 shows mild ringing and small negative values. As the remaining images show, ringing becomes significant for higher order filters. A BLPF of order 20 has a spatial kernel that exhibits ringing characteristics similar to those of the ILPF.

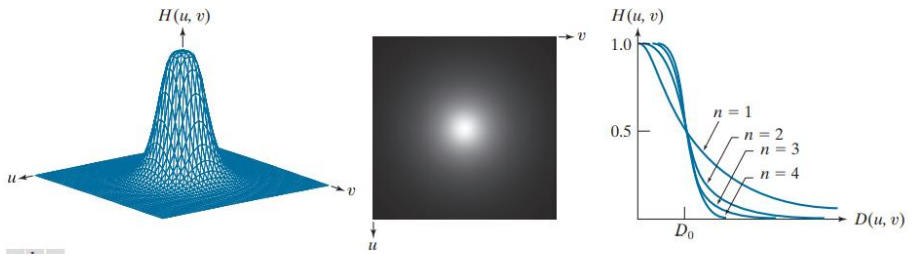


Fig 3.5 a) Perspective plot of BLPF b) Function as an image c) Radial cross section of order n=1 through 4

3) Gaussian Low pass filter

The transfer function of GLPF is of the form

$$H(u,v) = e^{-D^2(u,v)/2\sigma^2}$$

Where D (u, v)- the distance of point (u, v) from the center of the transform
$\sigma = D_0$- specified cut off frequency

The inverse Fourier transform of a frequency domain Gaussian function is also Gaussian. This means that a spatial Gaussian filter kernel, obtained by computing the IDFT, will have no ringing effect.

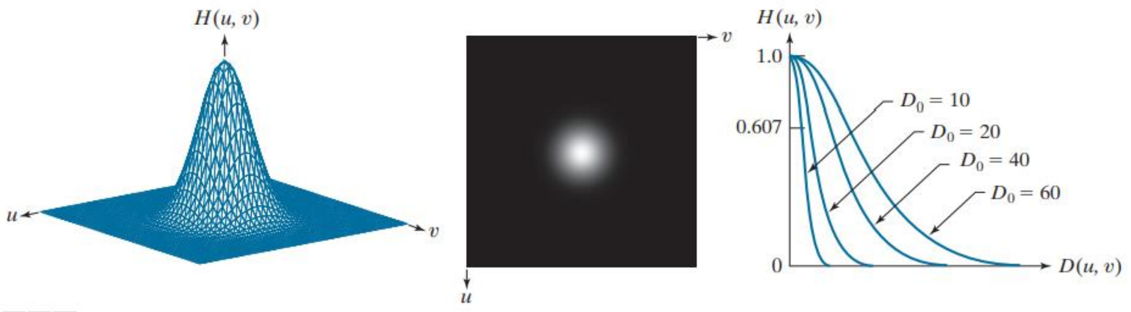


Fig 3.6 a) Perspective plot of GLPF b) Function as an image c) Radial cross section for D_0 values.

3.8 Sharpening in the Frequency Domain

Image sharpening can be achieved by a high pass filtering process, which attenuates the low-frequency components without disturbing high-frequency information. These are radially symmetric and completely specified by a cross section.

If we have the transfer function of a low pass filter the corresponding high pass filter can be obtained using the equation

$$H_{HP}(u, v) = 1 - H_{LP}(u, v)$$

Where $H_{LP}(u, v)$ is the transfer function of the low pass filter.

1) Ideal High Pass filter

An ideal highpass filter (IHPF) transfer function is given by

$$H(u,v) = \begin{cases} 0 & \text{if } D(u,v) \le D_0 \\ 1 & \text{if } D(u,v) > D_0 \end{cases}$$

where, D(u, v) is the distance from the center of the frequency rectangle

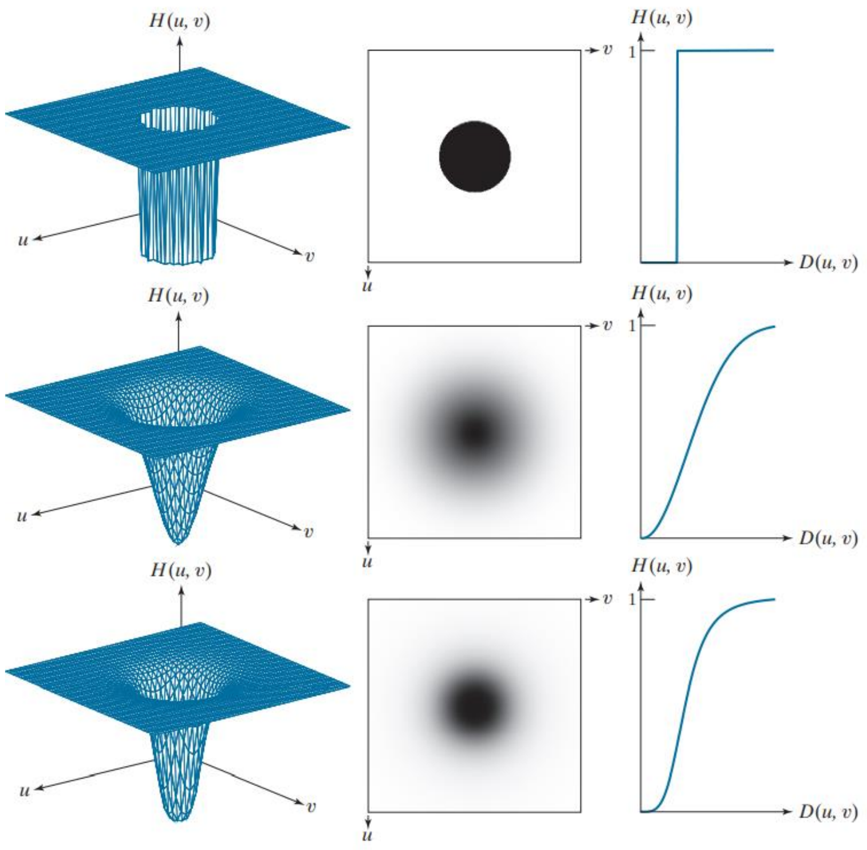


Fig 3.7 Perspective plot , image and radial cross section of ILPF, GLPF and BLPF

2) Butterworth High pass filter

The transfer function of Butterworth High Pass filter of order n is given by the equation

$$H(u,v)=\frac{1}{1+[D_0/D(u,v)]^{2n}}$$

3) Gaussian High Pass filter

The transfer function of a Gaussian High Pass Filter is given by the equation

$$H(u,v)=1-e^{-D^2(u,v)/2D_0^2}$$

Homomorphic Filtering

Homomorphic filters are widely used in image processing for compensating the effect of no uniform illumination in an image. Pixel intensities in an image represent the light reflected from the corresponding points in the objects. As per as image model, image f (x, y) may be characterized by two components: (1) the amount of source light incident on the scene being viewed, and (2) the amount of light reflected by the objects in the scene. These portions of light are called the illumination and reflectance components, and are denoted i (x, y) and r (x , y) respectively. The functions i (x , y) and r (x , y) combine multiplicatively to give the image function f (x , y):

$$f(x, y) = i(x, y) \cdot r(x, y) \qquad (1)$$

where $0 < i(x, y) < a$ and $0 < r(x, y) < 1$. Homomorphic filters are used in such situations where the image is subjected to the multiplicative interference or noise as depicted in Eq. 1. We cannot easily use the above product to operate separately on the frequency components of illumination and reflection because the Fourier transform of f (x , y) is not separable; that is

F [f (x, y)) not equal to F [i (x, y)]. F [r (x, y)].

We can separate the two components by taking the logarithm of

the two sides $\ln f(x, y) = \ln i(x, y) + \ln r$

(x, y).

Taking Fourier transforms on both sides we get,

$$F[\ln f(x, y)\} = F[\ln i(x, y)\} + F[\ln r(x, y)].$$

that is, F (x, y) = I (x, y) + R (x, y), where F, I and R are the Fourier transforms ln f (x, y), ln i (x, y), and ln r (x, y). respectively. The function F represents the Fourier transform of the sum of two images: a low-frequency illumination image and a high-frequency reflectance image. If we now apply a filter with a transfer function that suppresses low- frequency components and enhances high-frequency components, then we can suppress the illumination component and enhance the reflectance component. Taking the inverse transform of F (x, y) and then anti-logarithm, we get

f' (x, y) = i'

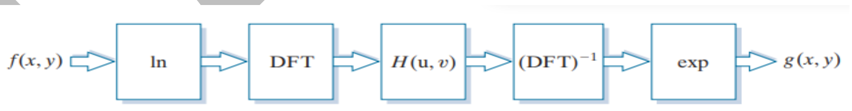


Fig 3.8 Steps in Homomorphic filtering

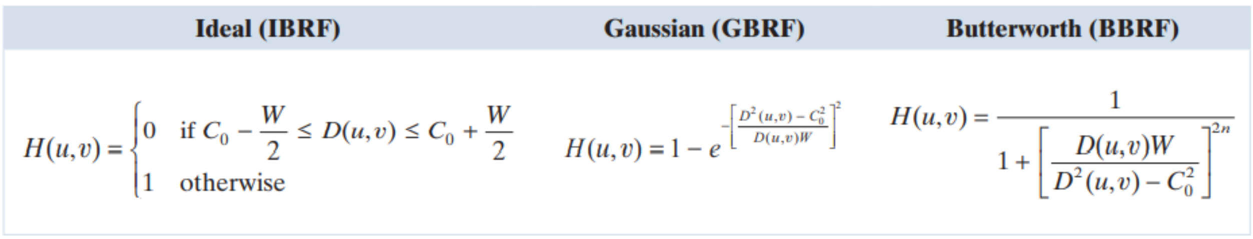
3.9 Selective Filtering

Selective filtering is to process specific bands of frequencies or small regions of the frequency rectangle. Filters in the first category are called band filters. If frequencies in the band are filtered out, the band filter is called a bandreject filter; similarly, if the frequencies are passed, the filter is called a bandpass filter. Filters in the second category are called notch filters. They are further classified as whether the frequencies in the notch areas are rejected or passed.

BANDREJECT AND BANDPASS FILTERS

Bandpass and Bandreject filter transfer functions in the frequency domain can be constructed by combining lowpass and highpass filter transfer functions. Lowpass filter transfer functions are the basis for forming highpass, bandreject, and bandpass filter functions. A bandpass filter transfer function is obtained from a bandreject function.

$H_{BP}(u, v) = 1 – H_{BR}(u, v)$



Ideal (IBRF)	Gaussian (GBRF)	Butterworth (BBRF)
$H(u,v) = \begin{cases} 0 & \text{if } C_0 - \frac{W}{2} \le D(u,v) \le C_0 + \frac{W}{2} \\ 1 & \text{otherwise} \end{cases}$	$H(u,v) = 1 - e^{-\left[\frac{D^2(u,v) - C_0^2}{D(u,v)W}\right]^2}$	$H(u,v) = \frac{1}{1 + \left[\frac{D(u,v)W}{D^2(u,v) - C_0^2}\right]^{2n}}$

Fig 3.9 Transfer functions of Bandreject filters

NOTCH FILTERS

Notch filters are the most useful of the selective filters. A notch filter rejects (or passes) frequencies in a predefined neighbourhood of the frequency rectangle. Zerophase-shift filters must be symmetric about the origin (center of the frequency rectangle), so a notch filter transfer function with center at (u_0, v_0) must have a corresponding notch at location $(-u_0, -v_0)$.Notch reject filter transfer functions are constructed as products of highpass filter transfer functions whose centres have been translated to the centres of the notches.

3.10 Fast Fourier Transform Algorithm

Discrete Fourier transform and Inverse Discrete Fourier transform takes a lot of computations in real time as learnt theoretically. To implement the transform practically, Fast Fourier Transform algorithm is used. Fast Fourier transform (FFT), reduces computations to the order of MN log2 MN multiplications and additions.

Review questions

1. Compare and contrast spatial domain and frequency domain
2. How filtering in frequency domain works?
3. What are the basic steps of filtering in frequency domain?
4. What are smoothing filters? Explain the various types of low pass filters.
5. Explain sharpening in frequency domain.
6. Write a short note on selective filtering.
7. Explain homomorphic filtering briefly.
8. What is FFT? Why FFT is used practically and not DFT?
9. Mention the properties of DFT.

References

1. Digital Image Processing, fourth edition by Gonzalez and Woods.
2. www.Imageprocessingplace.com
3. Refer nptel courses on digital image processing for further detail study on FFT and other related concepts.

UNIT II

Chapter 4 Image Restoration and Reconstruction

4.1 Introduction

Restoration attempts to recover an image that has been degraded by using a priori knowledge of the degradation phenomenon. Thus, restoration techniques are oriented toward modelling the degradation and applying the inverse process in order to recover the original image.

It deals with the characteristics of various noise models used in image processing, and how to estimate from image data the parameters that define those models. The chapter gives overview of minimum mean-square-error (Wiener) filtering and its advantages over direct inverse filtering.

Image Restoration refers to a class of methods that aim to remove or reduce the degradations that have occurred while the digital image was being obtained. All-natural images when displayed have gone through some sort of degradation:

a) During display mode
b) Acquisition mode.
c) Processing mode

The degradations may be due to

a) Sensor noise
b) Blur due to camera mis focus
c) Relative object-camera motion
d) Random atmospheric turbulence

4.2 A model of the Image Degradation/Restoration process

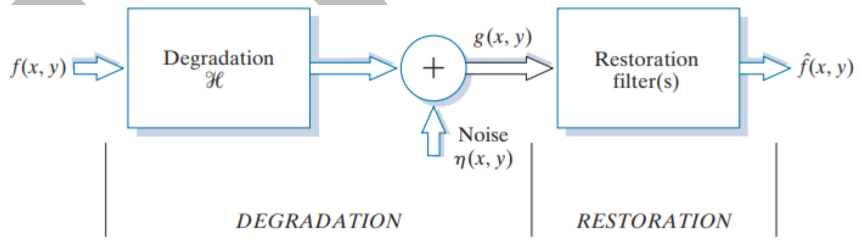


Fig 4.1 A model of the Image Degradation/Restoration process

Degradation process operates on a degradation function that operates on an input image with an additive noise term. Input image is represented by using the notation f(x,y), noise term can be represented as

η(x,y).These two terms when combined gives the result as g(x,y). If we are given g(x,y), some knowledge about the degradation function H or J and some knowledge about the additive noise teem η(x,y), the objective of restoration is to obtain an estimate *f'(x,y)* of the original image. We want the estimate to be as close as possible to the original image. The more we know about h and η, the closer f(x,y) will be to f'(x,y).
If it is a linear position invariant process, then degraded image is given in the spatial domain by

$$g(x,y)=f(x,y)*h(x,y)+\eta(x,y)$$

h(x,y) is spatial representation of degradation function and symbol * represents convolution. In frequency domain we may write this equation as

$$G(u, v) = F(u, v)\, H(u, v) + N(u, v)$$

The terms in the capital letters are the Fourier Transform of the corresponding terms in the spatial domain.

4.3 Noise models

The principal sources of noise in digital images arise during image acquisition and/or transmission. For example, in acquiring images with a CCD camera, light levels and sensor temperature are major factors affecting the amount of noise in the resulting image. Images are corrupted during transmission principally by interference in the transmission channel. For example, an image transmitted using a wireless network might be corrupted by lightning or other atmospheric disturbance.

Spatial and Frequency properties of Noise

1. When the Fourier spectrum of noise is constant, the noise is called white noise.
2. Noise is independent of spatial coordinates. Noise is uncorrelated with respect to the image itself

Common Noise found in Image Processing applications

1. Gaussian Noise
   These noise models are used frequently in practices because of its tractability in both spatial and frequency domain.
   The PDF of Gaussian random variable, z, is given by

$$p(z) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{(z-\bar{z})^2}{2\sigma^2}} \qquad -\infty < z < \infty$$

   where, z represents intensity, z(bar) is the mean (average) value of z and σ is the standard deviation.

2. Rayleigh Noise

The PDF of Rayleigh noise is given by

$$p(z) = \begin{cases} \frac{2}{b}(z - a)e^{-(z-a)^2/b} & z \geq a \\ 0 & z < a \end{cases}$$

The mean and variance of z when this random variable is characterized by a Rayleigh PDF

$$\bar{z} = a + \sqrt{\pi b/4}$$

$$\sigma^2 = \frac{b(4-\pi)}{4}$$

3. Erlang (gamma) noise
   The PDF of Erlang noise is given by

$$p(z) = \begin{cases} \frac{a^b z^{b\ \ 1}}{(b-1)!}e^{-az} & z \geq 0 \\ 0 & z < 0 \end{cases}$$

where the parameters are such that a > b, b is a positive integer, and "!" indicates factorial. The mean and variance of z are

$$\bar{z} = \frac{b}{a}$$

$$\sigma^2 = \frac{b}{a^2}$$

4. Exponential Noise

The PDF of exponential noise is given by

$$p(z) = \begin{cases} ae^{-az} & z \geq 0 \\ 0 & z < 0 \end{cases}$$

where a > 0. The mean and variance of z are

$$\bar{z} = \frac{1}{a}$$

$$\sigma^2 = \frac{1}{a^2}$$

5. Uniform Noise

The PDF of uniform noise is

$$p(z) = \begin{cases} \dfrac{1}{b-a} & a \leq z \leq b \\ 0 & \text{otherwise} \end{cases}$$

The mean and variance of z are

$$\bar{z} = \frac{a+b}{2}$$

$$\sigma^2 = \frac{(b-a)^2}{12}$$

6. Salt-and-Pepper Noise

The PDF of salt-and-pepper noise is given by

$$p(z) = \begin{cases} P_s & \text{for } z = 2^k - 1 \\ P_p & \text{for } z = 0 \\ 1 - (P_s + P_p) & \text{for } z = V \end{cases}$$

where V is any integer value in the range $0 < V < 2^k - 1$

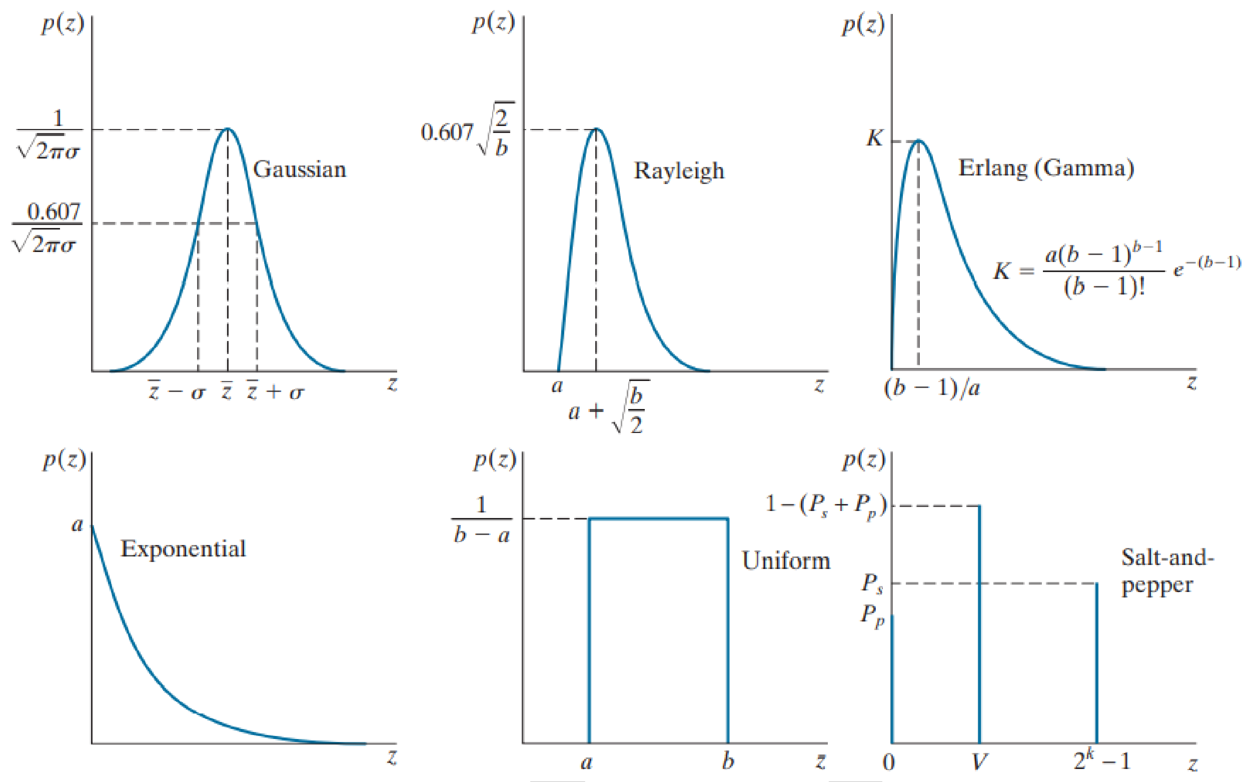


Fig 4.2 Noise Models with their probability density functions

4.4 Restoration in Spatial and Noise Reduction in Frequency Domain

When an image is degraded only by additive noise, the equation becomes

$$g(x, y) = f(x, y) + \eta(x, y)$$

or

$$G(u, v) = F(u, v) + N(u, v)$$

The noise terms are unknown so subtracting them from g (x, y) or G (u, v) is not a realistic approach. In the case of periodic noise it is possible to estimate N(u,v) from the spectrum G(u, v). So, N (u, v) can be subtracted from G (u, v) to obtain an estimate of original image. Spatial filtering can be done when only additive noise is present.

The following techniques can be used to reduce the noise effect

Mean Filters

1)Arithmetic Mean Filters

It is the simplest mean filter. Let S_{xy} represents the set of coordinates in the sub image of size m*n centered at point (x,y). The arithmetic mean filter computes the average value of the corrupted image g(x,y) in the area defined by S_{xy}. The value of the restored image f at any point (x,y) is the arithmetic mean computed using the pixels in the region defined by S_{xy}

$$\hat{f}(x,y) = \frac{1}{mn} \sum_{(r,c)\in S_{xy}} g(r,c)$$

This operation can be using a convolution mask in which all coefficients have value 1/mn
A mean filter smoothes local variations in image Noise is reduced as a result of blurring.

2) Geometric mean filters

An image restored using a geometric mean filter is given by the expression

$$\hat{f}(x,y) = \left[\prod_{(r,c)\in S_{xy}} g(r,c) \right]^{\frac{1}{mn}}$$

where indicates multiplication. Here, each restored pixel is given by the product of all the pixels in the subimage area, raised to the power 1/ mn
A geometric mean filter achieves smoothing comparable to an arithmetic mean filter, but it tends to lose less image detail in the process.

3)Harmonic mean filter

The harmonic mean filter works well for salt noise, but fails for pepper noise. It does well also with other types of noise like Gaussian noise.

The harmonic mean filtering operation is given by the expression

$$\hat{f}(x,y) = \frac{mn}{\sum_{(r,c)\in S_{xy}} \frac{1}{g(r,c)}}$$

4)Contraharmonic mean filter

The Contraharmonic mean filter yields a restored image based on the expression

$$\hat{f}(x,y) = \frac{\sum_{(r,c)\in S_{xy}} g(r,c)^{Q+1}}{\sum_{(r,c)\in S_{xy}} g(r,c)^{Q}}$$

where Q is called the order of the filter. This filter is well suited for reducing or virtually eliminating the effects of salt-and-pepper noise. For positive values of Q, the filter eliminates pepper noise. For negative values of Q, it eliminates salt noise. It cannot do both simultaneously.
The Contraharmonic filter reduces to the arithmetic mean filter if Q = 0, and to the harmonic mean filter if Q = −1.

ORDER STATISTICS FILTERS

Order statistics filters are spatial filters whose response is based on ordering the pixel contained in the image area encompassed by the filter. The response of the filter at any point is determined by the ranking result.

1) Median Filters

These filters replace the value of a pixel by the median of the intensity levels in a predefined neighbourhood of that pixel

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

where, as before, Sxy is a sub image (neighbourhood) centered on point (x, y). The value of the pixel at (x, y) is included in the computation of the median. Median filters provide excellent noise-reduction capabilities, with considerably less blurring than linear smoothing filters of similar size. Median filters are particularly effective in the presence of both bipolar and unipolar impulse noise.

2) Max and Min filters

Using the l00th percentile of ranked set of numbers is called the max filter and is given by the equation

$$\hat{f}(x,y) = \max_{(r,c)\in S_{xy}}\{g(r,c)\}$$

This filter is useful for finding the brightest points in an image or for eroding dark regions adjacent to bright areas

The 0th percentile filter is min filter

$$\hat{f}(x,y) = \min_{(s,t)\in Sxy}\{g(s,t)\}$$

This filter is useful for flinging the darkest point in image. Also, it reduces salt noise of the min operation.

3. Mid-point filter

The midpoint filter simply computes the midpoint between the maximum and minimum values in the area encompassed by the filter

$$\hat{f}(x,y) = \left(\max_{(s,t)\in Sxy}\{g(s,t)\} + \min_{(s,t)\in Sxy}\{g(s,t)\}\right)\Big/2$$

It combines the order statistics and averaging filter. This filter works best for randomly distributed noise like Gaussian or uniform noise.

4. Alpha-Trimmed mean filter

   A filter formed by averaging these remaining pixels is called an alpha-trimmed mean filter. The form of this filter is given by

$$\hat{f}(x,y) = \frac{1}{mn - d} \sum_{(r,c)\in S_{xy}} g_R(r,c)$$

where the value of d can range from 0 to mn − 1. When d = 0 the alpha-trimmed filter reduces to the arithmetic mean filter. If we choose d mn = − 1, the filter becomes a median filter.

Periodic Noise Reduction using frequency domain filtering

The basic idea is that periodic noise appears as concentrated bursts of energy in the Fourier transform, at locations corresponding to the frequencies of the periodic interference. The approach is to use a selective filter to isolate the noise. In restoration of images corrupted by periodic interference, the tool of choice is a notch filter.
The general form of a notch filter transfer function is

$$H_{\text{NR}}(u,v) = \prod_{k=1}^{Q} H_k(u,v) H_{-k}(u,v)$$

A notch pass filter transfer function is obtained from a notch reject function using the expression

$$H_{\text{NP}}(u,v) = 1 - H_{\text{NR}}(u,v)$$

4.5 Inverse Filtering

It is a process of restoring an image degraded by a degradation function H. This function can be obtained by any method. The simplest approach to restoration is direct, inverse filtering.
Inverse filtering provides an estimate F(u,v) of the transform of the original image simply by during the transform of the degraded image G(u,v) by the degradation function.

$$\hat{F}(u,v) = \frac{G(u,v)}{H(u,v)}$$

It shows an interesting result that even if we know the depredation function we cannot recover the underrated image exactly because N(u,v) is not known . If the degradation value has zero or very small values then the ratio N(u,v)/H(u,v) could easily dominate the estimate F(u,v).

4.6 Minimum Mean Square Error (Wiener) Filtering

Wiener Filtering is an approach that incorporates both the degradation function and statistical characteristics of noise into the restoration process. The method is founded on considering images and noise as random variables, and the objective is to find an estimate ˆ f of the uncorrupted image f such that the mean square error between them is minimized. This error measure is defined as

$$e^2 = E\left\{(f - \hat{f})^2\right\}$$

The minimum error function of the above expression is given by the expression

$$\begin{aligned}\hat{F}(u,v) &= \left[\frac{H^*(u,v)S_f(u,v)}{S_f(u,v)|H(u,v)|^2 + S_\eta(u,v)}\right]G(u,v)\\ &= \left[\frac{H^*(u,v)}{|H(u,v)|^2 + S_\eta(u,v)/S_f(u,v)}\right]G(u,v)\\ &= \left[\frac{1}{H(u,v)}\frac{|H(u,v)|^2}{|H(u,v)|^2 + S_\eta(u,v)/S_f(u,v)}\right]G(u,v)\end{aligned}$$

The product of a complex quantity with its conjugate is equal to the magnitude of the complex quantity squared. This result is known as the Wiener filter

The terms used are as follows

1. $\hat{F}(u,v)$ = Fourier transform of the estimate of the undegraded image.
2. $G(u,v)$ = Fourier transform of the degraded image.
3. $H(u,v)$ = degradation transfer function (Fourier transform of the spatial degradation).
4. $H^*(u,v)$ = complex conjugate of $H(u,v)$.
5. $|H(u,v)|^2 = H^*(u,v)H(u,v)$.
6. $S_\eta(u,v) = |N(u,v)|^2$ = power spectrum of the noise [see Eq. (4-89)][†]
7. $S_f(u,v) = |F(u,v)|^2$ = power spectrum of the undegraded image.

4.7 Geometric Mean filter

The generalized form of Wiener filter is the geometric mean filter.

$$\hat{F}(u,v) = \left[\frac{H^*(u,v)}{|H(u,v)|^2}\right]^{\alpha} \left[\frac{H^*(u,v)}{|H(u,v)|^2 + \beta\left[\frac{S_\eta(u,v)}{S_f(u,v)}\right]}\right]^{1-\alpha} G(u,v)$$

where α and β are nonnegative, real constants. The geometric mean filter transfer function consists of the two expressions in brackets raised to the powers α and 1 − α, respectively.

When α = 1 the geometric mean filter reduces to the inverse filter. With a = 0 the filter becomes the so-called parametric Wiener filter, which reduces to the “standard” Wiener filter when β = 1

4.8 Image Reconstruction from projections

The basic principles of projections apply to other CT imaging modalities, such as SPECT (single photon emission tomography), PET (positron emission tomography), MRI (magnetic resonance imaging), and some modalities of ultrasound imaging. One can use projections with Radon transform or fan beam projections or slice theorem for understanding image reconstruction from projections.

A filter function can be used for image reconstruction. This approach is appropriately called filtered back projection. In practice, the data are discrete, so all frequency domain computations are carried out using a 1-D FFT algorithm,

Review questions

1. Explain the model of image degradation and reconstruction with a diagram?
2. What are the different sources of noise?
3. Explain the various noise models.
4. Describe the various spatial filters/frequency-based filters used to reconstruct images.
5. Write a short note on Wiener Filtering.

References

1. Digital Image Processing, fourth edition by Gonzalez and Woods.
2. www.Imageprocessingplace.com
3. Refer nptel courses on digital image processing for further detail study on FFT and other related concepts

Unit 5

Image Segmentation II

10.1 Introduction

Image segmentation is a process of segregating foreground and background of images. Basically, canny edge detection algorithms are used in most applications to detect edges. However, in many applications like medical image processing and imagery applications, instead of basic algorithms active contours known as snakes are used. This chapter focuses on Image segmentation using snakes and level sets.

10.2 Image Segmentation using Snakes

In image segmentation methods, connectivity and homogeneity are based on image data. In medical image segmentation analysis, segmentation depends on anatomy. It is usually difficult to get good segmentation. Active contours are used for image segmentation. Prior knowledge is essential to get to know the real problem.

In an active contour framework, object segmentation is achieved by using a closed contour to the objects' boundary. The snake is active as it continuously evolving such that the energy is minimized.

The energy function for a snake works in two parts, internal and external energies

$$E_{snake} = E_{internal} + E_{external}$$

The internal energy depends on its intrinsic properties such as length and curvature. The external energy depends on factors such as image structure and constraints the user would have imposed. Snakes start with closed curve and minimizes the total energy function to deform until they reach an optimal state. The main advantage of active contour is that it results in closed coherent areas with smooth boundaries.

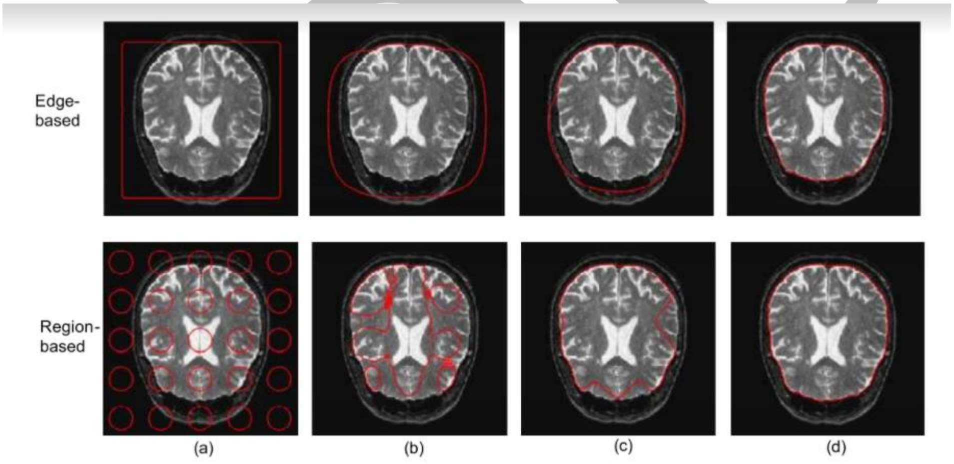


Fig 10.1 Active contour is semi-automatic as it requires the user to mark the initial contour

Edge based and region based active contour a) Initial contour b) and c) Intermediate contour d) Final contour

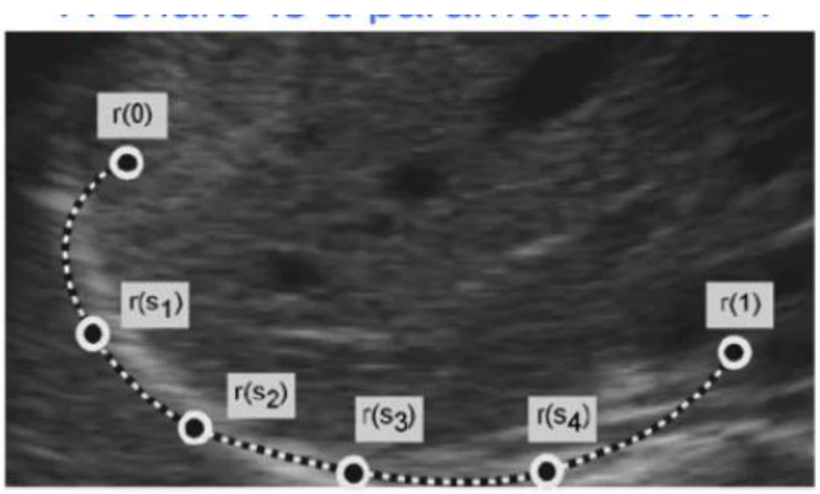


Fig 10.2 A snake is a parametric curve

A higher-level process or the user initializes any curve close to the object boundary. The snake then deforms and moves towards the desired object boundary. In the end, it completely shrink-wraps around the object. Deformable models are curves or surfaces defined within an image domain that can move under the influence of internal forces, which are defined within the curve itself and external forces within the image data.

Problems with Image segmentation using snakes

1. Snakes may over smooth the boundary
2. Initialization is crucial
3. Depends on number and spacing of control points
4. It may not follow topological changes of objects.

10.3 Image Segmentation using Level Sets

Level Set is an alternative representation to closed contours. Level sets fit and track objects of interest by modifying the embedding function instead of a curve function.

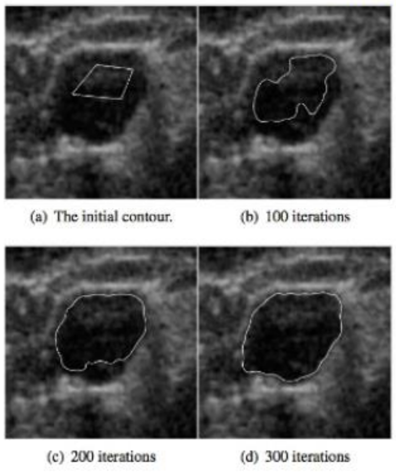


Fig 10.3 Ultrasound image segmentation without reinitialization using Level Set evolution

Review questions

1. What are the disadvantages of image segmentation classical methods?

2. Explain image segmentation using active contours.
3. What are level sets in image segmentation? Explain.

References

1. https://www.cs.cmu.edu/~galeotti/methods_course/Segmentation2-Snakes.pdf
2. Medical Image Processing by G.R. Sinha and Bhagwati Charan Patel, PHI publications
3. http://home.iitj.ac.in/~manpreet.bedi/btp/documents/sar.pdf
4. https://www.slideshare.net/UlaBac/lec11-active-contour-and-level-set-for-medical-image-segmentation (Reference to image Fig:10.3)
5. Credits to Mr. Ulas Bagci, Assist. Prof at UCF, Research center for Computer Vision
6. https://www.cs.cmu.edu/~galeotti/methods_course/Level_Sets_and_Parametric_Transforms.pdf

For detail study, use the mentioned links

Unit 5

Chapter 11 Feature Extraction

11.1 Introduction

Feature extraction almost always follows the output of a segmentation stage, which usually is raw pixel data, constituting either the boundary of a region (i.e., the set of pixels separating one image region from another) or all the points in the region itself. Feature extraction consists of feature detection and feature description. Feature detection refers to finding the features in an image, region, or boundary. Feature description assigns quantitative attributes to the detected features.

11.2 Boundary Preprocessing

11.2.1 Boundary Following (Tracing)

A boundary-following algorithm whose output is an ordered sequence of points. We assume (1) that we are working with binary images in which object and background points are labelled 1 and 0, respectively; and (2) that images are padded with a border of 0's to eliminate the possibility of an object merging with the image border.

The following algorithm traces the boundary of a 1-valued region, R, in a binary image.

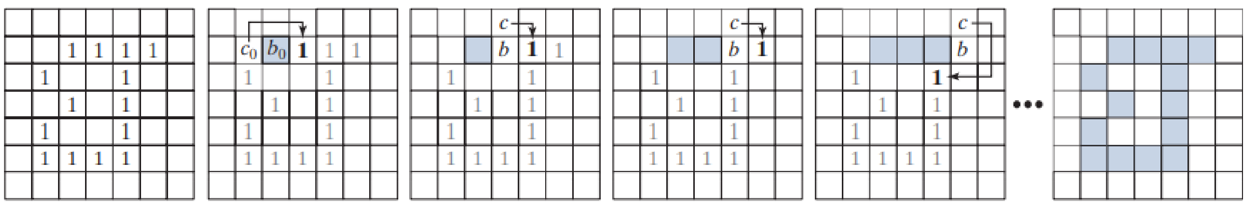
1. Let the starting point, b0, be the uppermost-leftmost point† in the image that is labelled 1. Denote by c0 the west neighbour of b0. Clearly, c0 is always a background point. Examine the 8-neighbors of b0, starting at c0 and proceeding in a clockwise direction. Let b1 denote the first neighbour encountered whose value is 1, and let c1 be the (background) point immediately preceding b1 in the sequence. Store the locations of b0 for use in Step 5.

2. Let b = b0 and c = c0.

3. Let the 8-neighbors of b, starting at c and proceeding in a clockwise direction, be denoted by $n_1, n_2, \ldots\ldots n_8$. Find the first neighbor labelled 1 and denote it by n_k

4. Let b = n_k and c =n_{k-1}.

5. Repeat Steps 3 and 4 until b =b0. The sequence of b points found when the algorithm stops is the set of ordered boundary points



a b c d e f

Fig 11.1 Illustration of the first few steps in the boundary-following algorithm. The point to be processed next is labelled in bold, black; the points yet to be processed are gray; and the points found by the algorithm are shaded. Squares without labels are considered background (0) values.

11.2.2 Chain Codes

Chain codes are used to represent a boundary by a connected sequence of straight-line segments of specified length and direction. A chain code representation is based on 4- or 8-connectivity of the segments. The direction of each segment is coded by using a numbering scheme. A boundary code formed as a sequence of such directional numbers is referred to as a Freeman chain code.

Digital images usually are acquired and processed in a grid format with equal spacing in the x- and y-directions, so a chain code could be generated by following a boundary in, say, a clockwise direction and assigning a direction to the segments connecting every pair of pixels

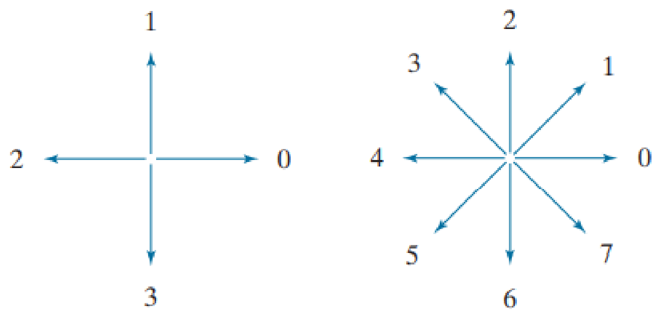


Fig 11.2 Direction numbers for 1) 4-directional 2) 8-directional chain code

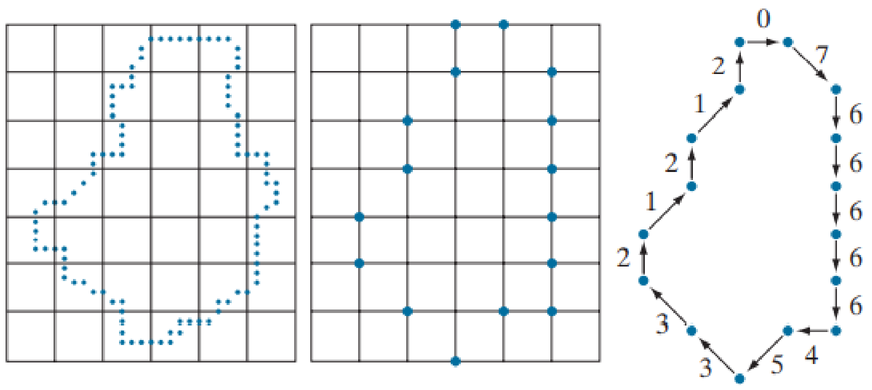
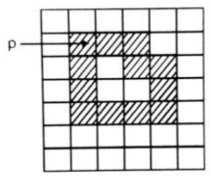


Fig 11.3 a) Digital boundary with resampling grid b) Resampled grid c) 8-directional chain-coded boundary

Example of 4-directional chain code



Consider the given image, the 4-directional chain code is given by

0 0 3 0 3 3 2 2 2 1 1 1

First difference is

0 0 3 0 3 3 2 2 2

0 3 1 3 0 3 0 0 3 0

Circular difference = last -first = 0 – 0 = 0

Therefore, circular chain code is 0 0 0 3 0 3 3 2 2 2 1 1 1

Shape number - Shape no of a boundary obtained from a chain code is defined as the smallest magnitude of the circular first difference.

11.2.3 Slope Chain codes (SCC)

Using Freeman chain codes generally requires resampling a boundary to smooth small variations, a process that implies defining a grid and subsequently assigning all boundary points to their closest neighbours in the grid. The SCC of a 2-D curve is obtained by placing straight-line segments of equal length around the curve, with the end points of the segments touching the curve. Obtaining an SSC requires calculating the slope changes between contiguous line segments, and normalizing the changes to the continuous (open) interval (-1, 1).

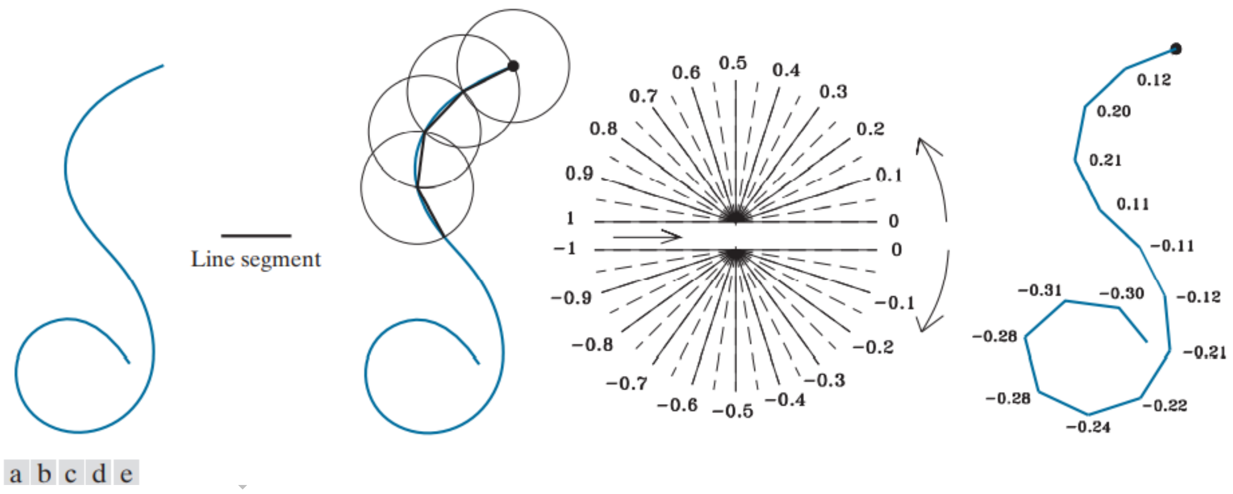


Fig 11.4 (a) An open curve. (b) A straight-line segment. (c) Traversing the curve using circumferences to determine slope changes; the dot is the origin (starting point). (d) Range of slope changes in the open interval (-1 ,1)

11.2.4 Boundary Approximation using Minimum-Perimeter Polygon

A digital boundary can be approximated with arbitrary accuracy by a polygon. For a closed curve, the approximation becomes exact when the number of segments of the polygon is equal to the number of points in the boundary, so each pair of adjacent points defines a segment of the polygon. A boundary can be represented by Minimum-Perimeter Polygon.

MPP Algorithm

1. The MPP bounded by a simply connected cellular complex is not self-intersecting.

2. Every convex vertex of the MPP is a W vertex, but not every W vertex of a boundary is a vertex of the MPP.

3. 3. Every mirrored concave vertex of the MPP is a B vertex, but not every B vertex of a boundary is a vertex of the MPP.

4. All B vertices are on or outside the MPP, and all W vertices are on or inside the MPP.

5. The uppermost-leftmost vertex in a sequence of vertices contained in a cellular complex is always a W vertex of the MPP.

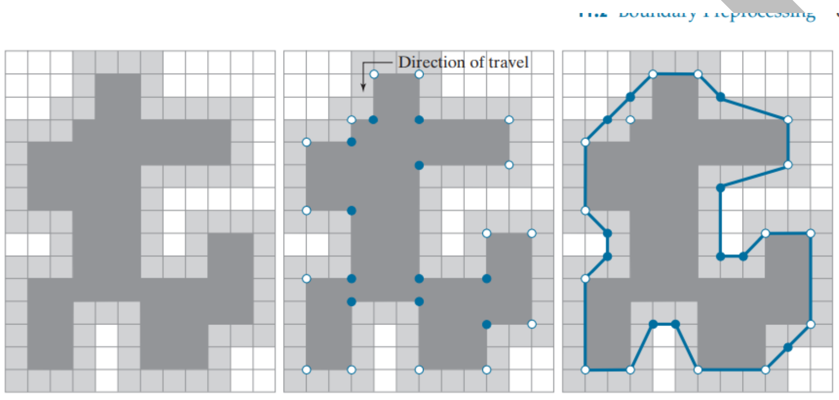


Fig 11.5 a) The Region b) Convex and Concave c) MPP superimposed on concave, convex vertices

11.2.5 Signatures

A signature is a 1-D functional representation of a 2-D boundary. Plot the distance from the centroid to the boundary as a function of angle. The basic idea of using signatures is to reduce the boundary representation to a 1-D function. Though simplicity is the advantage, the disadvantage is scaling of the entire function depends on only two values: the minimum and maximum.

Distance versus angle- Plot the distance from the centroid to the boundary as a function of angles: signature = r(θ), θ = 0 ~ 2π

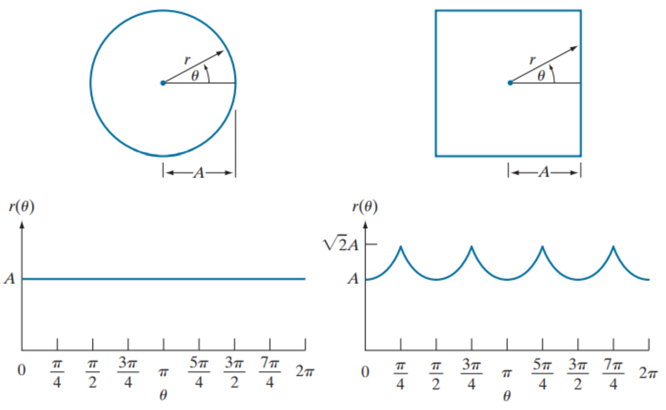


Fig 11.6 Distance versus angle signatures

11.2.6 Skeletons, Medial Axes and Distance transforms

The skeleton of a region is the set of points in the region that are equidistant from the border of the region. The skeleton is obtained using one of two principal approaches: (1) by successively thinning the region (e.g., using morphological erosion) while preserving end points and line connectivity (this is called topology-preserving thinning); or (2) by computing the medial axis of the region via an efficient implementation of the medial axis transform (MAT). The skeleton of a region is defined as its medial axis.

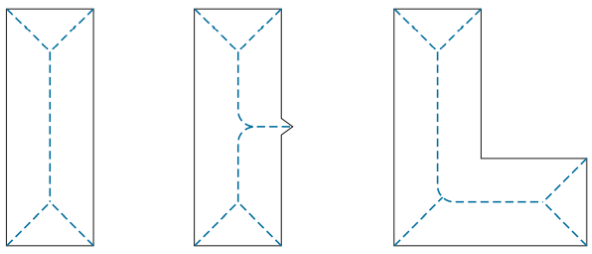


Fig 11.7 Medial axes of three simple regions

The distance transform of a region of foreground pixels in a background of zeros is the distance from every pixel to the nearest nonzero valued pixel.

$$\begin{matrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{matrix} \qquad \begin{matrix} 1.41 & 1 & 1 & 1 & 1.41 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1.41 & 1 & 1 & 1 & 1.41 \end{matrix}$$

Fig 11.8 An image with its distance transform

11.3 Boundary feature descriptors

11.3.1 Basic Boundary descriptors

The length of a boundary is one of its simplest descriptors.

The diameter of a boundary B is defined as

$$diameter(B) = \max_{i,j}\left[D\left(p_i, p_j\right)\right]$$

where D is a distance measure and pi and pj are points on the boundary. The value of the diameter and the orientation of a line segment connecting the two extreme points that comprise the diameter is called the major axis (or longest chord) of the boundary.

The length and orientation of the major axis are given by

$$length_m = \left[(x_2 - x_1)^2 + (y_2 - y_1)^2\right]^{1/2}$$

$$angle_m = \tan^{-1}\left[\frac{y_2 - y_1}{x_2 - x_1}\right]$$

The minor axis (also called the longest perpendicular chord) of a boundary is defined as the line perpendicular to the major axis.

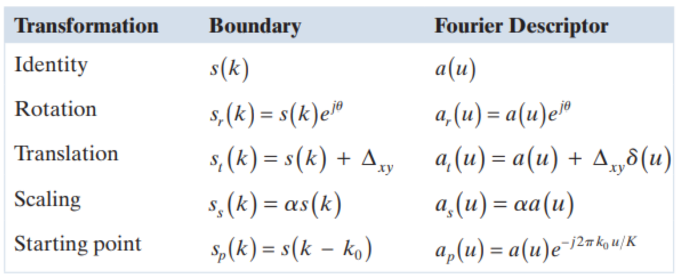
The curvature of a boundary is defined as the rate of change of slope.

11.3.2 Fourier descriptors

It represents the boundary as a sequence of coordinates and treats each coordinate pair as a complex number. The discrete Fourier transform (DFT) of s(k) is

$$a(u) = \sum_{k=0}^{K-1} s(k)e^{-j2\pi uk/K}$$

Where u = 0,1, 2, …., K-1. The complex coefficients a (u) is called the Fourier descriptors of the boundary.



Transformation	Boundary	Fourier Descriptor
Identity	$s(k)$	$a(u)$
Rotation	$s_r(k) = s(k)e^{j\theta}$	$a_r(u) = a(u)e^{j\theta}$
Translation	$s_t(k) = s(k) + \Delta_{xy}$	$a_t(u) = a(u) + \Delta_{xy}\delta(u)$
Scaling	$s_s(k) = \alpha s(k)$	$a_s(u) = \alpha a(u)$
Starting point	$s_p(k) = s(k - k_0)$	$a_p(u) = a(u)e^{-j2\pi k_0 u/K}$

Fig 11.9 Basic properties of Fourier descriptors

11.4 Region Feature Descriptors

11.4.1 Basic Descriptors

The area of a region is defined as the number of pixels in the region.

The perimeter of a region is the length of its boundary.

Compactness of a region, defined as the perimeter squared over the area

$$\text{compactness} = \frac{p^2}{A}$$

Another Dimensionless measure is circularity (also called roundness), defined as

$$\text{circularity} = \frac{4\pi A}{p^2}$$

The value of this descriptor is 1 for a circle (its maximum value) and p 4 for a square.

The eccentricity of a region relative to an ellipse as the eccentricity of an ellipse that has the same second central moments as the region.

11.4.2 Topological Descriptors

Topology is the study of properties of a figure that are unaffected by any deformation, provided that there is no tearing or joining of the figure (sometimes these are called rubber-sheet distortions). Topological property useful for region description is the number of connected components of an image. The number of holes H and connected components C in a figure can be used to define the Euler number, E. The Euler number is also a topological property. E =C - H

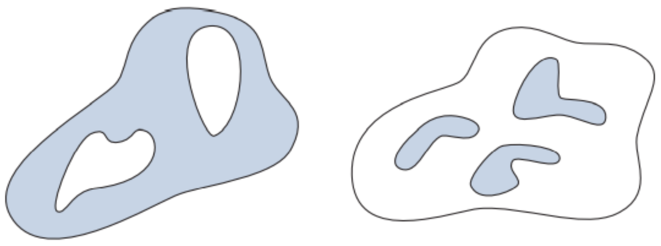


Fig 11.10 (a) A region with two holes. (b) A region with three connected components.

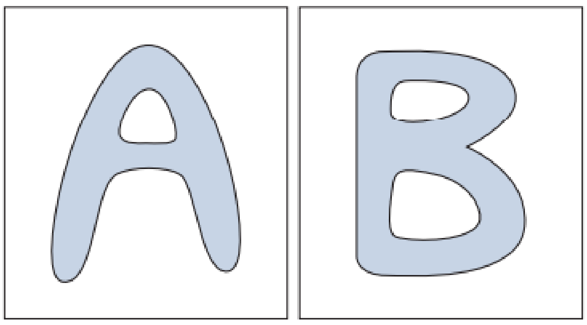


Fig 11.11 Region with Euler number 0 and -1 respectively

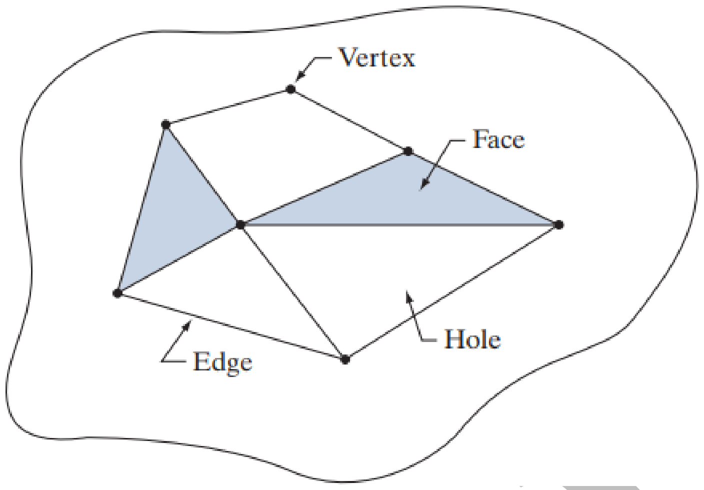


Fig 11.12 A region with polygonal network

11.4.3 Central Moments

The 2-D moment of order (p + q) of an M N× digital image, f (x, y), is defined as

$$m_{pq} = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} x^p y^q f(x,y)$$

where p = 0,1,2... and q = 0,1,2 ... are integers.

The central moment of order (p + q) is defined as

$$\mu_{pq} = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} (x-\bar{x})^p (y-\bar{y})^q f(x,y)$$

where p = 0,1,2... and q = 0,1,2 ... are integers

$$\bar{x} = \frac{m_{10}}{m_{00}} \text{ and } \bar{y} = \frac{m_{01}}{m_{00}}$$

The normalized central moment of order (p + q), is defined as

$$\eta_{pq} = \frac{\mu_{pq}}{\mu_{00}^{\gamma}}$$

Where $\gamma = \frac{p+q}{2} + 1$

A set of seven, 2-D moment invariants can be derived from the second and third normalized central moments

$$\phi_1 = \eta_{20} + \eta_{02}$$
$$\phi_2 = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2$$
$$\phi_3 = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{12} - \eta_{03})^2$$
$$\phi_4 = (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2$$
$$\phi_5 = (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})\left[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2\right] + (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})\left[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2\right]$$
$$\phi_6 = (\eta_{20} - \eta_{02})\left[(\eta_{30} + \eta_{12})^2 - (\eta_{21} - \eta_{03})^2 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03})\right]$$
$$\phi_7 = (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})\left[(\eta_{30} + \eta_{12})^2 - 3(\eta_{30} + \eta_{12})^2\right] + (\eta_{30} - 3\eta_{12})(\eta_{21} + \eta_{03})\left[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2\right]$$

11.5 Whole Image features

The principal feature detection methods are based on detecting corners and the other works with entire regions in an image.

The Harris-Stephens Corner Detector

The basic approach is this: Corners are detected by running a small window over an image. The detector window is designed to compute intensity changes

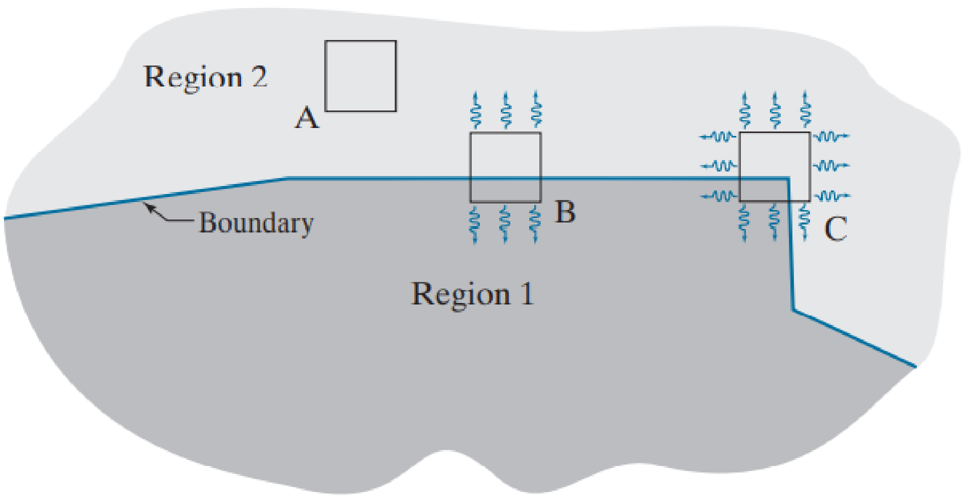


Fig 11.13 Illustration of Harris-Stephens Corner detector

Three scenarios

(1) Areas of zero (or small) intensity changes in all directions, which happens when the window is located in a constant (or nearly constant) region, as in location A.

(2) areas of changes in one direction but no (or small) changes in the orthogonal direction, which this happens when the window spans a boundary between two regions, as in location B
(3) areas of significant changes in all directions, a condition that happens when the window contains a corner (or isolated points), as in location C.

The HS corner detector is a mathematical formulation that attempts to differentiate between these three conditions.

Let f denote an image, and let f (s, t) denote a patch of the image defined by the values of (s, t). A patch of the same size, but shifted by (x, y), is given by f (s + x, t + y).

$$C(x,y) = \sum_s \sum_t w(s,t)\left[x f_x(s,t) + y f_y(s,t)\right]^2$$

The above equation can be written in the form

$$C(x,y) = [x \;\; y]\,\mathbf{M}\begin{bmatrix} x \\ y \end{bmatrix}$$

Where

$$\mathbf{M} = \sum_s \sum_t w(s,t)\,\mathbf{A}$$

$$\mathbf{A} = \begin{bmatrix} f_x^2 & f_x f_y \\ f_x f_y & f_y^2 \end{bmatrix}$$

Matrix M is called as Harris matrix.

11.6 Scale Invariant Feature Transform (SIFT)

SIFT is an algorithm developed by Lowe [2004] for extracting invariant features from an image. It is called a transform because it transforms image data into scale-invariant coordinates relative to local image features. SIFT is by far the most complex feature detection and description approach. In the presence of variables such as scale changes, rotation, changes in illumination, and changes in viewpoint, methods like SIFT is used.

SIFT features (called key points) are invariant to image scale and rotation, and are robust across a range of affine distortions, changes in 3-D viewpoint, noise, and changes of illumination. The input to SIFT is an image. Its output is an n-dimensional feature vector whose elements are the invariant feature descriptors

The first stage of the SIFT algorithm is to find image locations that are invariant to scale change. This is achieved by searching for stable features across all possible scales, using a function of scale known as scale space. The parameter controlling the smoothing is referred to as the scale parameter. In SIFT, Gaussian kernels are used to implement smoothing, so the scale parameter is the standard deviation.

SIFT subdivides scale space into octaves, with each octave corresponding to a doubling of σ. SIFT initially finds the locations of key points using the Gaussian filtered images, then refines the locations and validity of those key points using improving the accuracy and eliminating edge responses, compute key point orientations and descriptors.

Review questions

1. How can one represent and describe features after extraction?
2. What are region descriptors?
3. Explain Fourier descriptors briefly.
4. What are chain codes? Explain with examples.
5. Write a short note on signatures
6. Describe Whole image features.
7. Explain Harris-Stephens Corner detector algorithm.
8. Write a short note on SIFT.

References

1. Digital Image Processing by Gonzalez and Woods, 4th Edition.
2. All Images courtesy from Digital Image Processing by Gonzalez and Woods ,4th Edition
3. https://www.ee.columbia.edu/~xlx/courses/ee4830-sp08/notes/lec10.pdf

Chapter 8 Image Segmentation, Edge Detection, Thresholding, and Region Detection

8.0 Introduction

The chapter's segmentation algorithms are based on one of two fundamental properties of image intensity values: discontinuity or similarity. In the first category, an image is partitioned into regions according to abrupt changes in intensity, such as edges. The second category of approaches is based on segmenting an image into regions that are similar based on a set of predefined criteria. Methods in this category include thresholding, region growth, and region splitting and merging. We demonstrate that by combining methods from disparate categories, such as edge detection and thresholding, we can improve segmentation performance. Additionally, we discuss image segmentation using clustering and superpixels, as well as an introduction to graph cuts, a technique that is well-suited for extracting the image's principal regions. Following that, a discussion of image segmentation based on morphology is presented, an approach that combines several of the characteristics of segmentation based on techniques.

After reading this chapter, readers should be able to:

- Understand the characteristics of different types of edges encountered in exercise.
- Understand how spatial filtering can be used for edge detection.
- Familiarize yourself with additional types of edge detection techniques that extend beyond spatial filtering.
- Utilize several different approaches to gain a better understanding of image thresholding.
- Understand how to optimize segmentation by combining thresholding and spatial filtering.
- Familiarize yourself with region-based segmentation techniques, such as clustering and superpixels.
- Understand the segmentation techniques used with graph cuts and morphological watersheds.

- Understand the fundamental techniques for incorporating motion into image segmentation.

8.1 Fundamentals

Let R denote the the whole spatial region that an image occupies. Segmentation is a process that divides the image R into n subregions, R1, R2,..., Rn, in such a way that

a. $\bigcup_{i-1}^{n} R_i = R$
b. $R_i$is a connected set, for i=0,1,2,3..,n;
c. $R_i \cap R_j = \emptyset \; for \; all \; i \; and \; j, i \neq j$
d. $Q(R_i) = TRUE \; for$ i = 0,1,2,3.., n
e. $Q\left(R_i \cup \; R_j\right) = TRUE$ for any adjacent regions $R_i \; and \; R_j$

P (R_k) signifies a logical predicate described over the points in the set R_i and ∅ whereas is the null set. Condition (a) specifies that the segmentation should be complete; each pixel must be contained within a region. Conditions (b)   require that points within a region be connected in a predefined way. As indicated by condition (c), the regions must be disjoint. Condition (d) specifies the properties that all pixels within a segmented region must satisfy—for example, P (Ri) = TRUE if all pixels within Ri have the same grey level. Finally, condition (e) establishes that regions Ri and Rj are distinct in terms of the predicate P.

As we can see, the fundamental issue in segmentation is partitioning an image into regions that meet the aforementioned criteria.
Monochrome image segmentation algorithms are typically based on one of two important properties of intensity values: discontinuity or similarity. In the first category, we assume that region boundaries are sufficiently distinct from one another and from the background to permit boundary detection via local intensity discontinuities. Edge-based segmentation is the primary technique used in this category. The second category of Region-based segmentation approaches is based on partitioning an image into regions that are similar based on a set of predefined criteria.

The prior concepts are illustrated in Figure 8.1. Figure 8.1 (a) depicts a region of constant intensity overlaid on a darker, also constant-intensity background. The overall image is composed of these two regions. The result of computing the inner region's boundary using intensity discontinuities is shown in Figure 8.1 (b). Points on either side of the boundary are black (zero) due to the absence of intensity discontinuities in those regions. To segment the image, we allocate one level (for example, white) to all pixels on or inside the boundary and another level (for example, black) to all points outside the boundary. The result of such a procedure is depicted in Figure 8.1 (c). As we can see, this result satisfies conditions (a) through (c) stated at the beginning of this section. The predicate of condition (d) is as follows: If a pixel is on or within the boundary, it should be labelled white; otherwise, it should be labelled black. As shown in Fig. 8.1 (c), this predicate is TRUE for the point's labelled black or white. Likewise, both segmented regions (object and background) satisfy condition (e)

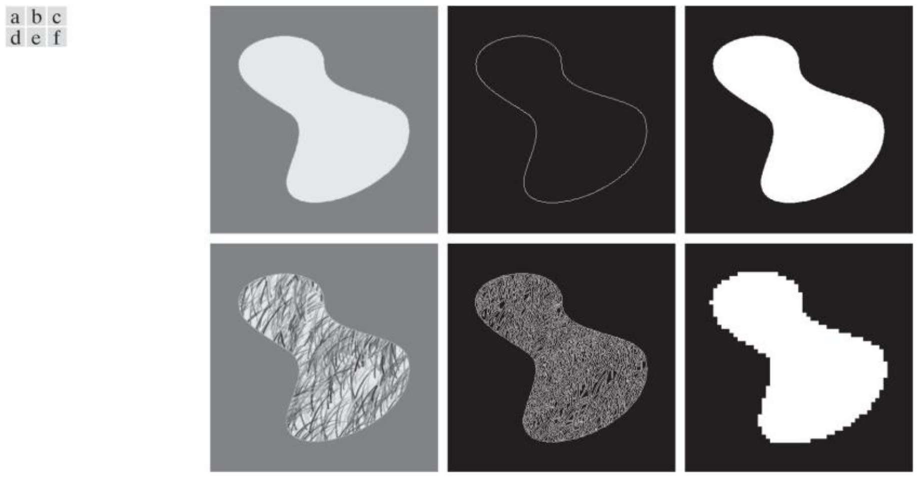


FIGURE 8.1 (a) Image of a constant intensity region. (b) Boundary based on intensity discontinuities. (c) Result of segmentation. (d) Image of a texture region. (e) Result of intensity discontinuity computations (note the large number of small edges). (f) Result of segmentation based on region properties

Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

The following three images demonstrate segmentation by region. Figure 8.1 (d) is similar to Figure 8.1 (a), other than that the inner region's intensities form a textured pattern. The outcome of computing intensity discontinuities in this image is shown in Figure 8.1 (e). Because of the many spurious intensity changes, it is hard to identify a distinctive boundary for the original image, as many of the nonnegative intensity variations are linked to the boundary, making edge-based segmentation ineffective. However, because the outer region is consistent, we only need a predicate that distinguishes among textured and constant regions to solve this segmentation problem. The standard deviation of pixel values is a measure that actually achieves this by being greater than zero in areas of the texture region and less than zero in all other regions. The result of segmenting the original image into subregions of varying sizes is depicted in Figure 8.1 (f). Each subregion was then labelled white if the standard deviation of its pixels was positive (i.e., if the predicate was TRUE), and zero if the standard deviation was negative. Due to the fact that groups of squares were labelled with the same intensity, the result has a "blocky" appearance around the region's perimeter (smaller squares would have given a smoother region boundary). Finally, keep in mind that these results also satisfy the five segmentation criteria stated at the start of this section.

Complete v/s partial segmentation

In complete segmentation

- Segmented disjoint regions uniquely correspond to objects in the input image.
- Cooperation with high computation levels that make use of problem's domain-specific knowledge is required.

In partial segmentation

- Segmented regions do not always correspond to the image object.

Discontinuity-based segmentation

In a discontinuity-based approach, an image's partitions or sub-divisions are determined by abrupt changes in the image's intensity level. We are primarily concerned with identifying isolated points, lines, and edges in an image. To do so, we employ the 3 X 3 Mask operation.

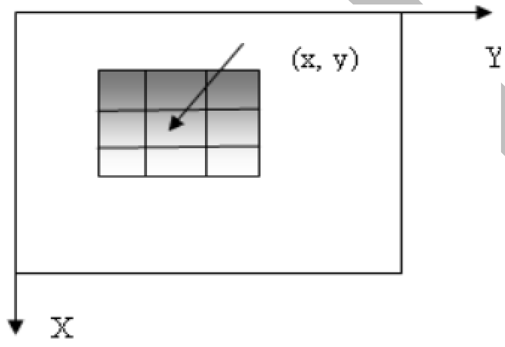


Figure 8.2. An image

$W_{-1,-1}$	$W_{-1,0}$	$W_{-1,1}$
$W_{0,-1}$	$W_{0,0}$	$W_{0,1}$
$W_{1,-1}$	$W_{1,0}$	$W_{1,1}$

Figure 8.3. 3X3 Mask

$$\boldsymbol{R = \sum_{i=1}^{1}\sum_{j=1}^{1} W_{i,j}\, f(x+1, y+1)}$$

Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

The discontinuity-based segmentation can be classified into three approaches: (1) Point detection, (2) Line detection, and (3) Edge detection.

Point, Line, and Edge Detection

This section discusses segmentation techniques that are relied on detecting sharp, localised transformations in intensity. We are focused in three types of image characteristics: isolated points, lines, and edges. Edge pixels are pixels where the image's frequency sudden modifies, while edges (or edge segments) are collections of connected edge pixels.

Edge detectors are low-level image processing tools that are used to detect pixel boundaries. A line can be thought of as a (typically) thin edge segment where the intensity of the background one on each side of the line is significantly greater or lesser than the intensity of the line pixels. Indeed, as we will explain shortly, lines outcome in what are referred to as "roof edges." Finally, an isolated point can be thought of as a foreground (background) pixel surrounded by other foreground (background) pixels.

Point Detection

In a digital image, the simplest type of discontinuity is a point. The most frequently used technique for detecting discontinuities is to apply a (n X n) mask to each point in the image. The mask is constructed in the manner depicted in Figure 2.

-1	-1	-1
-1	8	-1
-1	-1	-1

Figure 8.4. A mask for point detection

Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

The point is located at the (x, y) coordinates of the mask's centre in an image. If the equivalent value of R is

$$\| R \| > T$$

Where R denotes the mask's response at any point within the image and T denotes a non-negative threshold value. This indicates that an isolated point is detected at the specified value (x, y). This formulation is used to calculate the weighted differences between the centre point and its neighbours, as an isolated point's grey level will be significantly different than that of its neighbours [ ]. As illustrated in Figure 3, the result of the point detection mask is as follows.

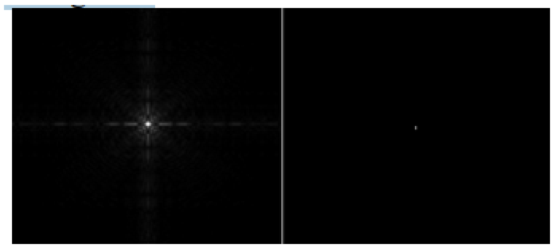


Figure 8.5. (a) Gray-scale image with a nearly invisible isolated black point (b) Image showing the detected point

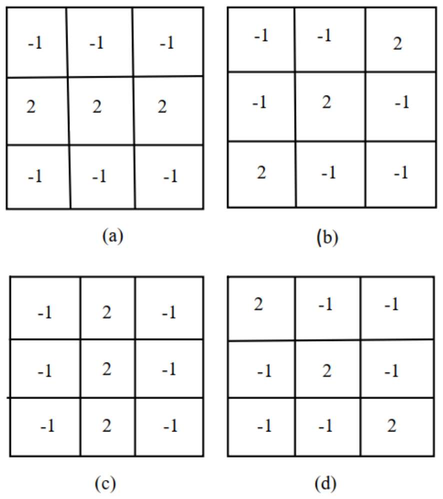
Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

Line Detection

Line detection is the next level of complexity in terms of image discontinuity in the direction of the image. A response could be computed for any point in the image that indicates the direction with which the point of a line is most closely associated. Two masks ith and jth are used for line detection. Following that, there is

$$\| R_i \| > \| R_j \|, \forall_j \neq i$$

It means that the corresponding points is more likely to be associated with a line in the direction of the mask *i*.



-1	-1	-1
2	2	2
-1	-1	-1

(a)

-1	-1	2
-1	2	-1
2	-1	-1

(b)

-1	2	-1
-1	2	-1
-1	2	-1

(c)

2	-1	-1
-1	2	-1
-1	-1	2

(d)

Figure 8.6. Line Detector masks in (a) Horizontal direction (b) 45° direction (c) Vertical direction (d) - 45° direction

Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

The position of a given pixel [] is defined by the greatest response calculation from these matrices. Figure 8.7 illustrates the result of the line detection mask.

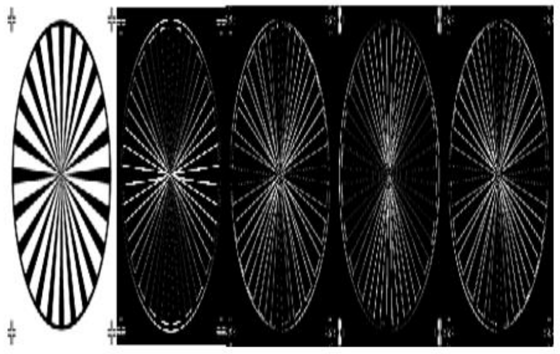


Figure 8.7. (a) Original Image (b) result showing with horizontal detector (c) with 45° detector (d) with vertical detector (e) with -45° detector

Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

We can detect lines in a specified direction using lines detector masks. For instance, we're interested in locating all lines that are one pixel thick and oriented at -45 degrees. For this, we use a digitized (binary) portion of an electronics circuit's wire-bond mask. The outcomes are depicted in Figure 8.8.

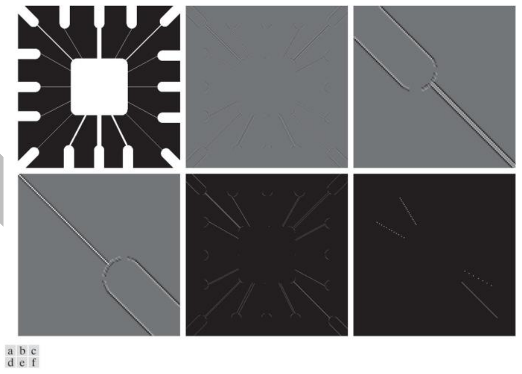


Figure 8.8. (a) Image of a wire-bond mask (b) Result of processing with the -45° detector (c) Zoomed view of the top, left region of -45° detector (d) Zoomed view of the bottom, right region of -45° detector (e) Absolute value of -45° detector (f) All points whose values satisfied the condition g >= T, where g is the image in (e)

Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

Edge detection

Due to the peculiarity of isolated points and lines of unitary pixel thickness in most practical applications, edge detection is the most frequently used technique for grey level discontinuity segmentation. An edge is a line that separates two regions of varying intensity. It is extremely useful for detecting discontinuities in images. When an image transitions from darkness to lightness or vice versa. Figure 8.9 illustrates the changes in intensity, first-order derivative, and second-order derivative.

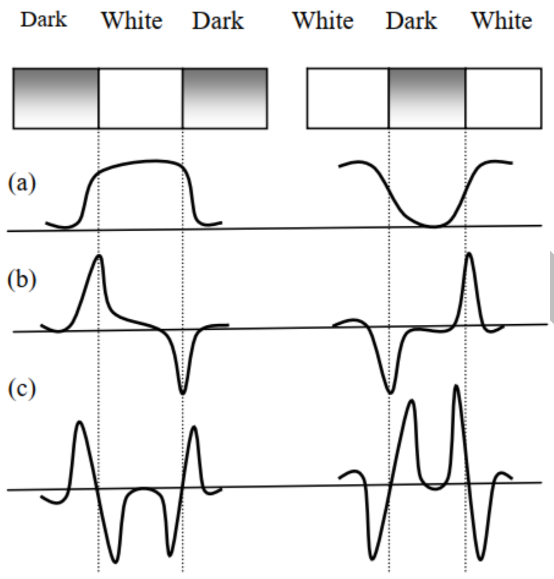


Figure 8.9. (a) Intensity profile (b) First-order derivatives (c) Second-order derivatives

Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

First-order derivatives.

First-order derivatives respond whenever there is a discontinuity in intensity levels. On the leading edge, it's positive, and on the trailing edge, it's negative. An image f(x, y) and the gradient operator ∇f are given in the following equation:

$$f = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\delta f}{\delta x} \\ \frac{\delta f}{\delta y} \end{bmatrix}$$

Strength of $\overleftarrow{\nabla f}$ is given by

$$\nabla f = \text{magnitude of } \overleftarrow{\nabla f}$$

$$= \sqrt{G_x^2 + G_y^2}$$

$$\cong \| G_x \| + \| G_y \|$$

It gives the strength of edge at location (x, y)

$$\alpha(x, y) = \tan^{-1}\frac{G_x}{G_y}$$

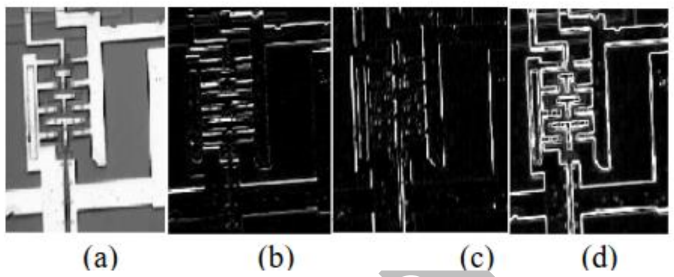


Figure 8.10. (a) Original Image (b) ‖ G_x ‖ component of the gradient along x-direction (c) ‖ G_y ‖component of the gradient along y-direction (d) Gradient Image ‖ G_x ‖ +‖ G_y ‖

Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

It is possible to calculate the gradient of an image in several ways.

Prewitt Edge operator

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

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

Figure 8.11. Masks used for Prewitt Edge operator

There is a gradient in the horizontal direction, and a gradient in the vertical direction, which is found by a mask. The mask also computes the gradient in the horizontal direction, which is found by another mask. G_x and G_y components can be found at different locations in an image by using these two masks. In this way, we can determine the strength and direction of the edge at a given location (x, y).

Sobel Edge operator

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

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

Figure 8.12. Masks used for Sobel Edge operator

Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

It calculates the average value of an image. Image noise has a negative impact on image quality. As a result, it is preferable to the prewitt edge operator because it produces a smoothing effect and reduces spurious edges caused by noise in the image.

Second-order derivatives

It is positive at the darker side and negative at the white side. It is very sensitive to noise present in an image. That's why it is not used for edge detection. But, it is very useful for extracting some secondary information i.e. we can find out whether the point lies on the darker side or the white side.

Zero-crossing: It is useful to identify the exact location of the edge where there is gradual transition of intensity from dark to bright region and vice-versa.

The derivative operators have a variety of higher-order relationships:

Laplacian operator

The Laplacian mask is given by:

0	-1	0
-1	4	-1
0	-1	0

Figure 8.13. Masks used for Laplacian operator

Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

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

If we consider the diagonal elements:

-1	-1	-1
-1	8	-1
-1	-1	-1

1	1	1
1	8	1
1	1	1

Figure 8.14. Masks used for Laplacian operator using 8-connectivity

Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

It is less sensitive to noise, and because of this, it can cause a double edge effect. However, extracting secondary information is quite helpful with it.

To reduce the effect of noise, we first apply a Gaussian operator to smooth the image, and then we use the Laplacian operator to refine it. Together, the LoG (Laplacian of Gaussian) operator is known as the Laplacian of Gaussian operation.

LoG operator

The LoG mask is given by

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

Figure 8.15. Masks used for LoG operator

Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

The Gaussian operator is given by:

$$h(x,y) = \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right)$$

Where $x^2 + y^2 = r^2$

$$\nabla^2 \boldsymbol{h} = \left(\frac{r^2 - \sigma^2}{\sigma^4}\right) \boldsymbol{exp}\left(-\frac{r^2}{2\sigma^2}\right)$$

Canny operator

It is very important method to find edges by isolating noise from the image before find edges of images, without affecting the features of the edges in the image and then applying the tendency to find the edges in the image and the critical value for threshold.

1. Convolve image f(r, c) with a Gaussian function to get smooth image g(r, c).
   f(r, c) = f(r, c) * G(r, c)
2. Apply first difference gradient operator to compute edge strength.
3. Apply non-maximal or critical suppression to the gradient magnitude.
4. Apply threshold to the non-maximal suppression image.

The result of all operators are shown

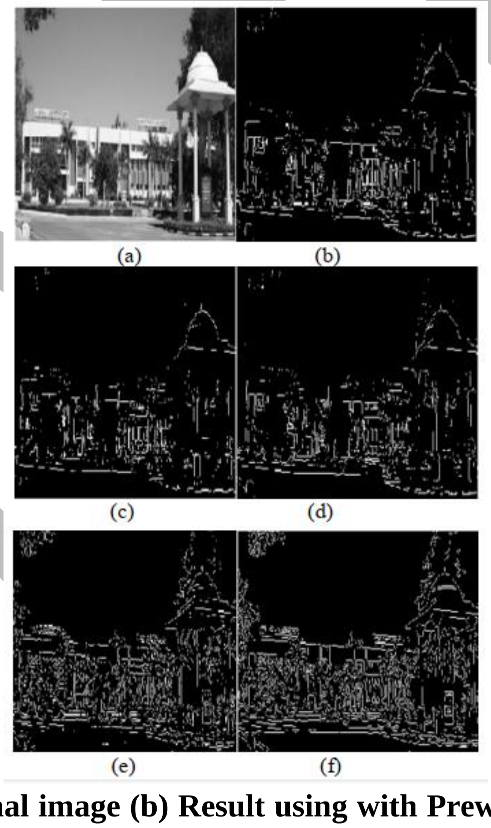


Figure 8.16. (a) Original image (b) Result using with Prewitt operator (c) Result using with Roberts operator (d) Result using with Sobel operator (e)Result using with LoG operator (f) Result using with Canny operator

Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

Marr Hildreth Edge Detector (Laplacian of Gaussian)

Neuroscience is one of the sources of inspiration for the Marr Hildreth edge detector. The filter Marr developed is a Laplacian filter. The Laplacian filter is especially susceptible to noise. Edge detection is commonly done using the Laplacian of an image, which is especially useful for revealing regions of rapid intensity change. Before applying the Laplacian filter, we should use a Gaussian filter to properly process the image. The associative property of the convolution operation applies both to the Gaussian smoothing filter and the Laplacian filter, which allows us to first perform the convolution and then do the hybrid filter operation in order to achieve the same result. The image to the right shows the results of this experiment.

The Laplacian L(x,y) of an image I(x,y):

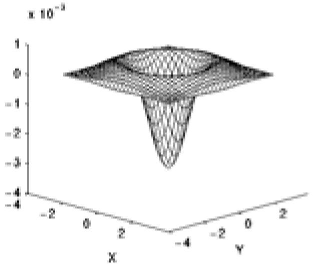
$$L(x,y) = \nabla^2 f(x,y) = \frac{\partial^2 f(x,y)}{\partial x^2} + \frac{\partial^2 f(x,y)}{\partial y^2}$$

Since Convolution is associative:

$$\nabla^2(I*G) = I * \nabla^2(G)$$

Laplacian of Gaussian:

$$\mathrm{LoG}(x,y) = -\frac{1}{\pi\sigma^4}\left[1 - \frac{x^2+y^2}{2\sigma^2}\right]e^{-\frac{x^2+y^2}{2\sigma^2}}$$



0	1	1	2	2	2	1	1	0
1	2	4	5	5	5	4	2	1
1	4	5	3	0	3	5	4	1
2	5	3	-12	-24	-12	3	5	2
2	5	0	-24	-40	-24	0	5	2
2	5	3	-12	-24	-12	3	5	2
1	4	5	3	0	3	5	4	1
1	2	4	5	5	5	4	2	1
0	1	1	2	2	2	1	1	0

Figure 8.17 a) shape of LoG with inverted hat b) 9X9 LoG mask when sigma = 1.4

Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

The shape of LoG is somewhat similar to an inverted hat. This image shows the 9X9 LoG mask when sigma = 1.4. The LoG of an image can be obtained directly by convolving the mask over the image. After obtaining the LoG of an image, what do we do?

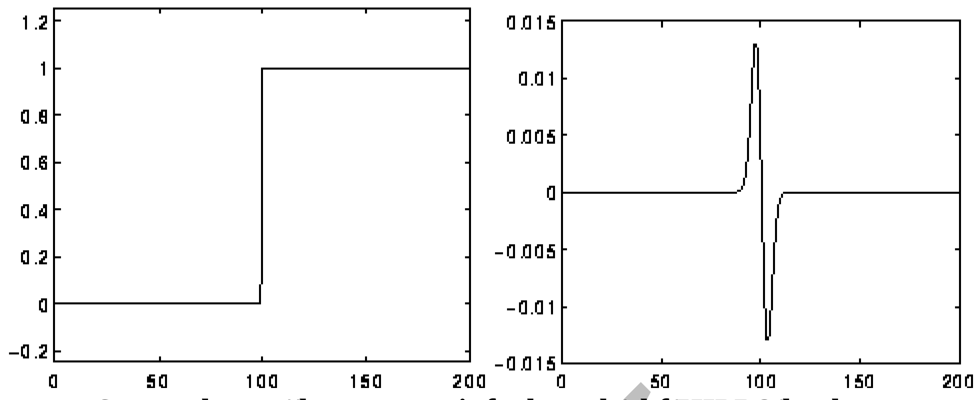

Source: **https://homepages.inf.ed.ac.uk/rbf/HIPR2/log.htm**

Figure 8.18 a) intensity variation of an image with a step edge b) LoG of the image

This image illustrates the intensity variation of an image with a step edge as it relates to the x-axis. The plot in the right side of this image illustrates the LoG of the image. When there is an edge in the image, there is a zero crossing in LoG. Since this is a simple and obvious demonstration, it shows that we can determine the edge when a zero crossing occurs.

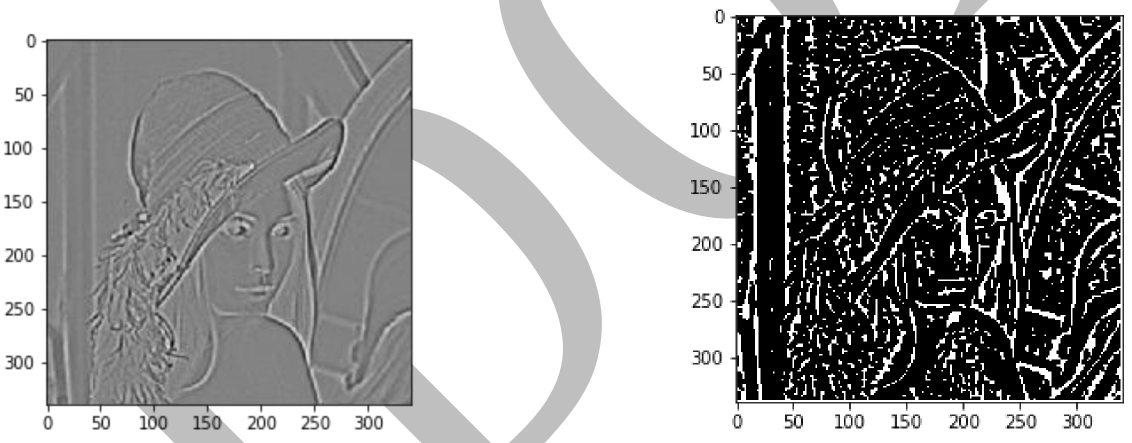


Figure 8.19 a) After applying LoG filter b) After applying threshold over zero crossing

The Canny Edge Detector

A multi-step edge detection algorithm is also known as a "canny edge detector." Detection of edges can be achieved by following the steps listed below.

1. Removal of random noise in an input image via Gaussian distribution filter
2. Gaussian filter is used to calculate the gradient magnitude, which is then used to calculate the magnitude of the image pixels along the x and y axes.
3. Considering a group of neighbors for any curve in a direction perpendicular to the given edge, suppress the non-max edge contributor pixel points.

4. Furthermore, use the Hysteresis Thresholding method to protect the pixels that are above the gradient magnitude, even if they are only slightly above it, while neglecting the ones that are lower threshold value.

Before getting too deep into the techniques below, consider the following three conclusions first showed up at by J.K. Canny who created the algorithm:

- Good Detection: To avoid getting a false positive or a false negative, the optimal detector must be reliable.
- Good Localization: The detected edges are considered to be close to true edges.
- Single Response Constraint: For each edge point, only one point must be returned by the detector.

Steps to follow during Canny Algorithm:

Noise Reduction

An edge detector is a type of high pass filter that amplifies high-frequency components while suppressing low-frequency ones. Due to the fact that both edges and noise are high-frequency components, edge detectors have a tendency to amplify noise. To avoid this, we use a low-pass filter to smooth the image. Canny accomplishes this through the use of a Gaussian filter.

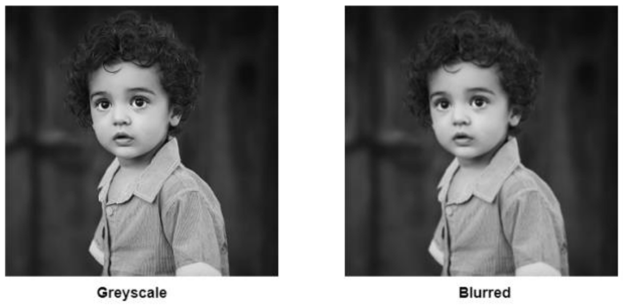


Figure 8.20

While a larger filter reduces noise, it degrades edge localization, and vice versa. In general, a value of 55 is a good choice, but this may vary according to the image.

Finding Intensity Gradient of the Image

The pixel might not be close to matching up with its neighboring pixels when the noise is present. Inappropriate edge detection may occur. In order to avoid having the same issue, we use the Gaussian filter, which is applied to the image, then convolved with the image, removing the noise, preventing the desired edges in output images.

Convoluting the Gaussian filter or kernel g(x,y) with an image I is demonstrated in the following example. Here, we want to ensure that any given pixel in the output is similar to its neighbors, and so we use the matrix [1 1 1] to maintain this similarity and remove noise.

$$S = I * g(x, y) = g(x, y) * I$$

$$g(x,y) = \frac{1}{\sqrt{2\pi\sigma}}\, e^{-\frac{x^2+y^2}{2\sigma^2}}$$

g(x,y)= Gaussian Distribution

I = input image

Derivative:

Calculate the derivative of the filter with respect to the X and Y dimensions and convolve it with I to obtain the magnitude of the gradient along the dimensions. Additionally, the image's direction can be calculated using the tangent of the angle formed by the two dimensions.

$$\nabla S = \nabla(g * I) = (\nabla g) * I$$

$$\nabla S = \begin{bmatrix} g_x \\ g_y \end{bmatrix} * I = \begin{bmatrix} g_x * I \\ g_y * I \end{bmatrix}$$

$$\nabla S = \begin{bmatrix} \frac{\partial_g}{\partial x} \\ \frac{\partial_g}{\partial y} \end{bmatrix} = \begin{bmatrix} g_x \\ g_y \end{bmatrix}$$

The convolution described above generates a gradient vector with magnitude and direction.

$$(S_{x\,,}\ S_y\ )\ Gradient\ vector$$

$$magnitude = \sqrt{\left(S_x^2 + S_y^2\right)}$$

$$direction = \theta = \tan^{-1}\frac{S_x}{S_y}$$

The following is an illustration of Gaussian Derivatives, which contribute to the edges in output images.

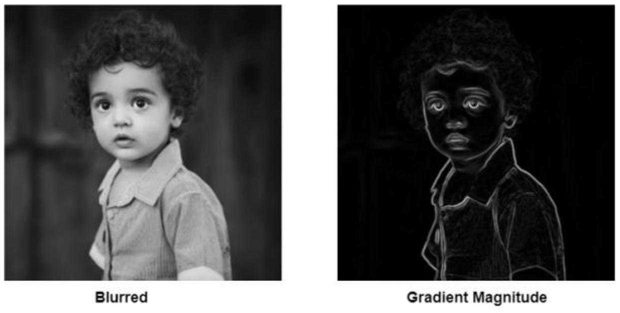


Figure 8.21 Gaussian Derivatives

Clearly, the edges remain quite blurred or thick. Keep in mind that an edge detector should produce a single precise response corresponding to the edge. As a result, we must thin the edges, or in other words, locate the largest edge. This is accomplished through the use of Non-max Suppression.

Non-Max Suppression

This is a technique for thinning the edges. This checks whether or not a pixel is a local maximum in its neighborhood in the direction of the gradient. It is retained as an edge pixel if it is a local maximum; otherwise, it is suppressed.

Neighboring pixels are located in the horizontal, vertical, and diagonal directions (0°, 45°, 90°, and 135°) for each pixel. As a result, we must round off the gradient direction at each pixel to one of these values, as illustrated below.

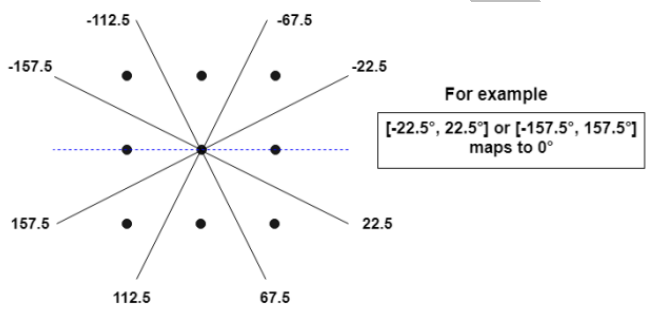


Figure 8.22 Neighboring pixels are located in the horizontal, vertical, and diagonal directions (0°, 45°, 90°, and 135°) for each pixel.

After rounding, we'll compare each pixel's value to the gradient direction's two neighboring pixels. If that pixel signifies a local maximum, it is managed to retain as an edge pixel; otherwise, it is suppressed. Thus, only the most substantial responses will remain.

Consider the following.

Assume the gradient direction is 17 degrees for pixel 'A.' we will round to 0 degrees because 17 is closer to 0. Then, using the rounded gradient direction, we select neighboring pixels (See B and C in below figure). If A has a greater intensity value than B and C, it is retained as an edge pixel; otherwise, and it is suppressed.

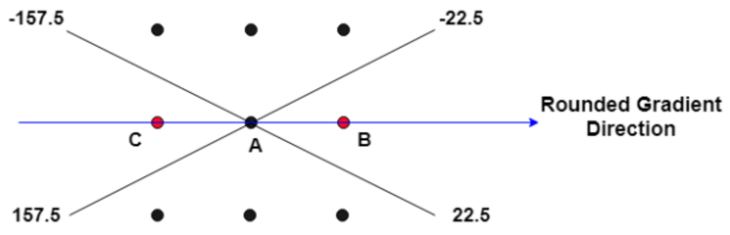


Figure 8.22 Gradient direction

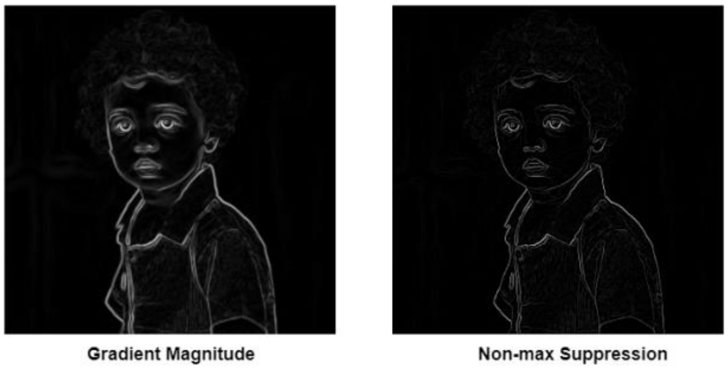


Figure 8.23 a) Gradient Magnitude b) Non-max suppression

Clearly, the edges are thinned, but some are brighter than others. The brighter ones can be considered strong edges, while the lighter ones may be true edges or may be noise.

Hysteresis Thresholding:

Non-max suppression produces an image with a far more accurate depiction of its real edges. However, as can be seen, certain edges are brighter than others. The brighter ones can be considered strong edges, whereas the lighter ones may be true edges or may be noise. Canny employs hysteresis thresholding to resolve the question of "which edges are truly edges and which are not." We define two thresholds in this section: 'High' and 'Low.'

- Any edges with intensity greater than 'High' are the sure edges.
- Any edges with intensity less than 'Low' are sure to be non-edges.
- The edges between 'High' and 'Low' thresholds are classified as edges only if they are connected to a sure edge otherwise discarded.

Let's take an example to understand

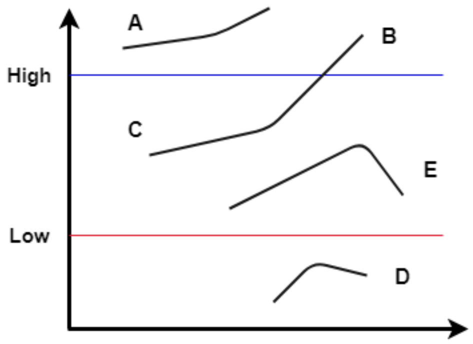


Figure 8.24 'High' and 'Low threshold

Here, A and B are sure-edges as they are above 'High' threshold. Similarly, D is a sure non-edge. Both 'E' and 'C' are weak edges but since 'C' is connected to 'B' which is a sure edge, 'C' is also considered as a strong edge. Using the same logic 'E' is discarded. This way we will get only the strong edges in the image.

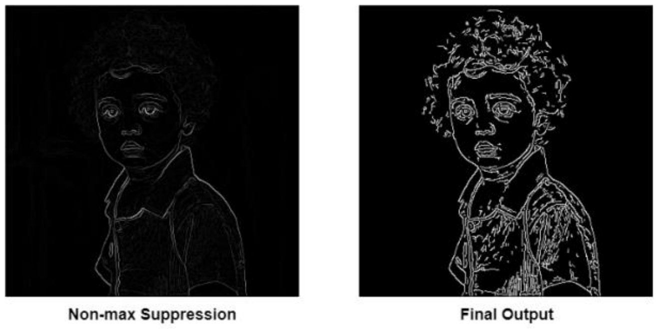


Figure 8.25 a) Non-max suppression b) Final Output

Local Processing

In this type of analysis, two primary properties are used to determine the similarity of edge pixels:
1. The strength of the response of the gradient operator used to generate the edge pixel.

2. The gradient's direction

Within a small neighbourhood, for example, 3x3, 5x5, all points with shared properties are connected. If both the magnitude and direction criteria are met, a point (x',y') in the vicinity of (x,y) is linked to the pixel at (x,y).

$$|\nabla f(x', y') - \nabla f(x, y)| \leq \text{Threshold Tm}$$

$$|\alpha(x', y') - \alpha(x, y)| \leq \text{Threshold}$$

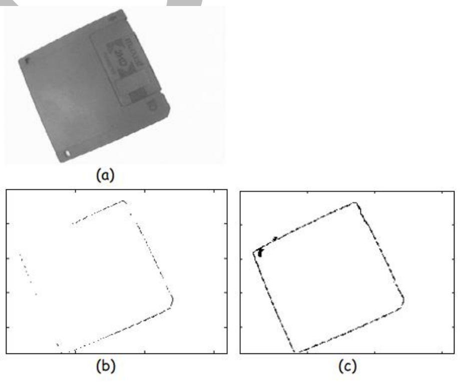


Figure 8.26 (a) Original image ; (b) detection result without local processing ; (c) detection result with local processing (Tm=0.15 x max(|∇f|) and Td=pi/9

Global Processing Using the Hough Transform

The Hough transform can be used to link pixels and identify curves. In the polar coordinate system, the straight line denoted by y=mx+c can be expressed as, ρ = xcos(θ)+ ysin(θ) …………………..(i)

Where, denotes a vector extending from the origin to the point on the straight line y=mx+c that is closest to the origin. This vector will be perpendicular to the line from the origin to the point closest to the line, as illustrated below.

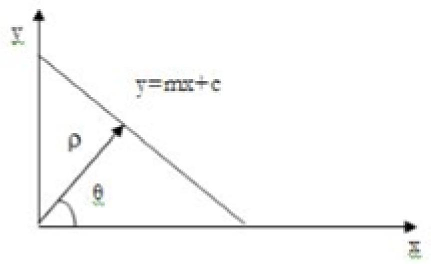


Figure 8.27 Vector perpendicular to the line from the origin to the point closest to the line

Any line in the x, y plane pertains to a point in the two-dimensional space defined by the parameter and θ. The Hough transform of a straight line in the x,y plane is a particular point in the ρ, θ , space, and these points must fulfil the specified equation with the constants x1,y1. Thus, the locus of all such lines in the x, y plane corresponds to the location of the particular sinusoidal curve in, space.

Assume we have the edge points xi,yi that are parallel to the straight line with parameters ρ0,θ0. Each edge point corresponds to a sinusoidal curve in, space, but these curves must intersect at 0,ρ0,θ0. Due to the fact that this is a shared line.

Consider the following equation for a line: y1= ax1+b

By varying the values of a and b in this equation, an infinite number of lines can pass through this point (x1,y1).

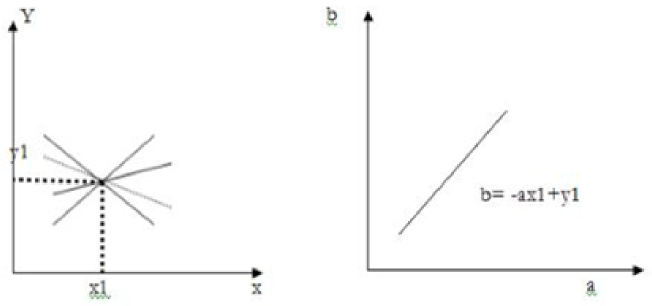


Figure 8.28 line: y1= ax1+b

If we rewrite the equation as follows:

B= -ax1+y1

A straight line is obtained if we consider the ab plane instead of the xy plane (xi,yi). This entire line in the ab plane is due to a single point in the xy plane and different values of and b. Another point (x2, y2) in the xy plane can be considered here. Its slope intercept equation is,

Y2= ax2+ b………………….(1)

This is what we get when we write it in terms of the ab plane.

B= -ax2+y2………………..(2)

In the ab plane, this is a different line. In the ab plane, these two lines will intersect only if they are part of a straight line in the x-axis plane. (a',b') is the coordinates of the point of intersection in the ab plane. A slope-intercept equation y=a'x+b' can be written as follows: y=a'x+b'. This line passes through the points (x1,y1), and (x2,y2) in the y-axis.

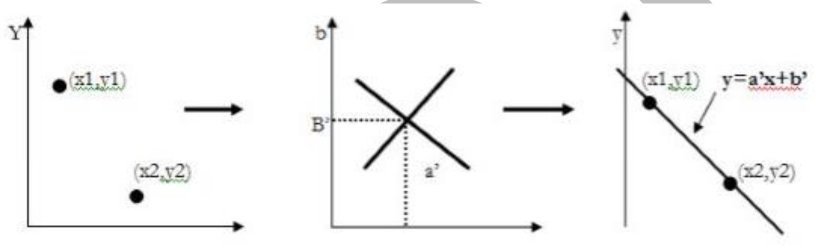


Figure 8.29 line passes through the points (x1,y1), and (x2,y2) in the y-axis.

8.2 Thresholding

In terms of image segmentation, thresholding is the simplest method. Using thresholding to create binary images from a grayscale image

The Basics of Intensity Thresholding

Assume that the intensity histogram in Fig. 8.30 (a) refers to an image, f(x, y), that is comprised of light objects on a dark background, and that the intensity values of the object and background pixels are sorted into two dominating modes. One simple method of distinguishing the objects from the background is to set a threshold, T, which differentiates the two modes. Then, any point (x, y) in the image at which the object is located is referred to as an object point. In any other case, the point is referred to as a background point. With another way of saying it, the segmented image, indicated by g(x, y), can be represented as

A coordinate f(x, y) is a measure of the intensity of f. (x, y).

$$g(x,y) = \begin{cases} 1 \; if \; f(x,y) > T \\ 0 \; if \; f(x,y) \leq T \end{cases}$$

Global thresholding occurs when T is a constant that is applied to a whole image. We use the term variable thresholding when the value of T varies over an image. As a result, the value of T at every given location (x, y) in an image depends on the features of the neighbourhood of (x, y) (for example, the average intensity of the pixels in the neighborhood). The term dynamic thresholding or adaptive thresholding is typically used when T relies on the spatial coordinates (x, y). These terms are not used universally.

Figure 8.30(b) illustrates a more complex and difficult thresholding problem involving a histogram with three dominant modes, for example, two types of light objects on a dark background. Multiple thresholding categorizes a point (x, y) as relating to the background if f(x,y) <T_1 , to one object if T_1 ≤f(x,y)≤T2 and to the other object class if f(x,y)> T2. That is to say, the segmented image is specified by

$$g(x,y) = \begin{cases} a \text{ if } f(x,y) < T_1 \\ b \ \ T_1 \le f(x,y) \le T_2 \\ c \ f(x,y) > T_2 \end{cases}$$

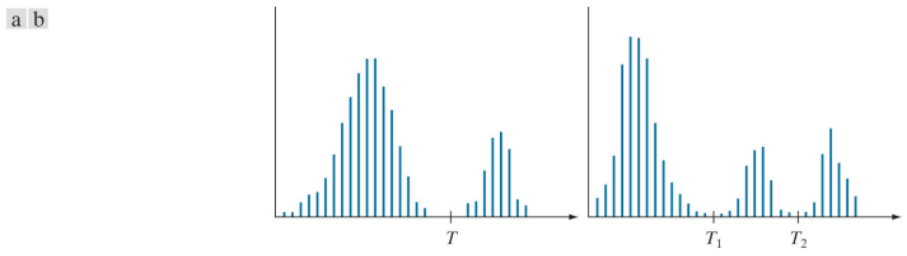


FIGURE 8.30 Intensity histograms that can be partitioned (a) by a single threshold, and (b) by dual thresholds

(Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

Basic Global Thresholding

Whenever the intensity distributions of objects and background pixels are substantially distinct it is possible to utilize a single (global) threshold that applies to the whole image In many other applications, there is typically sufficient variation across images so, even though global thresholding is a reasonable method an algorithm capable of predicting the threshold value for each image is needed. For this aim, the iterative approach described below can be used:

1. An initial threshold (T) is selected; this could be accomplished at random or by any other method that is preferred.
2. In the same manner as explained before, the image is segmented into object and background pixels, yielding two sets of pixels:
   a. G1 = {f(m,n):f(m,n)>T} (object pixels)
   b. G2 = {f(m,n):f(m,n) T} (background pixels) (note, f(m,n) is the value of the pixel located in the mth column, nth row)
3. The average of each set is computed.

   a. m1 = average value of G1
   b. m2 = average value of G2
4. A new threshold is created that is the average of m1 and m2
a. T‘ = (m1 + m2)/2
5. Return to step two, this time using the new threshold computed in step four, and repeat the process until the new threshold matches the one before it (i.e., until convergence has been achieved).

Segmentation using the preceding iterative algorithm is illustrated in Figure 8.31. The original image is shown in Figure 8.31 (a), and the image histogram, which demonstrates a distinct valley, is shown in Figure 8.31(b). The basic global algorithm produced the threshold after three iterations, starting with T equal to the image's average intensity, and using Figure 8.31 (c) to segment the original image. As expected given the histogram's distinct separation of modes, the segmentation of object and background was flawless.

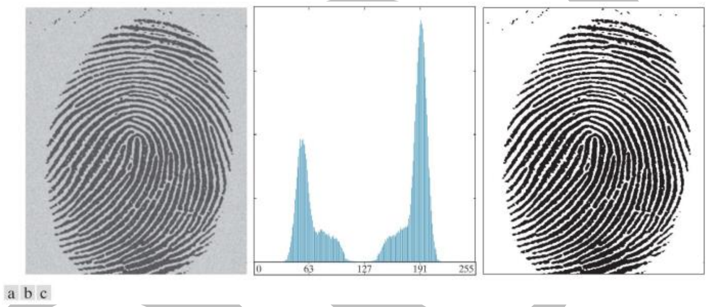


FIGURE 8.31(a) Noisy fingerprint. (b) Histogram. (c) Segmented result using a global threshold (thin image border added for clarity). (Original image courtesy of the National Institute of Standards and Technology.).

(Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

Optimum Global Thresholding Using Otsu’s Method

Let us begin by comprehending Otsu's approach. The method processes the image histogram, segmenting the objects based on the class variance minimization. Generally, this technique produces the desired results when dealing with bimodal images. The histogram of such an image contains two distinct peaks that correspond to distinct ranges of intensity values.

The fundamental idea is to divide the image histogram into two clusters using a threshold defined by the weighted variance of these classes, denoted by $\sigma_w^2(t)$

The entire computation equation can be summarized as follows:$\sigma_w^2(t) = w_1\,(t)\,\sigma_1^2(t) + w_2\,(t)\,\sigma_2^2(t)$, where $w_1\,(t)$ and $w_2\,(t)$are the probabilities of the two classes divided by a threshold t that is between 0 and 255 inclusively.

As demonstrated in Otsu's paper, there are usually two ways to determine the threshold. The first objective is to minimize the within-class variance defined above $\sigma_w^2(t)$, while the second objective is to maximise the between-class variance using the following expression:

$\sigma_b^2(t) = w_1(t)w_2(t)[\mu_1(t) - \mu_2(t)]^2$Where μ_i denotes the mean of class i.

The probability P is computed for every pixel value in two distinct clusters C_1 and C_2 using the cluster probability functions

$$w_1(t) = \sum_{i=1}^{t} P(i)$$

$$w_2(t) = \sum_{i=t+1}^{I} P(i)$$

It's important to note that the image can be viewed as an intensity function f(x, y), with gray-level values .The number of pixels that have a specified gray-level I is signified by i. The image's as a whole pixel count is n.

Thus, the probability of encountering gray-level I is:

$$P(i) = \frac{n_i}{n}$$

The C_1 pixel intensity values are in the range [1, t], while the C_2 pixel intensity values are in the range [t + 1, I], where I is the maximum pixel value (255).

The following phase is to achieve the means for C_1 and C_2, which are denoted appropriately by $\mu_1(t)$ and $\mu_2(t)$):

$$\mu_1(t) = \sum_{i=1}^{t} \frac{iP(i)}{w_1(t)}$$

$$\mu_2(t)) = \sum_{i=t+1}^{I} \frac{iP(i)}{w_2(t)}$$

Now distinctly remember the above equation for the weighted variance within classes. We will determine the remaining components (σ_1^2, σ_2^2) by combining all of the above-mentioned ingredients:

$$\sigma_1^2(t) = \sum_{i=1}^{t} [i - \mu_1\ (t)]^2\ \frac{iP(i)}{w_1(t)}$$

$$\sigma_2^2(t)) = \sum_{i=t+1}^{I} [i - \mu_2\ (t)]^2 \frac{iP(i)}{w_2(t)}$$

It's important to note that if the threshold is selected wrongly, the variance of certain class will be quite large. To obtain the total variance, we essentially have to add the variances within and between classes: $\sigma_T^2 = \sigma_w^2(t) + \sigma_b^2(t)$, where $\sigma_b^2(t) = w_1(t)w_2(t)[\mu_1(t) - \mu_2(t)]^2$. The image's total variance (σ_T^2) is independent of the threshold.

Thus, the pipeline of the general algorithm for the between-class variance maximization option can be represented as follows:

1. calculate the histogram and intensity level probabilities
2. initialize $w_1(0)$, $\mu_i(0)$
3. iterate over possible thresholds: t = 0,..., max_intensity
   - update the values of w_i , μ_i, where w_ii is a probability and μ_i is a mean of class i
   - calculate the between-class variance value $\sigma_b^2(t)$
4. the final threshold is the maximum $\sigma_b^2(t)$ value

Otsu's Binarization Application

The authors propose an enhanced Otsu's method as one approach for estimating the location of underwater landmarks. The advantages of this approach are that it allows for precise real-time segmentation of underwater features and has been demonstrated to perform better than threshold segmentation methods. Consider its concept in greater detail by examining the side-scan sonar (SSS) shipwreck image provided in the article. Modern SSS systems are capable of imaging a large area of the sea floor in two dimensions and producing realistic images. Thus, their background contains regions of sludge and aquatic animals in the form of spots that are typically equal to 30 pixels in diameter.

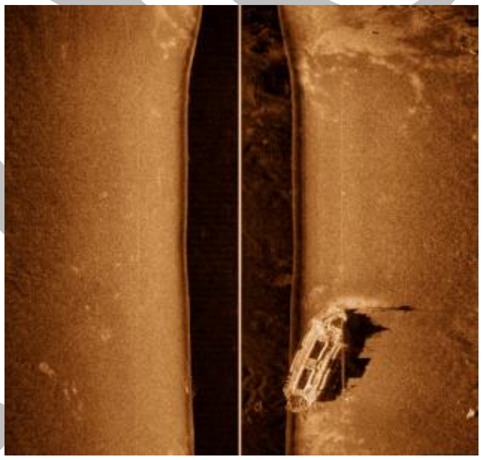


FIGURE 8.32: SSS sea bottom image

They distort proper image processing because their grey level is similar to that of certain zones of foreground objects. As shown below, the segmented image produced by the classical Otsu technique contains these artefacts:

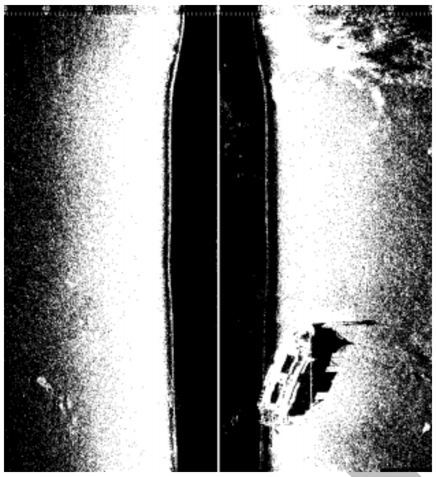


FIGURE 8.33: SSS sea bottom image

To address this spot challenge, a technique is based on Otsu's binarization was developed to restrict the search range of the suitable segmentation threshold for foreground object division. The following is the improved Otsu's method pipeline:

1. Calculate Otsu's threshold T
2. Calculate N_{30} using Canny edge detection
3. if $N_{30} > 300$, calculate final T value.

As a result, the wrecked ship stands out clearly from the background:

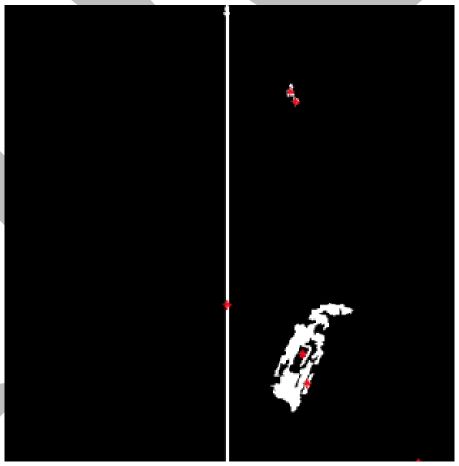


FIGURE 8.34: Improved Otsu's result

Using Image Smoothing to Improve Global Thresholding

The image from Figure 8.35 (a), the histogram is shown in Figure 8.35 (b), and the image is shown in Figure 8.35 (c) after it has been thresholded using Otsu's method. Each black point in the white region and each white point in the black region represents a thresholding error, indicating that the segmentation was a complete failure. The result of smoothing the noisy image with a size averaging kernel is shown in Figure 8.35 (d), and the histogram is shown in Figure 8.35 (e).

Smoothing improves the shape of the histogram, and we would expect the smoothed image's thresholding to be nearly perfect. As illustrated in Figure 8.35 (f), this is the case. The blurring of the boundary between object and background resulted in the slight distortion of the object-background boundary in the segmented, smoothed image. Indeed, the more aggressively we smooth an image, the more boundary errors in the segmented result we should anticipate.

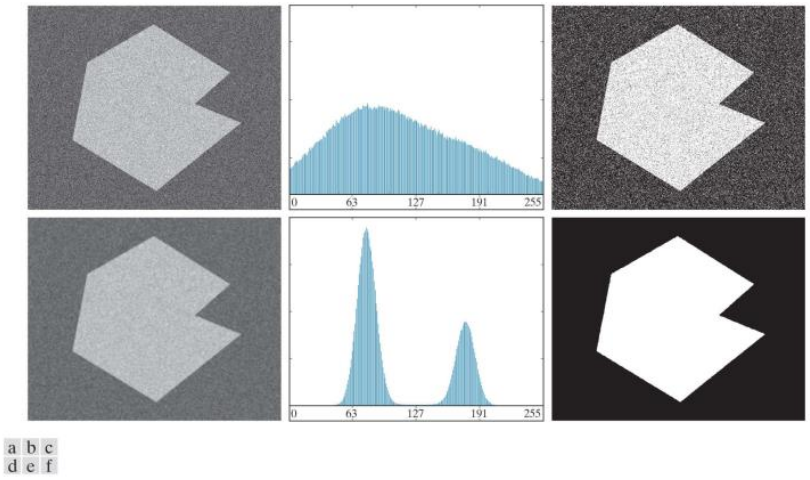


FIGURE 8.35 (a) Noisy image (c) and (b) its histogram. (c) Result obtained using Otsu's method. (d) Noisy image smoothed using an averaging kernel and (e) its histogram. (f) Result of thresholding using Otsu's method

(Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

Following that, we examine the effect of significantly shrinking the foreground region in comparison to the background. The result is depicted in Figure 8.36 (a). In this image, the noise is additive Gaussian with a mean of zero and a standard deviation of ten intensity levels (as opposed to 50 in the previous example). As illustrated in Fig. 8.36 (b), the histogram lacks a clear valley, implying that segmentation will fail, as confirmed by the result in Fig. 8.36 (c). The image smoothed with a size averaging kernel is shown in Figure 8.36 (d), and the corresponding histogram is shown in Figure 8.36 (e). As expected, the net effect was to narrow the histogram's spread, but the distribution remained unimodal. As illustrated in Fig. 8.36 (f), segmentation failed once more. The reason for the failure is that the region is so small that its contribution to the histogram is negligible in comparison to the noise-induced intensity spread. In these instances, the following section's approach is more likely to succeed.

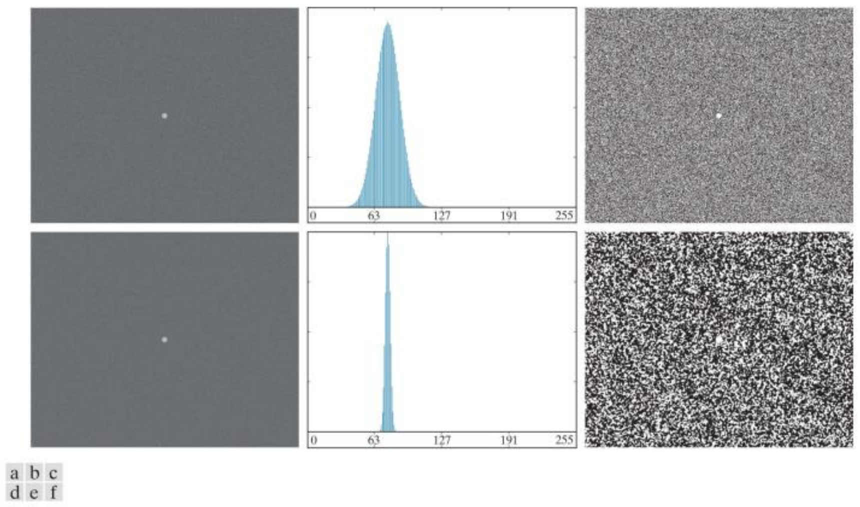


FIGURE 8.36 (a) Noisy image and (b) its histogram. (c) Result obtained using Otsu's method. (d) Noisy image smoothed using a averaging kernel and (e) its histogram. (f) Result of thresholding using Otsu's method. Thresholding failed in both cases to extract the object of interest

(Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

Using Edges to Improve Global Thresholding

The foreground and background of an image are a necessary condition for segmenting it using a global threshold. There is sufficient space between the histogram peaks, such that by adjusting the threshold between the gaps, the foreground can be distinguished from the background. OneOkHistograms, or histograms suitable for global threshold segmentation, frequently exhibit the following properties:

- tall
- narrow
- symmetric
- separated by deep valleys

However the human eye appears to have a high contrast between foreground and background at times, segmenting an image with a global threshold is not easy. Determine the threshold adaptively, because the foreground or background content is excessive, swamping the contribution of the other party to the grayscale histogram. This experiment is designed to address this issue. The central idea to solve this problem is Stat histogram only for the place where the foreground and the background are adjacent. And to find neighboring places can often be through magnitude of gradient of image or Laplacian operator get.

The following algorithm summarizes the preceding discussion, where f(x, y) denotes the input image:

1. using any of the methodologies, evaluate an edge image as the magnitude of the gradient or the absolute value of the Laplacian of f(x, y).

2. Enter a value for the threshold, T.

3. Using T from Step 2, threshold the image from Step 1 to create a binary image, $g_T(x, y)$

This image is used as a mask in the subsequent step to select pixels from f(x, y) that correspond to the mask's "strong" edge pixels.

4. Create a histogram by excluding the pixels in f(x, y) that correspond to the locations of the 1-valued pixels in $g_T(x, y)$

5. Using the histogram created in Step 4, globally segment f(x, y) using, for example, Otsu's method.

If T is set to a value less than the minimum value of the edge image, it will contain only 1's, implying that the image histogram will be computed using all pixels in f(x, y). In this particular instance, the previous algorithm performs global thresholding on the original image's histogram. It is common practice to specify T as a percentile, which is typically set high (e.g., in the high 90's) to ensure that only a small subset of the gradient/Laplacian image pixel is used in the computation. The following examples demonstrate the previously discussed concepts. The gradient is used in the first example, while the Laplacian is used in the second. Using either approach, similar results can be obtained in both examples.

EXAMPLE 1: Using edge information based on the gradient to improve global thresholding.

The critical point is the images and histograms in Figures 8.37(a) and (b) correspond to those in Figure 8.36. As you saw, smoothing accompanied by thresholding was unable to segment this image. The purpose of this example is to demonstrate how to solve a problem using edge information. The mask image in Figure 8.37(c) was created using a gradient magnitude image thresholded at the 99.7 percentile. The image in Figure 8.37 (d) is the result of multiplying the mask by the input image. The histogram in Figure 8.37(e) represents the nonzero elements in Figure 8.37(d). Take note that this histogram possesses the critical characteristics discussed previously; namely, it contains reasonably symmetrical modes separated by a deep valley. Thus, whereas the histogram of the original noisy image did not indicate the possibility of successfully thresholding, the histogram in Fig. 8.37(e) indicates that it is possible to successfully threshold the small object from the background. As illustrated in Fig. 8.37(f), this is the case. This image was created by applying Otsu's to the noisy image in Fig. 8.37 (a) and then applying the Otsu threshold globally to the noisy image. The outcome is nearly flawless generate an appropriate derivative image.

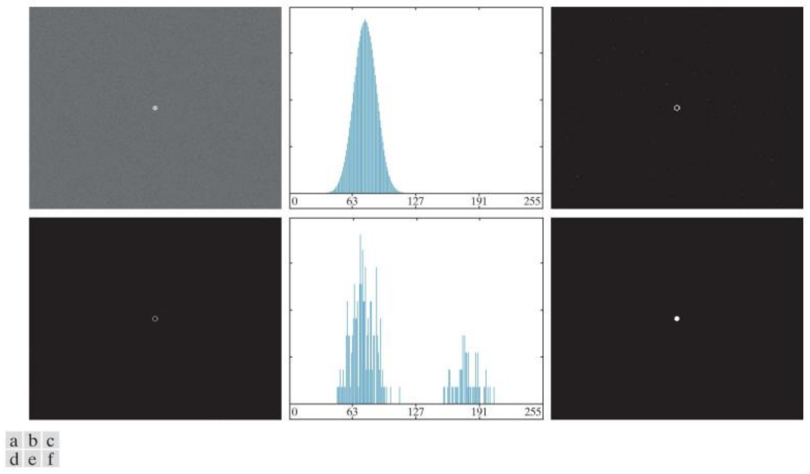


FIGURE 8.37(a) Noisy image from Fig. 8.36(a) and (b) its histogram. (c) Mask image formed as the gradient magnitude image thresholded at the 99.7 percentile. (d) Image formed as the product of (a) and (c). (e) Histogram of the nonzero pixels in the image in (d). (f) Result of segmenting image (a) with the Otsu threshold based on the histogram in (e). The threshold was 134, which is approximately midway between the peaks in this histogram

(Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

EXAMPLE 2: Using edge information based on the Laplacian to improve global thresholding.

We take into account a much more complicated thresholding problem in this example. Figure 8.38 (a) illustrates an 8-bit image of yeast cells on which we wish to perform global thresholding in order to achieve the regions pertaining to the bright spots. As a starting point, Fig. 8.38 (b) depicts the image histogram, and Fig. 8.38 (c) depicts the result obtained by applying Otsu's method directly to the image. As can be seen, Otsu's method fell short of the original objective of detecting bright spots. While the method was successful in isolating several cell regions, several of the segmented regions on the right were actually joined. The Otsu method determined a threshold of 42 and a separability measure of 0.636.

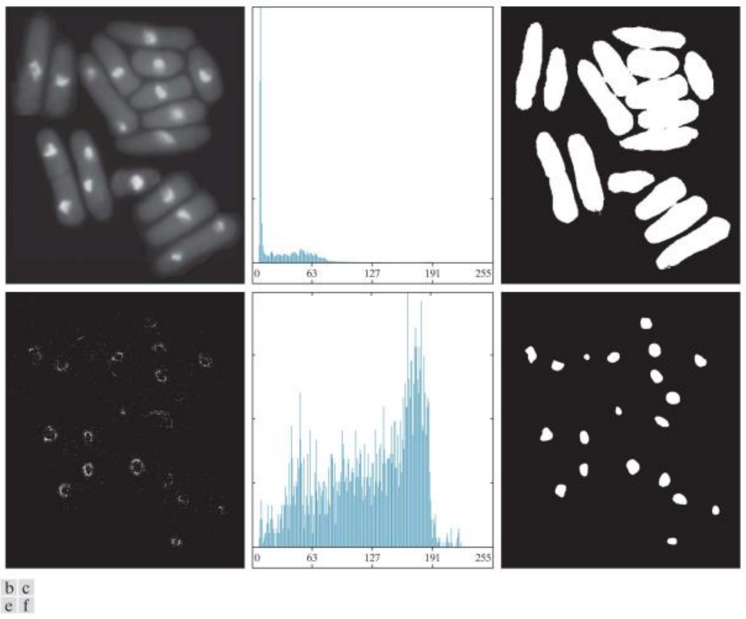


FIGURE 8.38 (a) Image of yeast cells. (b) Histogram of (a). (c) Segmentation of (a) with Otsu's method using the histogram in (b). (d) Mask image formed by thresholding the absolute Laplacian image. (e) Histogram of the nonzero pixels in the product of (a) and (d). (f) Original image thresholded using Otsu's method based on the histogram in (e).

(Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

The mask image $g_T(x,y)$is shown in Figure 8.38 (d). It was created by calculating the exact value of the Laplacian image and then thresholding it with T set to 115 on an intensity scale ranging from 0 to 255. T equals nearly to the 99.5 percentile of the values in the exact Laplacian image, and thus thresholding at this level results in a limited set of pixels, as illustrated in Fig. 8.38 (d). As anticipated from the previous section, the points cluster close to the edge of the bright spots in this image. The histogram in Figure 8.38 (e) represents the nonzero pixels in the product of (a) and (b) (d). Finally, Fig. 8.38 (f) illustrates the outcome of dividing the actual image globally using Otsu's method based on the histogram in Fig. 8.38 (e). This result is consistent with the locations of the image's bright spots. The Otsu method calculated a threshold of 115 and a separability measure of 0.762, which are both greater than the values obtained using the original histogram. We can even improve the segmentation of complete cell regions by varying the percentile at which the threshold is set. For instance, Fig. 8.38 illustrates the result obtained using the same procedure as described previously, but with the threshold set to 55, or approximately 5% of the maximum value of the absolute Laplacian image. This value is in the 53.9 percentile of the values contained in that image. This result is clearly superior to that shown in Fig. 8.38 (c) when Otsu's method is used in conjunction with the histogram of the original image.

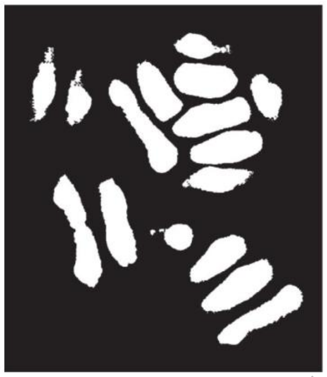


FIGURE 8.39 Image in Fig. 8.38 (a) segmented using the same procedure as explained in Figs. 8.38 (d) through (f), but using a lower value to threshold the absolute Laplacian image.

(Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

Multiple Thresholds

We have thus far concentrated on image segmentation via a single global threshold. Otsu's method is extensible to an infinite number of thresholds due to the fact that the separability way of measuring upon which it is based is also extensible to an infinite number of classes. The between-class variance generalizes to the expression in the case of K classes.

$$\sigma_B^2 = \sum_{k=1}^{k} P_k(\, m_k - m_G)^2$$

Where

$$P_k = \sum_{i \epsilon C_k} pi$$

And

$$m_k = \sum_{i \epsilon C_k} ip_i$$

As previously stated, the global mean is m_G . The K classes are separated by K −1 thresholds whose values $k_1^*, k_{2\,,}^* \ldots., k_{K-1}^*$

$$\sigma_B^2\left(k_1^*, k_{2\,,}^* \ldots., k_{K-1}^*\right) = \max_{0<k_1<k_2<\cdots k_{K-1}<L-1} \sigma_B^2\,(k_1\,, k_2, \ldots, k_{K-1})$$

Even though this outcome holds true for any number of classes, it ends up losing implying as the number of classes increases due to the fact that we have been dealing with a single variable (intensity). Indeed, the variance between classes is frequently expressed in terms of multiple variables expressed as vectors .In practice, whenever there is reason to assume that the issue can be resolved appropriately with two thresholds, multiple global thresholding is considered an appropriate approach. Usually, applications requiring more than two thresholds are solved using a combination of intensity values and other parameters. Rather than that, additional descriptors (for example, colour) are used, and the application is recast as a pattern recognition problem.

The between-class variance for three classes consisting of three intensity intervals (separated by two thresholds) is given by:

$$\sigma_B^2 = P_1(m_1 - m_G)^2 + P_2(m_2 - m_G)^2 + P_3(m_3 - m_G)^2$$

Where

$$P_1 = \sum_{i=0}^{k_1} P_i$$

$$P_2 = \sum_{i=k_1+1}^{k_1} P_i$$

$$P_3 = \sum_{i=k_2+1}^{L-1} P_i$$

And

$$m_1 = \frac{1}{P_1} \sum_{i=0}^{k_1} P_i$$

$$m_2 = \frac{1}{P_2} \sum_{i=k_1+1}^{k_1} P_i$$

$$m_3 = \frac{1}{P_3} \sum_{i=k_2+1}^{L-1} P_i$$

The following relationships also hold:

$$P_1 m_1 + P_2 m_2 + P_3 m_3 = m_G$$

And

$$P_1 + P_2 + P_3 = 1$$

The three classes are separated by two thresholds whose values k_1^* and k_2^* maximize

$$\sigma_B^2(k_1^*, k_2^*) = \max_{0<k_1<k_2<L-1} \sigma_B^2(k_1, k_2)$$

Algorithm

(1) Let $k_1 = 1$

(2) Increment k_2 through all its values greater than k_1 and less than L − 1

(3) Increment k_1 to its next value and increment k_2 through all its values greater than k_1 and less than L − 1

(4) Repeat until $k_1 = \text{L} - 3$

This results in a 2-D array $\sigma_B^2(k_1, k_2)$, after which k_1^* and k_2^* that correspond to the maximum value in the array, are selected

Segmentation is as follows: $g(x,y) = \begin{cases} a, & if\ f(x,y) \le k_1^* \\ b, & if\ k_1^* < f(x,y) \le k_2^* \\ c, & if\ f(x,y) > k_2^* \end{cases}$

Separability measure $: \eta(k_1^*, k_2^*) = \frac{\sigma_B^2(k_1^*, k_2^*)}{\sigma_G^2}$

EXAMPLE 3: Multiple global thresholding

Figure 8.40 (a) depicts an iceberg. The goal of this example is to segment the image into three distinct regions: the dark background, the illuminated portion of the iceberg, and the shadowed portion. The image histogram in Fig. 8.40 (b) demonstrates that two thresholds are required to resolve this problem. The procedure described previously resulted in the thresholds shown in Fig. 8.40 (b), which are located near the centres of the two histogram valleys. The segmentation shown in Figure 8.40 (c) is the result of using these two thresholds. The measure of separability was 0.954. The primary reason this example worked so well is that the histogram contains three distinct modes separated by fairly large, deep valleys. However, we can do even better by utilizing superpixels.

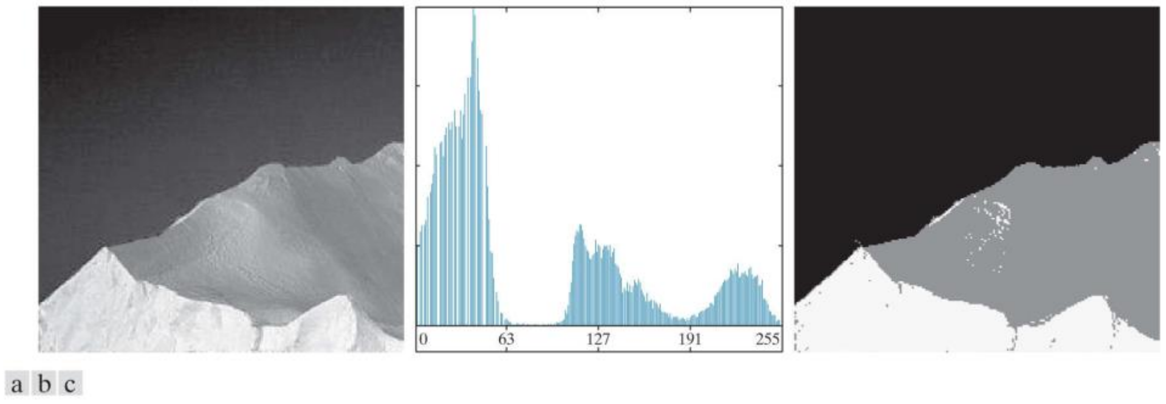


FIGURE 8.40 (a) Image of an iceberg. (b) Histogram. (c) Image segmented into three regions using dual Otsu thresholds.

(Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

Variable Thresholding

Noise and non uniform illumination have a significant impact on the performance of a thresholding algorithm. We demonstrated how image smoothing and utilize of edge information can be extremely beneficial. However, there are times when this type of preprocessing is either impractical or ineffective in addressing the problem, to the point where none of the thresholding methods discussed thus far can resolve the problem. In those kind of cases, as we will demonstrate in the discussion that follows, the next level of thresholding complexity involves variable thresholding.

Variable Thresholding Based on Local Image Properties

A fundamental concept to variable thresholding is to calculate a threshold at each point in the image, (x, y), depending on one or more required characteristics in the image's neighborhood (x, y). While this may appear to be a lengthy process, advanced algorithms and hardware allow rapid neighborhood processing, particularly for basic procedures such as logical and arithmetic operation.

The mean and standard deviation of the pixel values in the neighborhood of each point in an image are being used to illustrate the method. Since these two quantities are descriptors of average intensity and contrast, they are helpful in determining local thresholds. Let m_{xy} and σ_{xy} signify the mean and standard deviation of the set of pixel values in an image's neighborhood$S_{x,y}$, centred on the coordinates (x, y). The following are examples of common types of variable thresholds that are determined by the local image properties.

$$T_{xy} = a\sigma_{xy} + bm_{xy}$$

Where a and b are denotes nonnegative constants, and

Take note that T is threshold array also the same size as the image in which it was achieved. The threshold value at a particular location in the array (x, y) is used to segment the value of an image at that point.

$$T_{xy} = a\sigma_{xy} + bm_G$$

Where m_G denotes the mean of the global image. The segmented image is calculated as follows:

$$g(x,y) = \begin{cases} 1 & if\ f(x,y) > T_{xy} \\ 0 & if\ f(x,y) \le T_{xy} \end{cases}$$

Where f(x, y) denotes the input images. This equation is assessed for each pixel point in the image, and a different threshold is calculated with each (x, y) location utilizing the pixels in the neighborhood$S_{x,y}$.

Significant power could be inserted to variable thresholding (with only a modest improvement in calculation) through using predicates based on various parameters calculated in the neighborhood of a point (x, y):

$$g(x,y) = \begin{cases} 1 & if\ Q(local\ parameters\ )\ is\ TRUE \\ 0 & if\ Q(local\ parameters)\ )\ is\ FALSE \end{cases}$$

Where Q is a predicate depending on parameters calculated from neighbouring pixels. Consider the following predicate, which is derived from the local mean and standard deviation:

$$Q(\sigma_{xy}, m_{xy}) = \begin{cases} TRUE & if\ f(x,y) > a\sigma_{xy}\ AND\ f(x,y) > bm_{xy} \\ FALSE & Otherwise \end{cases}$$

EXAMPLE 4: Variable thresholding based on local image properties.

The yeast image is depicted in Figure 8.41 (a). Given that this image contains three distinct intensity levels, it is logical to assume that the dual thresholding would be an effective segmentation technique. The outcome of applying the dual thresholding method is summarized in Figure 8.41 (b). As illustrated in the figure, while it was necessary to distinguish the bright areas from the background, the mid-gray portions upon on right side of the image were not correctly segregated (i.e., separated). To demonstrate the application of local thresholding, we calculated the local standard deviation for all (x, y) values in the input image using a neighborhood of size the result is depicted in Figure 8.41 (c). Take note of how the faint outer lines accurately delineate the cell boundaries. Following that, we constructed a predicate in the manner, but by using global mean rather than When the background is fairly constant and then all the object intensities are above or below the background intensity, selecting the global mean usually gets better results. The values and have been used to accomplish the specification of the predicate (as is typically the case in applications such as this, these parameters were estimated experimentally). After that, the image was segmented. As illustrated in Fig. 8.41 (d), segmentation was quite successful. Notably, all outer regions have been segmented correctly, and the majority of the inner, brighter regions have been isolated correctly.

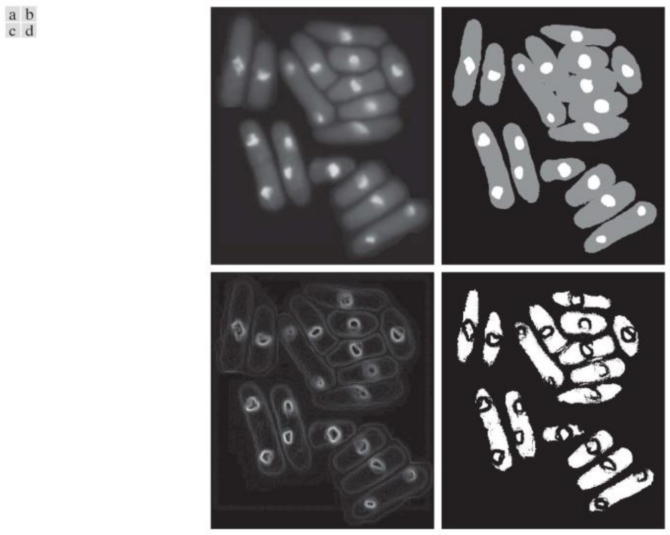


FIGURE 8.41 (a) Image from Fig. 8.40. (b) Image segmented using the dual thresholding approach (c) Image of local standard deviations. (d) Result obtained using local thresholding.

(Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

Variable Thresholding Based on Moving Averages

The moving average technique is a variation on the variable threshold processing technique. The variable threshold is comparative to the processing of global thresholds. The term "global threshold processing" relates to the procedure of determining a fixed threshold based on the entire image. If the size of each pixel in the image is greater than this Otherwise, the value is considered to be in the background. The variable threshold indicates that each pixel or pixel block in the image has a unique threshold. If a pixel exceeds its associated threshold, it is viewed to be in the foreground. The moving average method scans the entire image in a linear zigzag pattern, creates a threshold at each point, and compares the grey value at that point to the threshold to segment the image.

Method

Suppose a 5x5 picture is shown below, aij represents the gray value at position (i, j).

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} \\ a_{21} & a_{22} & a_{23} & a_{24} & a_{25} \\ a_{31} & a_{32} & a_{33} & a_{34} & a_{35} \\ a_{41} & a_{42} & a_{43} & a_{44} & a_{45} \\ a_{51} & a_{52} & a_{53} & a_{54} & a_{55} \end{pmatrix}$$

Due to the fact that it is linearly scanned in the z-shape, the two-dimensional matrix must be converted to a one-dimensional row matrix.

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} & a_{25} & a_{24} & a_{23} & a_{22} & a_{21} & . & . & . & a_{51} & a_{52} & a_{53} & a_{54} & a_{55} \end{pmatrix}$$

The moving average algorithm makes use of two parameters, n and b. The average of n pixels is represented by n, and b is a threshold coefficient. The following one-dimensional matrix could be used as a filter to filter and average the image obtained previously.

$$\begin{pmatrix} 1/n & 1/n & . & . & . & 1/n \end{pmatrix}$$

This allows for the calculation of the average mij at each point. The threshold at this pixel is calculated by multiplying the parameter b by mij.

$$\begin{pmatrix} m_{11} & m_{12} & m_{13} & m_{14} & m_{15} & m_{25} & m_{24} & m_{23} & . & . & . & m_{51} & m_{52} & m_{53} & m_{54} & m_{55} \end{pmatrix}$$

Then you can compare each pixel's grayscale to the threshold to obtain the final segmented image.

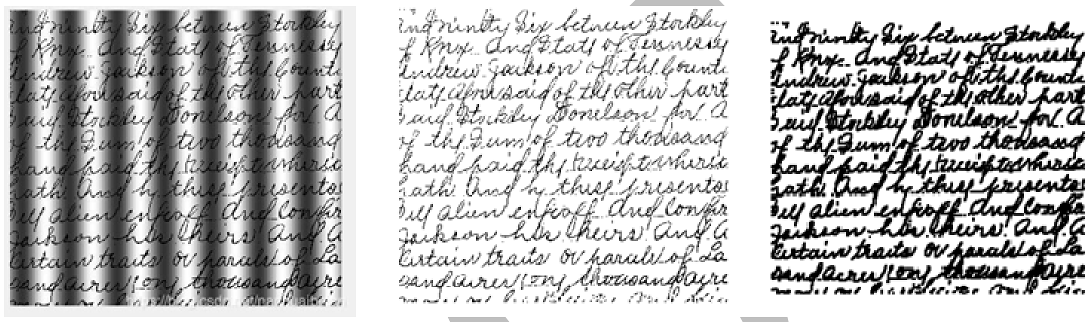


FIGURE 8.42 a) Original image b) Moving average processing c) Finally, perform a minimum filter on the result

(Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

8.3 Segmentation by Region Growing and by Region Splitting and Merging

Region-Based Segmentation:

Segmentation is used to divide an image into regions. We decided to approach this issue by defining regions using discontinuities in grey levels and segmenting them using thresholds based on the distribution of pixel properties such as gray-level values or colour.

Basic Formulation:

Assume that R is used to represent the entire image region. Segmentation can be thought of as a process that divides R into n subregions, R1, R2... Rn, in such a way that

a. $\bigcup_{i-1}^{n} R_i = R$
b. $R_i$is a connected set, for i=0,1,2,3..,n;
c. $R_i \cap R_j = \emptyset \; for\ all\ i\ and\ j, i \neq j$
d. $Q(R_i) = TRUE\ for$ i = 0,1,2,3.., n
e. $Q(R_i \cup R_j) = TRUE$ for any adjacent regions $R_i\ and\ R_j$

P (R_k) signifies a logical predicate described over the points in the set R_i and ∅ whereas is the null set. Condition (a) specifies that the segmentation should be complete; each pixel must be contained within a region. Conditions (b) require that points within a region be connected in a predefined way. As indicated by condition (c), the regions must be disjoint. Condition (d) specifies the properties that all pixels within a segmented region must satisfy—for example, P (Ri) = TRUE if all pixels within Ri have the same grey level. Finally, condition (e) establishes that regions Ri and Rj are distinct in terms of the predicate P.

Region Growing:

As the name implies, region growing is a process that groups pixels or subregions accordance with the given criteria into larger regions. The fundamental approach is to begin with a collection of "seed" points and grow regions from them by attaching to each seed those neighboring pixels that share similar properties (such as specific ranges of grey level or color). When no a priori information is provided, the operation is to calculate the same set of properties for each pixel that will eventually be used to assign pixels to regions during the growing process. If such outcome of the these calculations exposes clusters of values, the pixels with properties that put them close to the clusters' centroid could be used as seeds.

The choice of similarity criteria is determined not only by the nature of the problem at hand, but also from the type of image data available. For instance, analysis of land-use satellite imagery is highly dependent on the use of colour. Without the inherent information contained in colour images, this problem would be significantly more difficult, if not impossible, to solve. When analyzing monochrome images, region analysis must be performed using a set of descriptors based on grey levels and spatial properties (such as moments or texture).

Essentially, region expansion should halt when no additional pixels fulfill the region's inclusion criteria. Gray level, texture, and colour are all local in nature and do not take into account the region's "history." Additional criteria that enhance the power of a region growing algorithm include the concept of size, similarity between a candidate pixel and the pixels grown thus far (for example, a comparison of the candidate's grey level to the average grey level of the grown region), and the shape of the region being grown. The use of these types of descriptors is predicated on the assumption that at least a partial model of expected results is available.

Let f(x, y) represent an input image; S(x, y) represent a seed array that included 1's at seed point locations and 0's elsewhere ; and Q represent a predicate to be implemented at every location (x, y). Assume that arrays f and S are of equal size. The following is a simple region-growing algorithm based on 8-connectivity.

1. Identify all connected components in S(x, y) and reduce them to a single pixel; label all such pixels found as 1. All other pixels in S are labelled with a value of 0.

2. Create an image f_Q such that at each point (x, y), f_Q (x,y)=1 if the input image fulfils a provided predicate, Q, and f_Q (x,y)=0 otherwise.

3. Let g be an image created by adding all the 1-valued points in f_Q that are 8-connected to each seed point in S.

4. Assign a unique region label to each connected component in g. (e.g.,integers or letters). This is the image segmented as a result of region growing.

EXAMPLE 5: Segmentation by region growing

Figure 8.43 (a) illustrates an 8-bit X-ray image of a weld (the horizontal dark region) with multiple cracks and porosities (the bright regions running horizontally via the image's centre). By segmenting the defective weld regions, we prove use of region growing. These regions could have been used for number of purposes, such as weld inspection, archiving historical studies, and controlling an automated welding system.

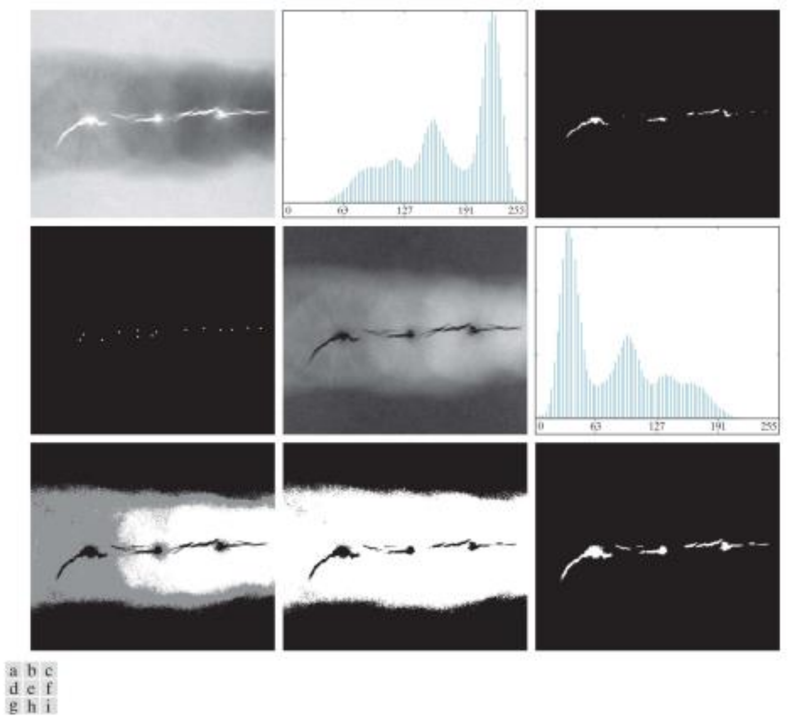


FIGURE 8.43 (a) X-ray image of a defective weld. (b) Histogram. (c) Initial seed image. (d) Final seed image (the points were enlarged for clarity). (e) Absolute value of the difference between the seed value (255) and (a). (f) Histogram of (e). (g) Difference image thresholded using dual thresholds. (h) Difference image thresholded with the smallest of the dual thresholds. (i) Segmentation result obtained by region growing

(Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

The first step is to establish the seed points. Because cracks and porosities attenuate X-rays significantly less than solid welds, we anticipate the regions probably contains such types of defects to be substantially brighter than the remainder of the X-ray image. We can obtain the seed points by thresholding the original image with a high percentile threshold. The histogram of the image is shown in Figure 8.43 (b), and the thresholded outcome is shown in Figure 8.43 (c), with a threshold equal to the image's 99.9 percentile of intensity values, which in this case was 254. . The result of morphologically eroding each connected component in Figure 8.43 (c) to a single point is shown in Figure 8.43 (d).

Following that, we must clearly state a predicate. In this example, we're interested in appending to each seed all pixels that are (a) eight-connected to it and (b) "similar" to it. Our predicate is applied at each location (x, y) using absolute intensity differences as a measure of similarity.

$$Q = \begin{cases} TRUE \;\; if\ the\ absolute\ difference\ of\ intensities\ between\ the\ seed\ and\ the\ pixel\ at\ (x,y)\ is \le T \\ FASLSE \quad Otherwise \end{cases}$$

Where T denotes a predefined threshold. Whereas this predicate is relied on intensity differences as well as contains a particular threshold, we can define more sophisticated methods in which each pixel has a distinctive threshold and other properties are used in addition to differences. As the remainder of this example demonstrates, the preceding predicate is sufficient to resolve the problem.

As stated previously, all seed values are 255 because the image was thresholded at a value of 254. The difference between the seed value (255) and the value in Figure 8.43 (a) is illustrated in Figure 8.43 (e). The image in Figure 8.43 (e) contains all of the distinctions necessary to compute the predicate at each location (x, y). The corresponding histogram is shown in Figure 8.43 (f). We require a threshold for establishing similarity in the predicate. Because the histogram contains three primary modes, we can begin by applying the dual thresholding technique to the difference image. In this case, the resulting two thresholds were and, as we can see, they closely correspond to the histogram's valleys. (As a brief aside, the image was segmented using these two thresholds. The result in Fig. 8.43 (g) demonstrates that despite the fact that the thresholds are in the deep valleys of the histogram, segmenting the defects cannot be accomplished using dual thresholds.)

The difference image is thresholded in Figure 8.43 (h) using only The black points represent pixels for which the predicate was TRUE; the white points represent pixels for which the predicate was FALSE. The critical finding here is that the points in the weld's good regions failed the predicate, and thus will be excluded from the final result. The region growing algorithm will consider the points in the outer region as candidates. Step 3 will, however, reject the outer points because they are not connected to the seeds in an 8-way fashion. Indeed, as illustrated in Fig. 8.43 (i), this step resulted in the correct segmentation, indicating that connectivity was a necessary condition in this case. Finally, note that we used the same value in Step 4 for all regions discovered by the algorithm. In this case, it was visually preferable to do so because all of those regions in this application have the same physical meaning—they all represent porosities.

Region Splitting and Merging

The method described earlier grows regions from seed points. An option available is to at first subdivide an image into a series of non - overlapping regions and then merge and/or split the regions in order to satisfy the segmentation conditions. Following that, the fundamentals of region splitting and merging are mentioned.

Assume that R represents the entire image region and that a predicate Q is chosen. One method for segmenting R is to subdivide it subsequently into smaller and smaller quadrant regions, such that we begin with the entire region, R. If the region is split into quadrants. If Q is FALSE for any quadrant, that quadrant is divided into sub-quadrants, and so forth. This technique is conveniently represented by so-called quadtrees; which is, trees in which each node will have precisely four descendants, as illustrated in Fig. 8.44 (the images equivalent to the nodes of a quadtree are occasionally referred to as quadregions or quadimages). Take note that the root of the tree represents the entire image, while each node represents the subdivision of a node into four descendant nodes. Only was subdivided further in this instance.

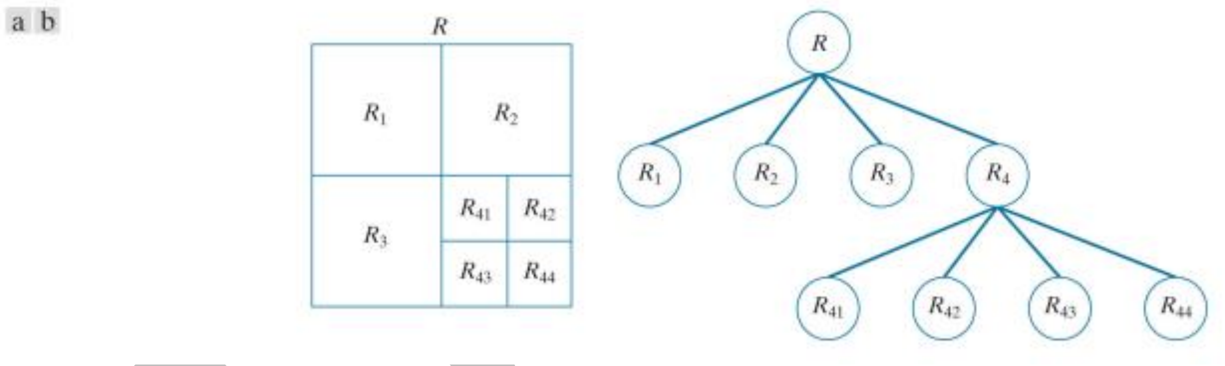


FIGURE 8.44 (a) Partitioned image. (b) Corresponding quadtree. R represents the entire image region.

(Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

When splitting is used exclusively, the final partition typically includes adjacent regions with identical properties. This disadvantage can be overcome by permitting for both merging as well as splitting. Satisfying segmentation constraints. It involves merging adjacent regions with pixels that satisfy the predicate Q. That is, two adjacent regions are merged only if they are adjacent.

The preceding discussion can be summarised as follows: at any step, we

1. Split any region R_i into four disjoint quadrants for which $Q(R_i) = FALSE$;

2. When no further splitting is possible, merge any adjacent regions R_j and R_k for which $Q(R_i \cup R_k) = TRUE$

3. Stop when no further merging is possible.

Numerous variations are possible on this basic theme. For instance, if we allow the merging of any two adjacent regions in Step 2 and each one satisfies the predicate independently, a significant simplification occurs. This results in a significantly simpler (and faster) algorithm, as predicate testing is restricted to individual quadregions. As demonstrated in the following example, this simplification is still capable of producing acceptable segmentation results.

EXAMPLE 6: Segmentation by region splitting and merging

The Cygnus Loop supernova is depicted in Figure 8.45 (a) in an X-ray image. The purpose of this example is to segment (extract) the "ring" of less dense matter that surrounds the dense inner region. The region of importance exhibits several readily apparent characteristics that should assist in segmentation. To begin, we recognize that the data in this region are random, implying that their standard deviation ought to be greater than that of the background (which is close to zero) and the large central region, which is smooth. Likewise, the mean value (average intensity) of a region containing outer ring data must be greater than the mean of the darker background and less than the mean of the lighter central region. Thus, using the following predicate, we should be able to segment the region of interest:

a b
c d

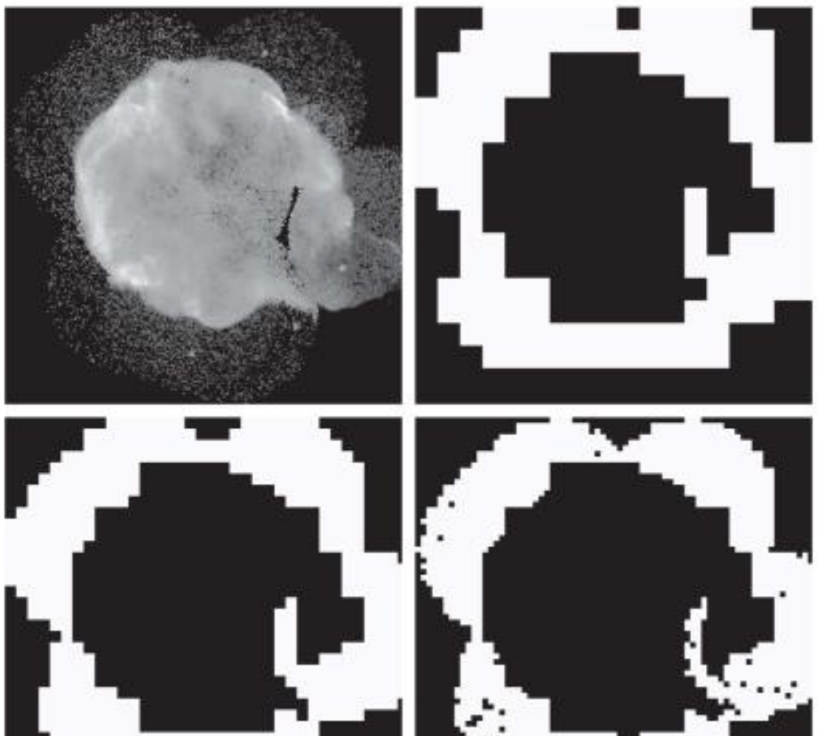


FIGURE 8.45 (a) Image of the Cygnus Loop supernova, taken in the X-ray band by NASA's Hubble Telescope. (b) Through (d) Results of limiting the smallest allowed quadregion to be of sizes of and pixels, respectively. (Original image courtesy of NASA.)

(Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

$$Q = \begin{cases} TRUE & if\ \sigma_R > a\ AND\ 0 < m_R < b \\ FALSE & otherwise \end{cases}$$

where σ_R and m_R denote the standard deviation and mean of the processed region, respectively, and a and b denote nonnegative constants.

Numerous regions within the external area of interest were analyzed, and the mean intensity of pixels in those regions was never greater than 125, and the standard deviation was always greater than 10. The figures 8.45 (b) through (d) display the results acquired by varying the minimum size allowed for quadregions from 32 to 8. White pixels were assigned to quadregions that satisfied the predicate; black pixels were assigned to the remainder of the region. The optimal result in contexts of obtaining the outer region's shape was achieved by using quadregions of size In Fig. 8.45 (d), the small black squares represent quadregions of size whose pixels do not satisfy the predicate. By using smaller quadregions, the number of such black regions would increase. Using larger regions than those shown here results in a more "block-like" segmentation. Not that the segmented region (white pixels) in each case was a connected region that completely separated the inner, smoother region from the background. Thus, segmentation partitioned the image effectively into three distinct areas that correspond to the image's three primary features: background, dense region, and sparse region. Using any of the white regions in Fig. 8.45 as a mask would make extracting these regions from the original image a relatively simple task.

8.4 Region Segmentation using Clustering and Superpixels

We will describe two relevant methods to region segmentation in this section. The first one is a more traditional approach based on clustering data according to variables such as intensity and color. The second approach is extremely quite advanced and is focused primarily on the extraction of "superpixels" from an image via clustering.

Region Segmentation Using K-Means Clustering

Clustering is a technique for categorizing a set of data into a predetermined number of groups. K-means clustering is a widely used technique. In k-means clustering, a collection of data is partitioned into k number groups.

It divides a given set of data into k distinct clusters. The K-means algorithm is divided into two distinct phases. It calculates the k centroid in the first phase and then assigns each point to the cluster with the closest centroid to the respective data point in the second phase. There are several ways to define the distance to the nearest centroid, with the Euclidean distance being one of the most frequently used. Once the grouping is complete, it recalculates the new centroid of each cluster and, using that centroid, calculates a new Euclidean distance between each centre and each data point, assigning the cluster points with the smallest Euclidean distance to the cluster. The partition's clusters are defined by their member objects and centroid. The centroid of each cluster is the point to which the sum of the distances between all of the cluster's objects is minimized. Thus, K-means is an iterative algorithm for minimizing the sum of the distances between each object and its cluster centroid across all clusters.

Consider an image with a resolution of x,y that needs to be clustered into k clusters. Let p(x, y) denote the clustering of an input pixel to be cluter and ck denote the cluster centres. The k-means13 clustering algorithm is as follows:

1. Initialize the cluster k and the centre.
2. Using the relation given below, calculate the Euclidean distance d between the image's centre and each pixel.
$$d = \| p(x,y) - c_k \|$$
3. Based on the distance d, assign all pixels to the nearest centre.
4. Once all pixels have been assigned, recalculate the center's position using the relationship shown below.
$$c_k = \frac{1}{k} \sum_{y \epsilon c_k} \sum_{x \epsilon c_k} p(x,y)$$
5. Repeat the process until the tolerance or error value is satisfied.
6. Resize the cluster pixels to match the dimensions of the image.

While k-means has the substantial advantage of being simple to implement, it does have some disadvantages. The final clustering results' quality is determined by the randomly chosen initial centroid. As a result, if the initial centroid is chosen randomly, the result will be different for different initial centres. Thus, the initial centre will be carefully chosen to achieve the segmentation we desire. Additionally, computational complexity is a factor to consider when designing the K-means clustering algorithm. It is dependent on the number of data elements, clusters, and iterations.

EXAMPLE 7: Using k-means clustering for segmentation

Figure 8.46 (a) depicts an image of size pixels, while Figure 8.46 (b) depicts the segmentation produced by the k-means algorithm, the algorithm extracted all of the image's meaningful regions with high accuracy. Compare the quality of the characters in both images, for instance. It is critical to understand that the entire segmentation process was carried out using clustering of a single variable (intensity). Due to the fact that k-means operates on vector observations in general, its ability to discriminate between regions increases as the number of components in vector z increases.

a b

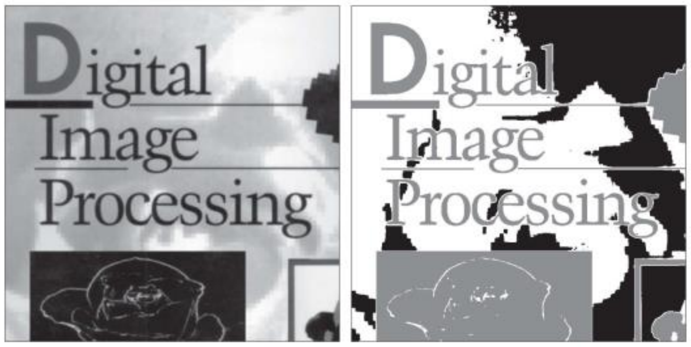


FIGURE 8.46 (a) Image of size pixels. (b) Image segmented using the k-means algorithm with k=3

(Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

Region Segmentation using Superpixels

A superpixel is a collection of pixels that share certain characteristics (like pixel intensity). Superpixels are becoming increasingly useful in a variety of computer vision and image processing algorithms, such as image segmentation, semantic labelling, object detection and tracking, and so on, because they carry more information than pixels.

Because pixels belonging to a given superpixel share similar visual properties, superpixels have a perceptual meaning.

They represent images in a convenient and compact manner, which can be extremely useful for computationally intensive problems.

The concept behind superpixels is to replace the conventional pixel grid with primitive regions that are more perceptually important than specific pixels. The objective is to decrease computational burden and also to strengthen segmentation performance of the algorithm by removing superfluous detail. A simple illustration will assist in explaining the fundamental concept of superpixel representations.

Figure 8.47 (a) depicts a 480,000-pixel-wide image with varying degrees of detail that could be defined verbally as follows: "This is an image of two large carved figures in the foreground and at least three much smaller carved figures resting on a fence behind the large figures." The figures are on a beach, against a background of ocean and sky." Figure 8.47 (b) depicts the identical image defined by 4,000 superpixels and their boundaries, while Figure 8.47 (c) depicts the superpixel image. Somebody could make the argument that the superpixel image's level of particulars outcomes in the same description as the initial, however the the latter contains only 4,000 primitive units, compared to the original's 480,000. Whether the superpixel representation is "adequate" is application-dependent. Yes, if the objective is to define the image at the level of detail mentioned previously. On the other hand, if the primary objective is to identify defects at pixel-level resolutions, the answer is obviously no. Consequently, there are applications, such as computerised medical diagnosis, where any form of approximate representation is unacceptable. Nonetheless, there are numerous application areas, such as image database queries, autonomous navigation, and certain branches of robotics, where the cost savings and potential performance improvements far outweigh any discernible loss of image detail.

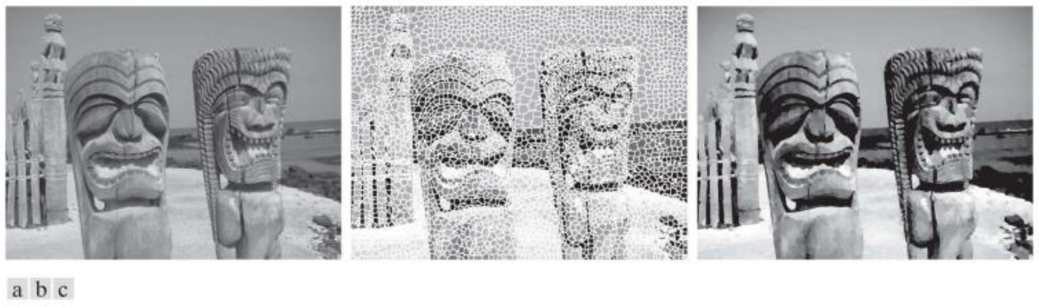


FIGURE 8.47(a) Image of size (480,000) pixels. (b) Image composed of 4,000 superpixels (the boundaries between superpixels (in white) are superimposed on the superpixel image for reference—the boundaries are not part of the data). (c) Superpixel image. (Original image courtesy of the U.S. National Park Services.)

(Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

Boundaries are one of the most important aspects of any superpixel presentation. That is, superpixel images must preserve boundaries between regions of interest. As shown in Fig. 8.47 (c), this is the case. Note, for instance, how distinguishable the figures are from the background. The same is accurate of the ocean-to-sky boundaries and between the beach and the ocean. Other features include topological property preservation and, of course, computational efficiency. This section's superpixel algorithm meets these criteria. Reduce the difference between a superpixel image and its parent image to save storage and computation time. A similar image is shown in Fig. 8.48 (a), while Fig. 8.48 (b) is a superpixel image made up of 40,000 superpixels. The only visual difference between these images is a slight difference in contrast caused by averaging intensities in each superpixel region (we will discuss this in more detail later in this section). FIG. 8.48 (c) shows an overall difference in contrast, as well as minor differences around sharp edges. Remember that the superpixel image has fewer elements than the original and that contrast differences can be easily corrected using histogram processing.

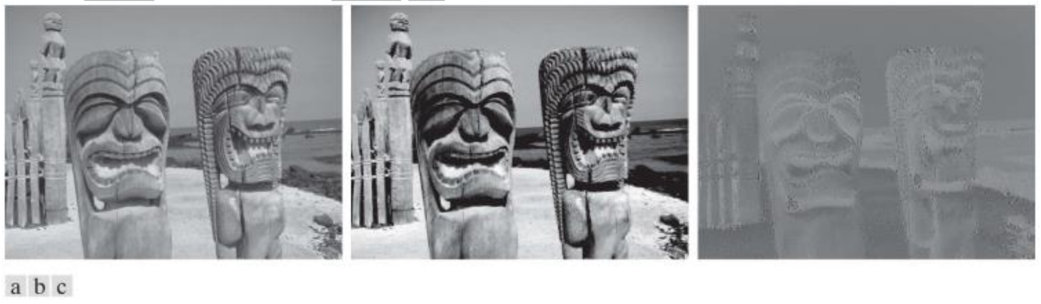


FIGURE 8.48 (a) Original image. (b) Image composed of 40,000 superpixels. (c) Difference between (a) and (b).

(Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

Finally, we show the results of reducing superpixels to 1,000, 500, and 250. Compared to Fig. 8.47 (a), Fig. 8.49 shows a significant loss of detail, but the first two images contain most of the detail relevant to the image description. Two of the three small carvings on the back fence were removed. So did the 250-element superpixel image. However, the principal region boundaries and basic topology of the images were preserved.

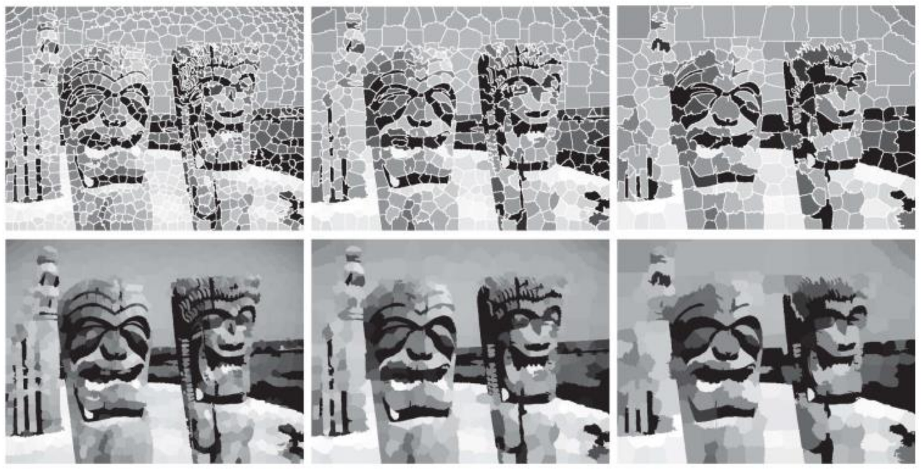


FIGURE 8.49 Top row: Results of using 1,000, 500, and 250 superpixels in the representation of Fig. 8.47 (a) . As before, the boundaries between superpixels are superimposed on the images for reference. Bottom row: Superpixel images

(Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

SLIC Superpixel Algorithm

Simple Linear Iterative Clustering is a state-of-the-art algorithm for segmenting superpixels that is computationally efficient. In a nutshell, the algorithm clusters pixels in the consolidated five-dimensional colour and image plane space to generate compact, nearly uniform superpixels efficiently.

Actually, the approach is quite straightforward. SLIC performs a local clustering of pixels in a five-dimensional space defined by the CIELAB colorspace's L, a, and b values and the pixels' x, y coordinates. It uses a different distance measurement, which results in more compact and regular superpixel shapes, and it can be used on both grayscale and colour images.

SLIC creates superpixels by clustering pixels according to their colour similarity and image plane proximity. Clustering is performed using a five-dimensional [labxy] space. For small colour distances, the CIELAB colour space is considered to be eternally uniform. It is not recommended to use Euclidean distance alone in 5D space, and thus the authors introduced a new distance measure that takes superpixel size into account.

Distance Measure

SLIC accepts an input of a desired number of approximately equal-sized superpixels K. As a result, each superpixel will consist of approximately N/K pixels. Thus, for superpixels of equal size, there will be a superpixel centre at each grid interval $S = \sqrt{(N/K)}$

K cluster centres for superpixels $C_k = [l_k, a_k, b_k, x_k, y_k]$ where k = [1, K] are chosen at regular grid intervals S. Given that any cluster has a spatial extent of approximately S2, it can be believed that pixels related with this cluster are located within a 2S x 2S area in the xy plane around the superpixel centre.

For small distances in CIELAB colorspace, Euclidean distances are meaningful. When spatial pixel distances exceed this perceptual colour distance threshold, spatial pixel distances begin to outweigh pixel colour similarity.

Distance measure Ds is defined as follows.

$d_{lab} = \sqrt{((l_k - l_i)^2 + (a_k - a_i)^2 + (b_k - b_i)^2)}$

$d_{xy} = \sqrt{((x_k - x_i)^2 + (y_k - y_i)^2)}$

$D_s = d_{lab} + (m / S)* d_{xy}$

D_s is the sum of the lab and xy plane distances normalised by the grid interval S. In D_s, we introduce the variable m, which allows us to adjust the superpixel's compactness. The larger the value of m, the more emphasis is placed on spatial proximity and the cluster becomes more compact. This can be any value between [1 and 20]. The algorithm's authors chose m=10 as the starting point.

Algorithm

It starts by randomly sampling K frequently spaced cluster centres and relocating them to seed locations corresponding to the neighborhood's lowest gradient position. This avoids placing them on an edge and minimises the possibility of selecting a noisy pixel. The gradients of an image are calculated as

$G(x, y) = \| I(x + 1, y) - I(x - 1, y) \|^2 + \| I(x, y + 1) - I(x, y - 1) \|^2$

where I(x, y) denotes the lab vector associated with the pixel at position (x, y), and ||.|| denotes the L2 norm. This takes both colour and intensity information into account.

Each pixel in the image corresponds to the cluster centre closest to it whose search area overlaps this pixel. After associating all pixels with the nearest cluster centre, a new centre is computed as the average labxy vector of all cluster pixels.

At the conclusion of this process, a few stray labels may remain, i.e., pixels adjacent to a larger segment with the same label but not connected to it. In the final step of the algorithm, it enforces connectivity by relabeling disjoint segments with the labels of the largest neighbouring cluster.

SLIC Algorithm can be summarized as-

Algorithm 1 Efficient superpixel segmentation

```
1. Initialize cluster centers $C_k = [l_k, a_k, b_k, x_k, y_k]^T$ by sampling pixels at regular grid steps S.
2. Perturb cluster centers in an n x n neighborhood* to the lowest gradient position.
3.  repeat
4. For each cluster center $C_k$ do
5. Assign the best matching pixels from a 2S x 2S square neighborhood around the cluster
   center according to the distance measure.
6. end for
7. Compute new cluster centers and residual error E { L1 distance between previous centers
   and recomputed centers}
8. until E < threshold
9. Enforce connectivity.
```

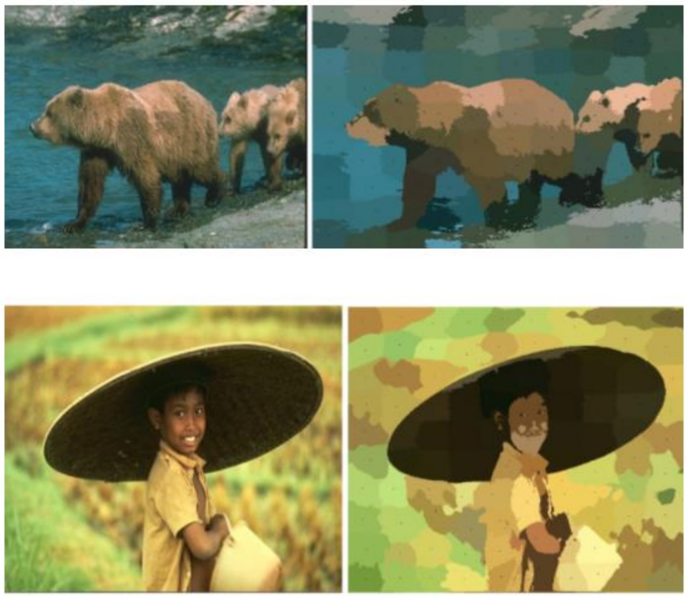


FIGURE 8.50 results of implementation (with K = 100 and m = 20)

8.5 Region Segmentation Using Graph Cuts

This section discusses a technique for dividing an image into regions that involves demonstrating the image's pixels as nodes in a graph and then determining the best partition (cut) of the graph into groups of nodes. Optimality is defined by criteria that have a high value for members of a group (i.e., a region) and a low value for members of other groups. As demonstrated later in this section, graph-cut segmentation is capable of generating results that are sometimes superior to those obtained using any of the previously studied segmentation methods. The cost of this potential benefit is increased implementation complexity, which generally results in slower execution.

Images as Graphs

A graph, G, is a mathematical structure comprised of a set V of nodes and an associated set E of edges connecting those vertices:

Nodes and edges are referred to as vertices and links.

$$G= (V, E)$$

Where V is a set and Cartesian product $V \times V$

$$E \subseteq V \times V$$

is a collection of elements from V that are ordered pairs. If $(u, v) \in E$ Indicates that $(v, u) \in E$ the graph is stated to be undirected; otherwise, the graph is stated to be directed. For instance, we can think of a street map as a graph, with nodes representing street intersections and edges representing the streets that connect those intersections. The graph is undirected if all streets are bidirectional (meaning that we can travel both ways from any two intersections). Otherwise, the graph is directed if at least one street is one-way.

We are focused in undirected graphs whose edges are even farther characterized by a matrix, W, for whom the element w (i, j) represents the weight associated with the edge connecting nodes I and j. Due to the fact that the graph is undirected, w (i, j) = w (j, i) indicating that W is a symmetric matrix. The weights are chosen so that they are proportional to one or more measures of similarity between all pairs of nodes. A weighted graph is a graph whose edges are associated with weights.

The purpose of this section is to represent an image to be segmented as a weighted, undirected graph, with nodes representing the image's pixels and edges connecting each pair of nodes. Each edge's weight, w (i, j), is proportional to the similarity between nodes i and j. We then actively sought to partition the graph's nodes into disjoint subsets V_1, $V_{2\ldots}$ V_K, where the resemblance between nodes within a subset is high and the resemblance between nodes in different subsets is low. The partitioned subsets' nodes correspond to the segmented image's regions.

Additionally, superpixels are well-suited for use as graph nodes. Thus, when we pertain to "pixels" in an image in this section, we are implicitly referring to superpixels.

By cutting the graph, set V is partitioned into subsets. A graph cut is a division of V into two subsets A and B in such a way that

$$A \cup B = V \text{ and } A \cap B = \emptyset$$

Where the cut is accomplished by omitting the edges that connect subgraphs A and B. There are two critical aspects to incorporating graph cuts for image segmentation: (1) associating the graph with the image; and (2) cutting the graph in such a way that makes sense in terms of partitioning the image into background and foreground (object) pixels.

The simplified approach depicted in Figure 8.51 demonstrates how to generate a graph from an image. The nodes of the graph correspond to the pixels in the image, and to simplify the explanation, we permit edges only between adjacent pixels via 4-connectivity, which implies that neither diagonal edges link the pixels. However, take note that edges are stipulated in between each pair of pixels in overall. Weights for edges are usually evaluated using spatial relationships (for instance, distance from the vertex pixel) and intensity measures (for example, texture and colour), which are consistent with pixel similarity. The degree of similarity among two pixels is described in this simple example as the opposite of the differentiation in their intensities. – i.e., for two nodes (pixels) n_i and n_j, the weight of the edge between them is $w(i,j) = 1/(|I(n_i) - I(n_j)| + c)$, where $I(n_i)$ and $I(n_j)$ are the intensities of the two nodes (pixels), and c is a constant

That can be used to prevent division by zero. Thus, the closer the intensity values between adjacent pixels are, the greater the value of w.

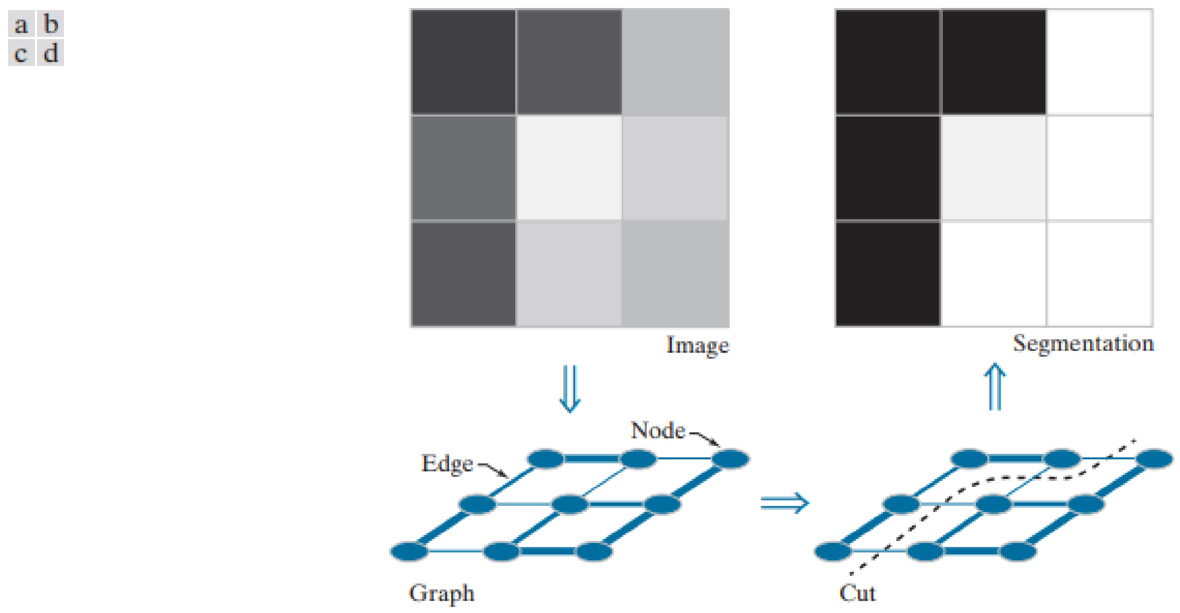


FIGURE 8.51(a) A 3 × 3 image. (c) A corresponding graph. (d) Graph cut. (c) Segmented image.

(Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

While the general form in Fig. 8.51 is what we want to concentrate on in this section, we also provide another, more common approach for developing image graphs just for completeness. The same graph as shown in Figure 8.52 is shown here, however, two additional nodes, the source and sink terminal nodes, are presented in addition to the previous, called here the "source" and "sink" terminal nodes, respectively, all of which are connected via unidirectional links called t-links.

The terminal nodes do not play a part to the image; rather, they simply serve to connect each pixel to one of two possible background or foreground states (an object or not).

Possible outcomes are the t-links' weights. Thickness of each t-link is proportional to the probability that the graph node it is connected to is a foreground or background pixel in Figures 8.52 (c) and (d) (the thicknesses shown are so that the segmentation result would be the same as in Fig. 8.51). It is up to the designer to decide which of the two nodes we call "background" or "foreground".

MINIMUM GRAPH CUTS

After expressing an image as a graph, the graph is cut into two or more subgraphs. Each resulting subgraph's nodes (pixels) correlate to a region in the segmented image. According to Fig. 8.52, methods depend on analysing the graph as a flow network (of pipes, for instance) and determining what is generally referred to as a minimum graph cut. This methodology is established on the Max-Flow, Minimum-Cut Theorem. This theorem states that the maximum volume of flow moving from source to sink equals the minimum cut in a flow network. This minimum cut is described as the smallest total weight of the edges that, if eliminated, will indeed cause the sink to become disconnected from the source:

$$cut(A,B) = \sum_{u \in A, V \in B} w(u,v)$$

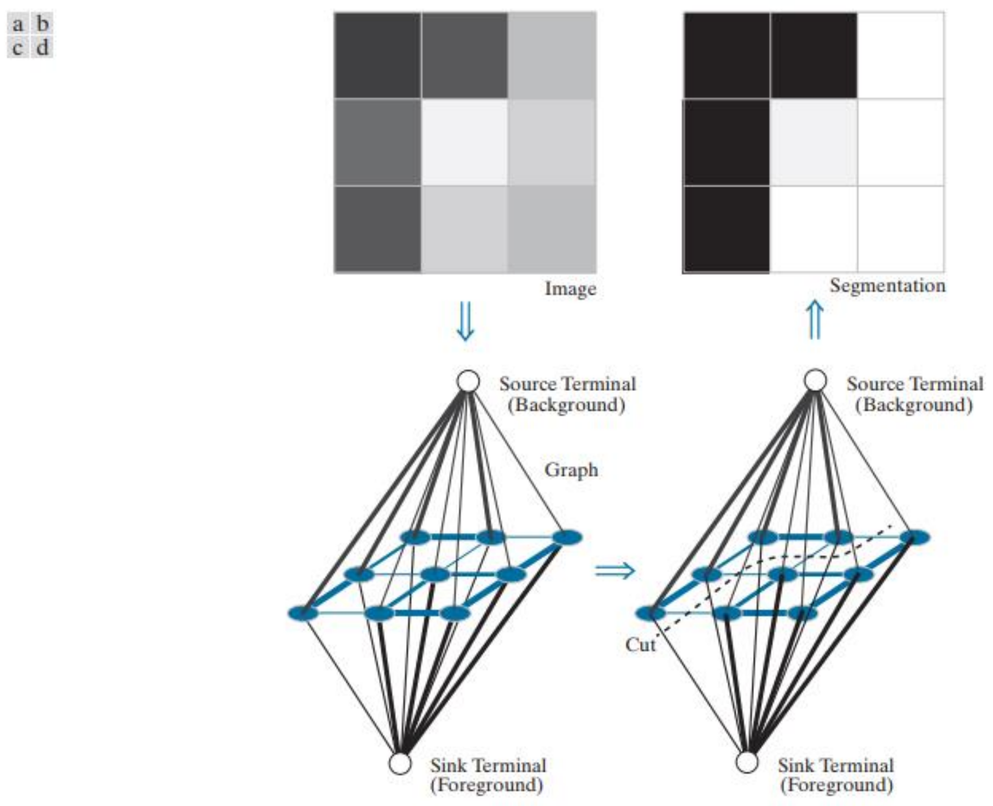


FIGURE 8.52 (a) Same image as in Fig. 8.51 (a). (c) Corresponding graph and terminal nodes. (d) Graph cut. (b) Se nted image.

(Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

While the min-cut method is elegant, it can contribute in groupings that favor cutting small groups of isolated nodes in a graph, resulting in incorrect segmentation. In Figure 8.53, the two regions of interest are denoted by the tightness of the pixel groupings. Meaningful edge weights will be inversely proportional to the distance between two points. However, this would result in smaller weights for isolated points, resulting in min cuts such as the one shown in Fig. 8.53.

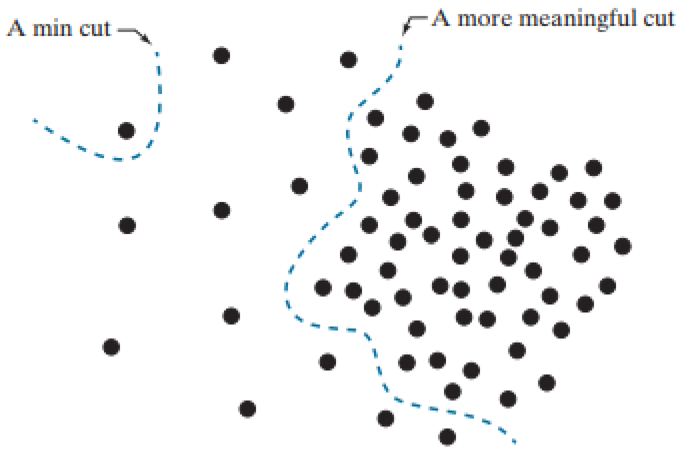


FIGURE 8.53 An example showing how a min cut can lead to a meaningless segmentation. In this example, the similarity between pixels is defined as their spatial proximity, which results in two distinct regions

(Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

The normalized cut algorithm calculates the cost of the cut as a fraction of the total number of edge connections to all nodes in the graph.

$$Ncut(A,B) = \frac{cut(A,B)}{assoc(A,V)} + \frac{cut(A,B)}{assoc(B,V)}$$

where $cut(A,B)$ is given by

$$assoc(A,V) = \sum_{u\in A, z\in V} w(u,z)$$

is the weighted sum of all edges connecting nodes in subgraph A to nodes in the entire graph. Similarly,

$$assoc(B,V) = \sum_{u\in B, z\in V} w(v,z)$$

The total weight of all the edges from all the edges in B, and connecting all the vertices. As you can see, the $assoc(A,V)\, is$ the cut of 'A' from the rest of the graph, and in the same way, $assoc(B,V) is$ the cut of 'B' from the rest of the graph.

We can define a metric for total normalized association within graph partitions if we consider concepts with the same basic building blocks.

$$Nassoc(A,B) = \frac{assoc(A,A)}{assoc(A,V)} + \frac{assoc(B,B)}{assoc(B,V)}$$

Where $assoc(A,A)$ and $assoc(B,B)$are the overall weights linking the nodes within A and B. It is trivial to demonstrate that

$$Ncut(A,B) = 2 - \ Nassoc(A,B)$$

Which indicates that minimizing $Ncut(A,B)$ maximizes $Nassoc(A,B)$ simultaneously.

8.6 Segmentation Using Morphological Watersheds

Previously, we discussed segmentation using three primary concepts: edge detection, thresholding, and region extraction. All of these methods was found to have some advantages (for example, global thresholding is fast) and some disadvantages (for example, the need for post processing, such as edge linking, in edge based segmentation). We will discuss a method based on the concept of so-called morphological watersheds in this section. Segmentation by watersheds incorporates several of the concepts from the other three methods and thus frequently results in more stable segmentation results, including connected segmentation boundaries. Additionally, this approach establishes a straightforward framework for incorporating knowledge-based constraints into the segmentation process.

Watersheds and catchment basins are well-known concepts in topography. Individual catchment basins are divided by watershed lines. Watershed segmentation is a widely used segmentation technique that originated in the field of mathematical morphology. Watershed segmentation is a powerful and rapid technique that can be used for contour detection as well as region-based segmentation. Watershed segmentation is reliant on ridges for proper segmentation, a property that is frequently satisfied in contour detection, where the objects' boundaries are expressed as ridges. By calculating an image's edge map, it is possible to convert the edges of objects into ridges for region-based segmentation. Watershed management is typically accomplished through region-based growth guided by a set of markers in order to avoid severe over-segmentation.

In topography, the Watersheds are well-known. It was initially proposed as a technique for image segmentation. It is a method of image segmentation that is morphologically based. For the watershed transformation, the gradient magnitude of an image is treated as a topographic surface. Watershed lines can be located in a variety of ways. The complete division of an image via watershed transformation is largely dependent on an accurate estimation of the image gradients. The watershed transform result is degraded by the background noise, resulting in the over segmentation. Additionally, under segmentation is caused when low-contrast edges generate small magnitude gradients, resulting in the erroneous merging of distinct regions. Watershed lines can be located in a variety of ways. Different approaches to segmentation based on the watershed principle are possible.

The image data can be interpreted as a topographic surface with altitudes represented by the gradient image grey levels.

Region boundaries correspond to high watersheds, while the interiors of lowgradient regions correspond to catchment basins.

The catchment basins of the topographic surface are homogeneous in the sense that all pixels belonging to the same catchment basin are connected to the basin's region of minimum altitude (gray-level) via a simple path of pixels with monotonically decreasing altitude (gray-level) along the path. These catchment basins then represent the segments of the segmented image.

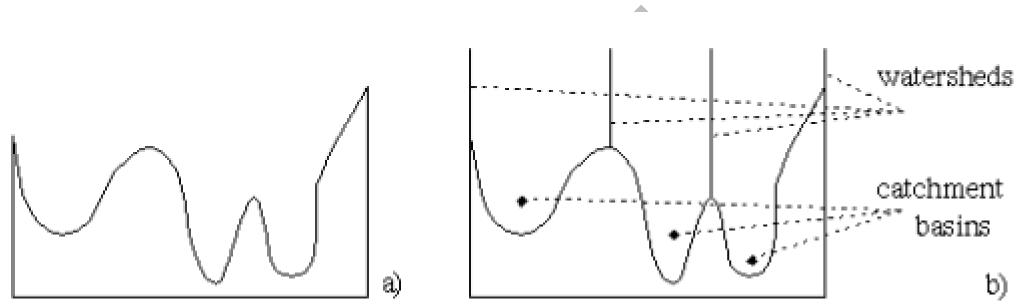


FIGURE 8.54 Gray Level profile of image data b) watershed segmentation –Local minima of gray level yield catchment basins, local maxima define the watershed lines.

Two fundamental approaches to segmenting watershed images. The first one begins by determining a downstream path from each image pixel to a local minimum of image surface altitude. A catchment basin is described as the sum of pixels whose downstream paths all terminate at the same altitude minimum.

Whereas the downstream paths for continuous altitude surfaces can be easily determined by measuring the local gradients, there are no rules that define the downstream paths uniquely for digital surfaces.

The second approach is nearly identical to the first; rather than identifying downstream paths, catchment basins are filled from the bottom up. Assume that each local minimum has a hole in it and that the topographic surface is submerged in water; water begins to fill all catchment basins, the minima of which are below the water level. If two catchment basins merge as a result of further immersion, a dam is built to the highest surface altitude and serves as a watershed boundary.

An efficient algorithm begins by sorting the pixels in ascending order of their grey values, followed by a flooding step that involves a fast breadth-first scan of all pixels in ascending order of their grey levels.

A brightness histogram is computed during the sorting step. Simultaneously, a list of pointers to gray-level h pixels is created and associated with each histogram gray-level, allowing direct access to all gray-level pixels. The flooding step makes extensive use of information about the image's pixel sorting.

Assume that the flooding has reached a level (graylevel, altitude) k. Then, for each pixel with a grey value less than or equal to k, a unique catchment basin label has already been assigned.

Following that, pixels with a grey level of k+1 must be processed. If at least one of its neighbours already has this label, a pixel with gray-level k+1 may belong to a labelled catchment basin.

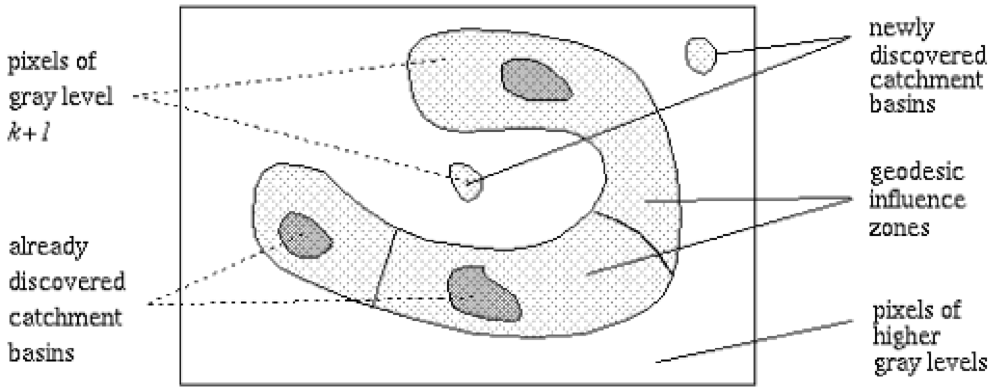

pixels of gray level $k+1$
newly discovered catchment basins
geodesic influence zones
already discovered catchment basins
pixels of higher gray levels

FIGURE 8.55 A geodesic influence zone of a catchment basin

All pixels with a grey level of k+1 that are within a catchment basin's influence zone are labelled l, indicating that the catchment basin is growing. The queued pixels are processed sequentially, and any pixels that cannot be assigned to an existing label represent newly discovered catchment basins and are assigned new and unique labels.

Watershed -flooding analogy

Assume that the grayscale image is a landscape. Allow water to rise from the bottom of each valley, i.e. give each valley's water its own label. Construct a dam or a watershed as soon as the water from two valleys converges. These watersheds will define the boundaries between the image's various regions. The watershed effect can be applied directly to an image, to an edge-enhanced image, or to a distance-transformed image.

Watershed -drop of water analogy

This grayscale image is presented as a landscape. A drop of water that lands anywhere in the landscape will flow down to a local minimum. For each local minimum in the landscape, there is a collection of points referred to as the catchment basin from which a drop of water will flow to that particular minimum. The watershed is defined by the boundaries between adjacent catchment basins. Following are some examples of watersheds directly applied to grey level images.

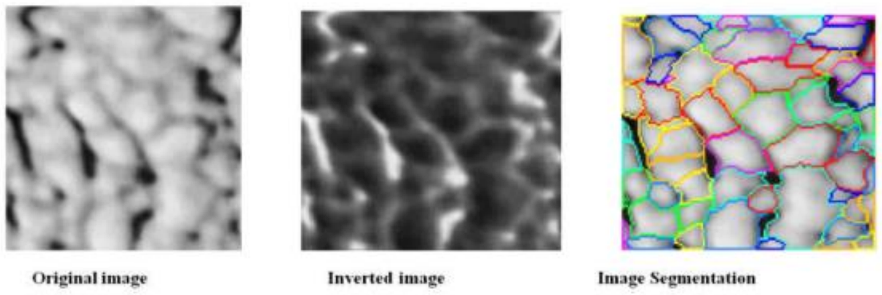


FIGURE 8.56 watershed directly applied on gray level image

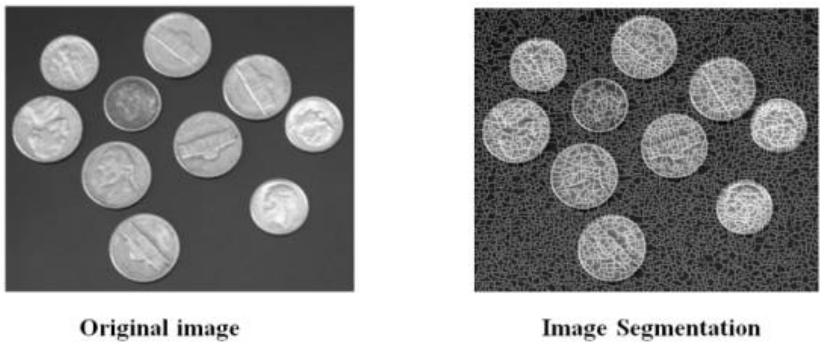


FIGURE 8.57 watershed directly applied on gray level image

(Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

Example for seeded watershed

Each cell contains a nucleus, which can be distinguished using threshold and watershed segmentation. Using these nuclei as seeds, it is simple to locate the cytoplasm.

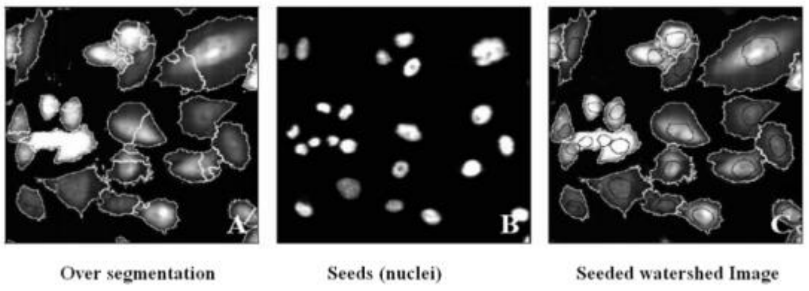


FIGURE 8.58 seeded watershed

(Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

Marker Based Watershed Segmentation

There will be two markers. The first is an internal marker, while the second is an external marker. Internal markers are being used to restrict the number of regions by specifying the objects of interest. For example, seeds can be assigned manually or automatically to regions. Merging regions devoid of markers is permitted (no dam is to be built) External markers are pixels that we are certain are in the background. Watershed lines are common external markers that denote the boundaries of a region. Internal markers can be used to obtain watershed lines for the gradient of the segmented image. As external markers, use the obtained watershed lines.

Each externally defined region contains a single internal marker and a portion of the background. The issue is simplified by dividing each region into two sections: an object (which contains internal markers) and a single background (containing external markers). The following figure illustrates the segmentation of a watershed using markers.

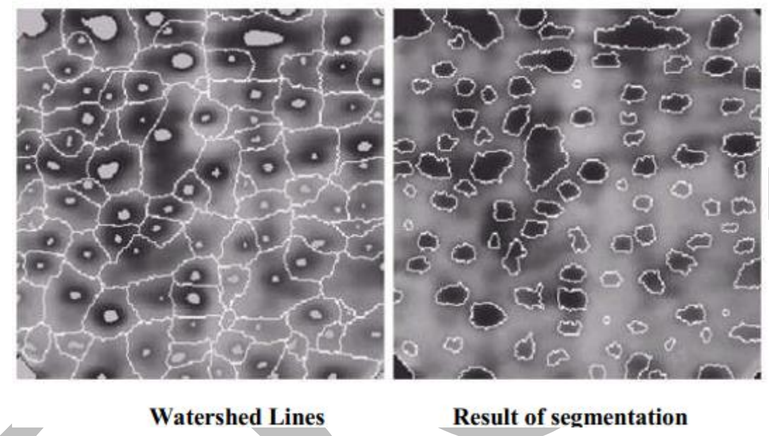


FIGURE 8.59 Marker Based Watershed Segmentation

(Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

8.7 Use of Motion in Segmentation

Motion is a powerful tool that is used in a variety of applications, including robotics, autonomous motion navigation, and dynamic scene analysis, to retrieve objects of interest from a background of undesired detail. The motion is caused by the sensing system's relative displacement from the scene being used. There are two distinct methods for applying motion to segmentation, and they are as follows.

1. Spatial domain approach
2. Frequency domain approach.

In the following section, we will discuss spatial domain techniques.

Spatial domain techniques

Consider the following: There are a number of image frames available, each taken at a different point in time. Let f(x, y, t1) and f(x, y, t2) denote the two images taken at times t1 and t2. Additionally, assume that the image contains a large number of stationary objects and only one moving object. It is possible to detect the boundary of a moving object in such circumstances. The procedure for determining the boundary of a moving object is to find the difference image between images taken at various points in time. The image taken at the 0th interval could be used as a reference image, and subsequent images could be used to subtract from it, resulting in an image that contains only the boundary of the moving object and cancels out all stationary objects.

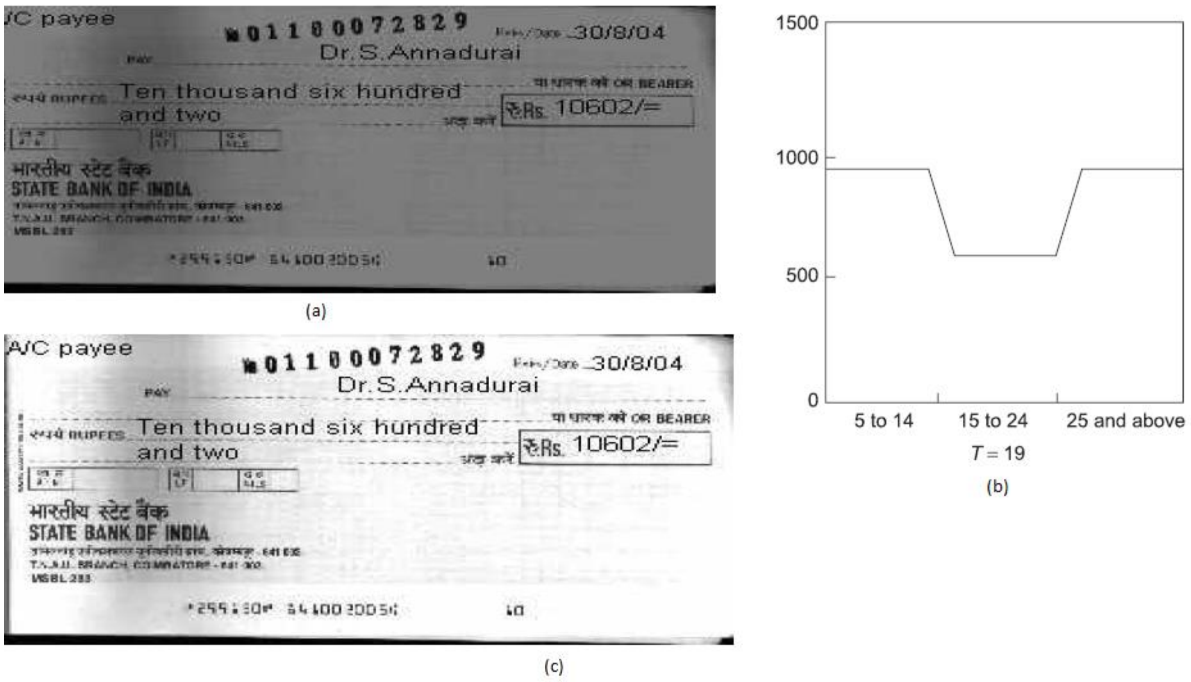


FIGURE 8.60 (a) Image of a cheque (b) Histogram of the image (c) Segmented image

The difference image between two images taken at time t1 and t2 may be defined as

$$d_{i,j}(x,y) = \begin{cases} 1 & if\left|f(x,y,t_i) - f(x,y,t_j)\right| > T \; where\ T\ is\ a\ threshold \\ 0 & otherwise \end{cases}$$

where T denotes a nonnegative threshold. Notably, $d_{i,j}(x,y)$ has a value of 1 at spatial coordinates (x,y)only if the difference in intensity among the two images is meaningfully different at all of those coordinates, as defined by T. Consequently, note that the coordinates (x,y) cover the dimensions of the two images, asserting that the difference image is the same size as the sequence's images.

All pixels with value 1 in $d_{i,j}(x,y)$ are considered to be the result of object motion. This technique is applicable only if the two images are spatially registered and the illumination is relatively constant within the bounds defined by T. In practice, noise can also result in 1-valued entries in$d_{i,j}(x,y)$. Typically, these entries are isolated points in the difference image, and removing them is as simple as creating four or eight connected regions of 1's in image $d_{i,j}(x,y)$ then ignoring any region with less than a predefined number of elements. While this approach may result in the omission of small and/or slow-moving objects, it increases the likelihood that the remaining entries in the difference image are due to motion and not noise.

Accumulative Differences

Consider a series of image frames taken at times t1, t2, t3,..., tn and denoted by the variables f(x, y, t1), f(x, y, t2),..., f(x, y, tn) (x, y, tn). Assume that the reference image is the first image frame f(x, y, t1). By comparing the reference image in the sequence, an accumulative difference image is obtained. Each pixel location in the accumulative image has its count incremented whenever a difference between the reference and the image in the sequence occurs at that pixel location. Thus, when the kth frame is compared to the reference, the entry in the given pixel of the accumulative image indicates the number of times the grey level at that position differed from the reference image's corresponding pixel value. The equation is used to calculate the differences. Figure 6.20 illustrates these concepts**.**

Suppose that the intensity values of moving objects are greater than those of the background, three types of ADIs are considered. To simplify the notation, let $R(x, y)$denote the reference image and k denote t$_k$, such that $f(x, y, k) = f(x, y, t_k)$We begin by assuming that $R(x, y)$=f(x,y,1) Then, for any k > 1, and remembering that the ADI values are counts, we define the following accumulative differences for all relevant values of (x, y):

$$A_k(x,y) = \begin{cases} A_{k-1}(x,y) + 1 & if|R(x,y,) - f(x,y,k)| > T \\ A_{k-1}(x,y) & otherwise \end{cases}$$

$$P_k(x,y) = \begin{cases} P_{k-1}(x,y) + 1 & if|R(x,y,) - f(x,y,k)| > T \\ P_{k-1}(x,y) & otherwise \end{cases}$$

$$N_k(x,y) = \begin{cases} N_{k-1}(x,y) + 1 & if|R(x,y,) - f(x,y,k)| < -T \\ N_{k-1}(x,y) & otherwise \end{cases}$$

where $A_k(x, y)$, $P_k(x, y)$, and $N_k(x, y)$denote the absolute, positive, and negative ADIs computed with the kth image in the sequence, respectively. Each of the three ADIs begins with zero counts and is the same size as the sequence's images. If the intensity values of the background pixels are greater than the values of the moving objects, the order of the inequalities and signs of the thresholds in above equation are reversed.

EXAMPLE: Computation of the absolute, positive, and negative accumulative difference images.

The three ADIs are expressed as intensity images in Figure 8.61 for a rectangular object with a dimension of 75 X 50 pixels which is moving southeasterly at a speed of 5 2 pixels per frame. The images are 256 X 256 pixels in size. We consider the following into account: (1) The positive ADI's nonzero area equals the width of the moving object; (2) the positive ADI's location correlates to the location of the moving object in the frame of reference; (3) the positive ADI's count number stops growing when the moving object is completely displaced from the same object in the reference frame; (4) The absolute ADI encompasses the positive and negative ADI regions; and (5) The direction and speed of a moving object can be evaluated by analyzing the entries in the absolute and negative ADIs.

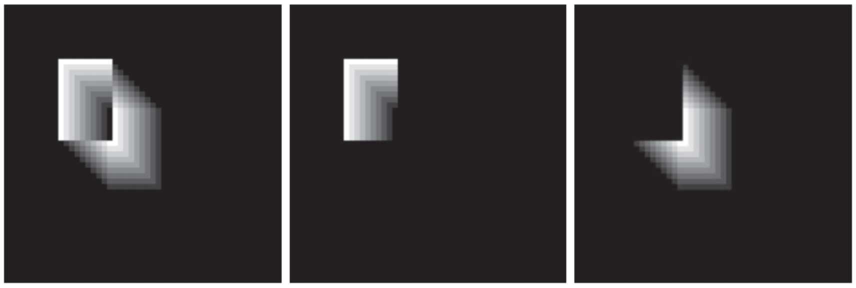


FIGURE 8.61 ADIs of a rectangular object moving in a southeasterly direction. (a) Absolute ADI. (b) Positive ADI. (c) Negative ADI.

(Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

8.8 Unit End questions

1. What are the derivative operators useful in image segmentation? Explain their role in segmentation
2. What is thresholding? Explain about global thresholding
3. Explain about basic adaptive thresholding process used Understand in image segmentation
4. Explain in detail the threshold selection based on boundary characteristics
5. Explain about region based segmentation.
6. What are the derivative operators useful in image segmentation? Explain their role in segmentation.
7. Explain about the Global processing via the Hough Transform for edge linking
8. Explain about the Global processing via graph-theoretic techniques for edge linking
9. Explain about Region Splitting and Merging with an example.
10. Write about edge detection
11. Explain about the Local processing for edge linking
12. Write short note on Region Growing
13. Write the mask for prewitt operator
14. Write the mask for sobel operator
15. Write the mask for laplacian operator

8.9 Reference for further reading

1. Digital Image Processing Gonzalez and Woods Pearson/Prentice Hall Fourth 2018
2. Fundamentals of Digital Image Processing A K. Jain PHI
3. The Image Processing Handbook J. C. Russ CRC Fifth 2010

Chapter 5 Wavelet and Other Image Transforms

5.0 Introduction

In image compression, the wavelet transformation plays an extremely important role. Wavelet transformation is a more appropriate technique than Fourier for image compression applications. Fourier transform is not practical for spectral computing information as all the previous and future information on the signal are essential throughout the entire of the time domain and frequencies cannot be observed varying in time, as the resulting function is time-independent. The Fourier discrete transform is member of an essential class of linear transformations, including Hartley, sine, cosine, Walsh-Hadamard, Slant, Haar and the wavelet transform. Such transformations, decompose tasks into weighted amounts of orthogonal or biorthogonal functions that can be studied with the use of linear algebra tools and functional analysis tools.

5.1 Preliminaries

In Linear Algebra, A vector space (called as linear space) is a group of objects that can be added and multiplied ('scaled') by numbers called scalars. Scalars are sometimes taken by real numbers, but there are vector space with scalar multiplication by complex numbers, rational numbers or usually any field also exists. Vector addition and scalar multiplication operations must fulfil some requirements, known as vector axioms. An inner product is a widespread of the dot product. In a vector space, it is a method of multiplying vectors together, the result being a scalar one.

An inner product <·,·> fulfils four characteristics for a real vector space. Let u, v and w be scalar vectors, and α be a scalar, then:

1. <u+v,w>=<u,w>+<v,w>.

2. <alphav,w>=α <v,w>.

3. <v,w>=<w,v>.

4. <v,v>>=0 and equal if and only if v=0.

It's known as the positive-definite fourth condition in the above list. Notice that certain authors define an inner product could be a function <·,·> that satisfies only the third of the above terms with a (weaker) non-degenerated (i.e., when = 0 for all w, then v=0) addition of the (weaker) condition. Functions fulfilling all four criteria are usually referred to in such literature as positive-definite interior products, while internal products that are not defined positively are often referred to as uncertain. While subtly, this disparity presents many remarkable phenomena. Inner products that are not positive-definite, for example, can create 'standards' under which those vectors (such vectors are called spacelike) have imaginary magnitude and cause 'metrics' that are not actual metrics. The inner product of Lorentzia is an example of an indefinite inner product.

An inner product space is called a vector space with an inner product this description also refers in any field to an abstract vector space.

1. The real numbers $\mathbb{R}$ of the inner product

<x,y>=xy.

Euclidean space R^N is an infinite set of all real N-tuples.

$$\langle \mathbf{u}, \mathbf{v} \rangle = \mathbf{u}^T\mathbf{v} = u_0v_0 + u_1v_1 + \dots + u_{N-1}v_{N-1} = \sum_{i=0}^{N-1} u_iv_i$$

eq -1

Where u and v are column vectors

3. Unit space through complex C number with inner product function. Inner product spaces over the field of complex numbers are sometimes referred to as unitary spaces.

$$\langle \mathbf{u}, \mathbf{v} \rangle = \mathbf{u}^{*T}\mathbf{v} = \sum_{i=0}^{N-1} u_i^*v_i = \langle \mathbf{v}, \mathbf{u} \rangle^*$$

eq -2

where * indicates the complex operation of the conjugate, and u and v are complex values column vector

4. The vector space C[a; b] of all real-valued continuous functions on a closed interval [a; b] is an inner product space, whose inner product is defined by

$$\langle f(x), g(x) \rangle = \int_a^b f^*(x)g(x)dx$$

eq -3

The norm or length of vector z in all three inner product spaces is as specified

$$\|z\| = \sqrt{\langle z, z \rangle}$$

eq -4

Equations through (5 to 15) are valid for all inner product spaces, including those defined by Eqs.

(3) to (4) . and the angle between two nonzero vectors z and w is

Although the meaning must always be considered, the term 'vector' is usually used for abstract vectors. A matrix (e.g., column vector) or a continuous function may be a vector.

$$\theta = \cos^{-1}\frac{\langle z, w \rangle}{\|z\|\|w\|}$$

eq-5

If the z norm is 1, z is considered to have been normalized. If it is said that Eq-5, z and w are orthogonal. A natural result of these definitions is that a number of non-zero vectors are jointly or equally orthogonal, provided it only

$$\langle w_k, w_l \rangle = 0 \quad \text{for } k \neq l$$

eq-6

They constitute an orthogonal basis for the inner product space .They are an orthonormal basis, when the basic vectors are normatively

$$\langle w_k, w_l \rangle = \delta_{kl} = \begin{cases} 0 & \text{for } k \neq l \\ 1 & \text{for } k = l \end{cases}$$

eq-7

Similarly, a set of w0, W1, W2, ... vectors and an additional set of $\widetilde{w}_0, \widetilde{w}_1, \widetilde{w}_2,$ dual vectors is said to be biorthogonal and biorthogonal basis vector space

$$\langle \widetilde{w}_k, w_l \rangle = 0 \quad \text{for } k \neq l$$

eq-8

They are a biorthonormal basis if and only if

$$\langle \widetilde{w}_k, w_l \rangle = \delta_{kl} = \begin{cases} 0 & \text{for } k \neq l \\ 1 & \text{for } k = l \end{cases}$$

eq-9

As a method for coherently representing an infinite set of vectors, the foundation of inner product space is among the most important principles in linear algebra. The subsequent derivation that depends on the orthogonally of the basis vectors is fundamental to the matrix-based transformations of the next section. Let W = {} 0 1 {}, w 2,... Be the orthogonal basis of the inner product area V, and let z V be supported. The vector z can then be expressed as the following linear combination of base vectors.

$$z = \alpha_0 w_0 + \alpha_1 w_1 + \alpha_2 w_2 + \ldots$$

eq-10

whose inner product with basis vector w_i is

$$\begin{aligned} \langle w_i, z \rangle &= \langle w_i, \alpha_0 w_0 + \alpha_1 w_1 + \alpha_2 w_2 + \ldots \rangle \\ &= \alpha_0 \langle w_i, w_0 \rangle + \alpha_1 \langle w_i, w_1 \rangle + \ldots + \alpha_i \langle w_i, w_i \rangle + \ldots \end{aligned}$$

eq-11

Since they are mutually orthogonal the inner products on the right side of Eq. (11) are 0 unless the vector subscriptions for which the inner products are computed match [see Eq. (7)]. Thus, the only nonzero word is to exclude zero terms and to divide the two sides of the equation by giving

$$\alpha_i = \frac{\langle w_i, z \rangle}{\langle w_i, w_i \rangle}$$

eq-12

Which reduces to

$$\alpha_i = \langle w_i, z \rangle$$

eq-13

If the basic vector norms are 1

$$\alpha_i = \frac{\langle \tilde{w}_i, z \rangle}{\langle \tilde{w}_i, w_i \rangle}$$

eq-14

And

$$\alpha_i = \langle \tilde{w}_i, z \rangle$$

eq-15

5.2 Matrix-Based Transforms

The 1-D discrete Fourier transform is one of a class of significant transformations that can be represented in a general relation. The word transform is used in mathematics to describe a change in form without any corresponding change in value.

$$T(u) = \sum_{x=0}^{N-1} f(x) r(x,u)$$

eq-16

Where x is a spatial variable, T(u) is the transform of f(x), r(x,u) is a forward transformation kernel, and integer u is a transform variable with values in the range Similarly, the inverse transform of T(u) is

$$f(x) = \sum_{u=0}^{N-1} T(u) s(x,u)$$

eq-17

If s(x,u) is a inverse transformation kernel and x is a transformation kernel for the range r(x,u), and s(x,u) in Eqs.(16) and (17) that only depends on x and u and not on f(x) or T(u) values, then the existence of the transform pair that is defined will be determined.

Graphical equations (17) are shown in Fig.5.1. It should be taken into consideration that the f(x) is a weighted total of N inverse (e.g. s(x,u) for) kernel functions and that the T(u) for weight . Each N s(x,u) contributes f(x) to each x. Expanding the right side of Eq. (17)

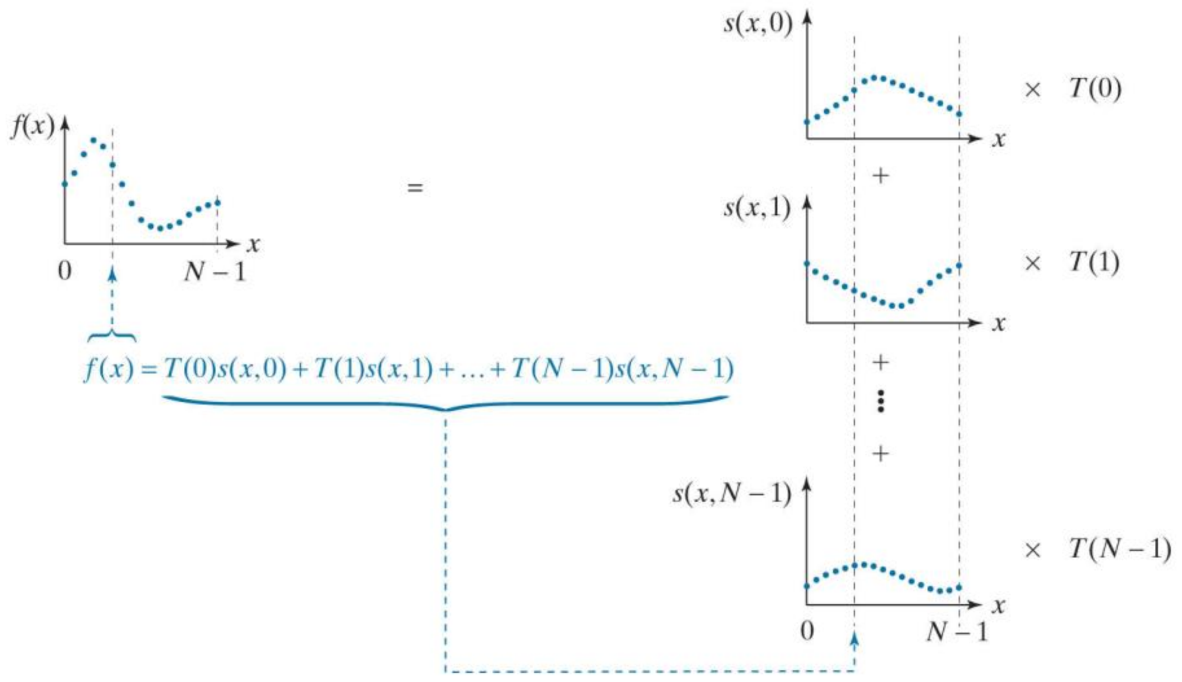


FIGURE 5.1

(Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

A graphical illustration of Eq. (18)

$$f(x) = T(0)s(x,0) + T(1)s(x,1) + \dots + T(N-1)s(x,N-1)$$

eq-18

It is readily evident which a linear expansion is represented as Eq. (11), with Eq. (18) s(x,u), T(u) in place (e.g. the basis of vectors) and Equ. (11). In this case, a linear expansion is immediately apparent. When we suppose s(x,u) is orthonormal vectors of a product's inner space in Eq. (18), Eq. (13) tells us that

$$T(u) = \langle\, s(x,u), f(x)\, \rangle$$

eq-19

And T(u) can be calculated for transform through the inner products

We frequently use subscripts to define the matrix or vector elements. This means the first column vector f element, which is f (0), and the first column of column vector element s (0, 3).

$$\mathbf{f} = \begin{bmatrix} f(0) \\ f(1) \\ \vdots \\ f(N-1) \end{bmatrix} = \begin{bmatrix} f_0 \\ f_1 \\ \vdots \\ f_{N-1} \end{bmatrix}$$

eq-
20

$$\mathbf{t} = \begin{bmatrix} T(0) \\ T(1) \\ \vdots \\ T(N-1) \end{bmatrix} = \begin{bmatrix} t_0 \\ t_1 \\ \vdots \\ t_{N-1} \end{bmatrix}$$

eq-21

And

$$\mathbf{s}_u = \begin{bmatrix} s(0,u) \\ s(1,u) \\ \vdots \\ s(N-1,u) \end{bmatrix} = \begin{bmatrix} s_{u,0} \\ s_{u,1} \\ \vdots \\ s_{u,N-1} \end{bmatrix} \quad \text{for } u = 0, 1, \ldots, N-1$$

eq-22

and using them to rewrite Eq. (19) as

$$T(u) = \langle \mathbf{s}_u, \mathbf{f} \rangle \quad \text{for } u = 0, 1, \ldots, N-1$$

eq-23

Combining the N-base vectors in the transform matrix

$$\mathbf{A} = \begin{bmatrix} \mathbf{s}_0^T \\ \mathbf{s}_1^T \\ \vdots \\ \mathbf{s}_{N-1}^T \end{bmatrix} = [\mathbf{s}_0 \quad \mathbf{s}_1 \quad \dots \quad \mathbf{s}_{N-1}]^T$$

eq-24

Then Eq. (23) is replaced into Eq. (21) and EQ. (1) can be used to achieve

By using Eq. (1), we assume that real-value vector is the most common case. For complex inner product space, equation (2) must be used.

$$\mathbf{t} = \begin{bmatrix} \langle \mathbf{s}_0, \mathbf{f} \rangle \\ \langle \mathbf{s}_1, \mathbf{f} \rangle \\ \vdots \\ \langle \mathbf{s}_{N-1}, \mathbf{f} \rangle \end{bmatrix} = \begin{bmatrix} s_{0,0}f_0 + s_{1,0}f_1 + \dots + s_{N-1,0}f_{N-1} \\ s_{0,1}f_0 + s_{1,1}f_1 + \dots + s_{N-1,1}f_{N-1} \\ \vdots \\ s_{0,N-1}f_0 + s_{1,N-1}f_1 + \dots + s_{N-1,N-1}f_{N-1} \end{bmatrix}$$

$$= \begin{bmatrix} s_{0,0} & s_{1,0} & \cdots & s_{N-1,0} \\ s_{0,1} & s_{1,1} & & \\ \vdots & & \ddots & s_{N-1,N-2} \\ s_{0,N-1} & & s_{N-2,N-1} & s_{N-1,N-1} \end{bmatrix} \begin{bmatrix} f_0 \\ f_1 \\ \vdots \\ f_{N-1} \end{bmatrix}$$

eq-25

Or

$$\mathbf{t} = \mathbf{Af}$$

eq-26

$$\mathbf{AA}^T = \begin{bmatrix} \mathbf{s}_0^T \\ \mathbf{s}_1^T \\ \vdots \\ \mathbf{s}_{N-1}^T \end{bmatrix} \begin{bmatrix} \mathbf{s}_0 & \mathbf{s}_1 & \dots & \mathbf{s}_{N-1} \end{bmatrix}$$

$$= \begin{bmatrix} \mathbf{s}_0^T\mathbf{s}_0 & \mathbf{s}_0^T\mathbf{s}_1 & \dots & \mathbf{s}_0^T\mathbf{s}_{N-1} \\ \mathbf{s}_1^T\mathbf{s}_0 & \mathbf{s}_1^T\mathbf{s}_1 & & \vdots \\ \vdots & & \ddots & \\ \mathbf{s}_{N-1}^T\mathbf{s}_0 & \dots & & \mathbf{s}_{N-1}^T\mathbf{s}_{N-1} \end{bmatrix}$$

$$= \begin{bmatrix} \langle \mathbf{s}_0, \mathbf{s}_0 \rangle & \langle \mathbf{s}_0, \mathbf{s}_1 \rangle & \dots & \langle \mathbf{s}_0, \mathbf{s}_{N-1} \rangle \\ \langle \mathbf{s}_1, \mathbf{s}_0 \rangle & \langle \mathbf{s}_1, \mathbf{s}_1 \rangle & & \vdots \\ \vdots & & \ddots & \\ \langle \mathbf{s}_{N-1}, \mathbf{s}_0 \rangle & \dots & & \langle \mathbf{s}_{N-1}, \mathbf{s}_{N-1} \rangle \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & & \vdots \\ \vdots & & \ddots & \\ 0 & \dots & & 1 \end{bmatrix} = \mathbf{I}$$

eq-27

Where Eqs. (1) and (7) respectively follow the two last steps. Since **AA^T=I** it gives Eq (26) premultiplication by **A^T** and simplification gives **f=A^{T}t** Eqs. (16) and (17) thus become the matrix-based pair of transformations.

$$\mathbf{t} = \mathbf{Af}$$ eq-28

And

$$\mathbf{f} = \mathbf{A}^T\mathbf{t}$$ eq-29

It is essential to understand that we assumed that the N transform basis vectors (i.e. s $_u$ for u =0, 1….N-1) of the Transformation matrix A were true and orthonormal in the derivations of Eqs (28) and (29). In compliance with Eq (7)

$$\langle \mathbf{s}_k, \mathbf{s}_l \rangle = \mathbf{s}_k^T\mathbf{s}_l = \delta_{kl} = \begin{cases} 0 & k \neq l \\ 1 & k = l \end{cases}$$ eq-30

The assumed orthonormality makes it possible to calculate forward transforms without explicit references to a kernel of forward transformation, i.e. **t=Af**, when A is a function acts in the reverse transformation kernel s(x,u).

Because of the reality that the base vector A is real and orthonormal, the transform is termed the orthogonal transform defined in Eq. (28). It preservatives the inner products is i.e., **$\langle f_1, f_2 \rangle$** = **$\langle t_1, t_2 \rangle$** = **$\langle A f_1, A f_2 \rangle$** and therefore, the distances and angles pre and post transformation among vectors. The rows and columns of A are orthonormal **$AA^T = A^TA$** = I, so **$A^{-1} = A^T$.** The outcome in the Eqs. (28) And Eq(29) are a reversible transform pair . Substituted by Eq. (29) with (28) produces **$t= Af= AA^Tt=t$**, while replacing Eq. (28) with (29), **$f= A^Tt=A^TAF=f$** gives.

$$T(u,v) = \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x,y) r(x,y,u,v)$$

eq-31

And

$$f(x,y) = \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} T(u,v) s(x,y,u,v)$$

eq-32

Where the inverse transformation kernels r(x,y,u,v) and s(x,y,u,v) respectively are forward . Transform T(u,v) and invert kernel transformation s(x,y,u,v), respectively, with eq. (32) way to set a linear expansions to f(x,y), are regarded as another weighting coefficients and basis vectors . Forward transformation kernel r(x,y,u,v) is separable if

$$r(x,y,u,v) = r_1(x,u) r_2(y,v)$$ eq-33

and symmetric if $r_1$is functionally equal to $r_2$so

$$r(x,y,u,v) = r_1(x,u) r_1(y,v)$$ eq-34

When the transformation kernels are real and orthonormal; both r and s are distinguishable and symmetrical, then the matrix equivalents for Eqs. (6-31) and (6-32) shall be substitute r in Eqs. (33) and r in Eqs. (34), respectively, for separable, separable symmetrical inverse kernels.

$$\mathbf{T=AFA^T}$$ eq-35

And

$$\mathbf{F = A^T T A} \qquad \textbf{eq-36}$$

When F is a matrix of f(x, y), T is a transform and A is as defined in the Equation (24) Equation. The column and row transformations of F are measured according to the F pre and post-multiplication by A and Eq. (35). This actually breaks the 2-D transformation into two 1-D transformations, which mirror the 2-D DFT process.

EXAMPLE 1 A simple orthogonal transformation.

Consider the 2-element basis vectors

$$\mathbf{s}_0 = \frac{1}{\sqrt{2}}\begin{bmatrix}1\\1\end{bmatrix} \quad \text{and} \quad \mathbf{s}_1 = \frac{1}{\sqrt{2}}\begin{bmatrix}1\\-1\end{bmatrix}$$

And note they are orthonormal in accordance with Eq. (30) :

$$\langle\, \mathbf{s}_0, \mathbf{s}_1 \,\rangle = \mathbf{s}_0^T\mathbf{s}_1 = \tfrac{1}{2}\begin{bmatrix}1 & 1\end{bmatrix}\begin{bmatrix}1\\-1\end{bmatrix} = \tfrac{1}{2}(1-1) = 0$$

$$\langle\, \mathbf{s}_1, \mathbf{s}_0 \,\rangle = \mathbf{s}_1^T\mathbf{s}_0 = \tfrac{1}{2}\begin{bmatrix}1 & -1\end{bmatrix}\begin{bmatrix}1\\1\end{bmatrix} = \tfrac{1}{2}(1-1) = 0$$

$$\langle\, \mathbf{s}_0, \mathbf{s}_0 \,\rangle = \mathbf{s}_0^T\mathbf{s}_0 = \tfrac{1}{2}\begin{bmatrix}1 & 1\end{bmatrix}\begin{bmatrix}1\\1\end{bmatrix} = \tfrac{1}{2}(1+1) = 1$$

$$\langle\, \mathbf{s}_1, \mathbf{s}_1 \,\rangle = \mathbf{s}_1^T\mathbf{s}_1 = \tfrac{1}{2}\begin{bmatrix}1 & -1\end{bmatrix}\begin{bmatrix}1\\-1\end{bmatrix} = \tfrac{1}{2}(1+1) = 1$$

Substitute and transformation matrix in or out of Eq. (6-24).

$$\mathbf{A} = \begin{bmatrix}\mathbf{s}_0 & \mathbf{s}_1\end{bmatrix}^T = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\1 & -1\end{bmatrix} \qquad \text{eq-37}$$

and the transform of matrix

$$\mathbf{F} = \begin{bmatrix}20 & 63\\21 & 128\end{bmatrix}$$

follows from Eq. (35)

$$\mathbf{T} = \left(\frac{1}{\sqrt{2}}\right)^2 \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 20 & 63 \\ 21 & 128 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}^T$$

$$= \frac{1}{2} \begin{bmatrix} 41 & 191 \\ -1 & -65 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 232 & -150 \\ -66 & 64 \end{bmatrix}$$

$$= \begin{bmatrix} 116 & -75 \\ -33 & 32 \end{bmatrix}$$

In accordance with Eq. (36) , the inverse of transform T is

$$\mathbf{F} = \left(\frac{1}{\sqrt{2}}\right)^2 \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}^T \begin{bmatrix} 116 & -75 \\ -33 & 32 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 83 & -43 \\ 149 & -107 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} = \begin{bmatrix} 20 & 63 \\ 21 & 128 \end{bmatrix}$$

In conclusion, we note that A is an orthogonal transformation matrix for which

$$\mathbf{AA}^T = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}^T = \frac{1}{2} \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \mathbf{I}$$

Rectangular Arrays

When rectangular arrays are transformed, Eqs. (35) and (36) are changed. In comparison to square.

$$\mathbf{T} = \mathbf{A}_M \mathbf{F} \mathbf{A}_N^T \quad \textbf{eq-38}$$

And

$$\mathbf{F} = \mathbf{A}_M^T \mathbf{T} \mathbf{A}_N \quad \text{eq-39}$$

where F, $\mathbf{A_M}$ and $\mathbf{A_N}$ are of size MXN ,MXM,NXN and respectively. Both and are defined in accordance with Eq. (24)

EXAMPLE 2: Computing the transform of a rectangular array.

A simple transformation in which M and N are 2 and 3, respectively, is

$$\mathbf{T} = \mathbf{A}_2\mathbf{F}\mathbf{A}_3^T = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\1 & -1\end{bmatrix}\begin{bmatrix}5 & 100 & 44\\6 & 103 & 40\end{bmatrix}\frac{1}{\sqrt{3}}\begin{bmatrix}1 & 1 & 1\\1 & 0.366 & -1.366\\1 & -1.366 & 0.366\end{bmatrix}^T$$

$$= \frac{1}{\sqrt{6}}\begin{bmatrix}11 & 203 & 84\\-1 & -3 & 4\end{bmatrix}\begin{bmatrix}1 & 1 & 1\\1 & 0.366 & -1.366\\1 & -1.366 & 0.366\end{bmatrix} = \begin{bmatrix}121.6580 & -12.0201 & -96.1657\\0 & -3.0873 & 1.8624\end{bmatrix}$$

Where matrices F, $\mathbf{A}_2$ **and** $\mathbf{A}_3$ in the first step of the computation are defined. As expected, 2X3 the transforming output T is the same as F

To demonstrate that it is an orthogonal matrix of transformation, and that Eq reverses transformation (39)

Complex Orthonormal Basis Vectors

An orthonormal basis is the basis of which vectors are orthogonal to each other and have a unit norm.

Orthonormal bases are essential in applications because a vector is especially simple to represent on an orthonormal basis, called Fourier expansion.

We have to be familiar with the concepts of internal product and standard to understand this concept.

Orthonormal sets

Remember that if the inner product equals zero, two vectors are orthogonal.

Definition Let S have an inner product vector space <...>. A set of K vectors $S_1 \ldots S_k$ Є S, if and only if are an orthonormal set.

$$\langle \mathbf{s}_k, \mathbf{s}_l \rangle = \langle \mathbf{s}_l, \mathbf{s}_k \rangle^* = \mathbf{s}_k^{*T}\mathbf{s}_l = \delta_{kl} = \begin{cases}0 & k \neq l\\1 & k = l\end{cases}$$ eq-40

Where * is the complex operation of the conjugate. If vectors of basis (35) and (36) are problematic as opposed to real-value,

$T=AFA^T$ eq-41

And

$F=A^{*T}\,TA^*$ eq-42

Transformation matrix A is then considered the unitary matrix and the unitary transformation pair Eq (41) and Eq(42) is equal. The 1-D counterpart of Eq. (41) and (42) are thus an essential and useful property of A:

Orthogonal transformations are a unique case wherein the functions of expansion are real-valued.

Both transforms maintain inner products.

$$\mathbf{t=Af} \qquad \textbf{eq-43}$$

$$\mathbf{f=A^{*T}\,t} \qquad \textbf{eq-44}$$

EXAMPLE 3: A transform with complex-valued basis vectors.

The inverse of a unitary transformation matrix, which is the inverse of the transformation matrix, is the inverse.

$$\mathbf{A} = \frac{1}{\sqrt{3}}\begin{bmatrix} 1 & 1 & 1 \\ 1 & -0.5-j0.866 & -0.5+j0.866 \\ 1 & -0.5+j0.866 & -0.5-j0.866 \end{bmatrix}$$

eq-45

is its conjugate transpose. Thus,

$$\mathbf{A}^{*T}\mathbf{A} = \frac{1}{\sqrt{3}}\begin{bmatrix} 1 & 1 & 1 \\ 1 & -0.5-j0.866 & -0.5+j0.866 \\ 1 & -0.5+j0.866 & -0.5-j0.866 \end{bmatrix}^{*T} \frac{1}{\sqrt{3}}\begin{bmatrix} 1 & 1 & 1 \\ 1 & -0.5-j0.866 & -0.5+j0.866 \\ 1 & -0.5+j0.866 & -0.5-j0.866 \end{bmatrix}$$

$$= \frac{1}{3}\begin{bmatrix} 1 & 1 & 1 \\ 1 & -0.5+j0.866 & -0.5-j0.866 \\ 1 & -0.5-j0.866 & -0.5+j0.866 \end{bmatrix}\begin{bmatrix} 1 & 1 & 1 \\ 1 & -0.5-j0.866 & -0.5+j0.866 \\ 1 & -0.5+j0.866 & -0.5-j0.866 \end{bmatrix}$$

$$= \frac{1}{3}\begin{bmatrix} 3 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 3 \end{bmatrix} = \mathbf{I}$$

Where $j=\sqrt{-1}$ the matrix and where A is a single matrix, which in Eqs (41) can be used through eq(44).. It can be shown easily if the base vectors in A satisfy Eq. (40).

Biorthonormal Basis Vectors

Expansion functions $S_0, S_1 \ldots, S_{N-1}$ are biorthonormal in Eq. (24) if there are dual expansion functions ~S0, $\sim S_1 \ldots, \sim S_{N-1}$ that

$$\langle \tilde{\mathbf{s}}_k, \mathbf{s}_l \rangle = \delta_{kl} = \begin{cases} 0 & k \neq l \\ 1 & k = l \end{cases}$$ eq-46

Even the expansion functions and dual functions do not need to be orthonormal. With a set of biorthonormal expansion features, Eqs. (35) and (36) are transformed

$$\mathbf{T} = \tilde{\mathbf{A}}\mathbf{F}\tilde{\mathbf{A}}^T$$ eq-47

And

$$\mathbf{F} = \mathbf{A}^T\mathbf{T}\mathbf{A}$$ eq-48

Transformation matrix A remains as described by Eq. (24); dual transformation matrix $\tilde{\mathbf{A}} = [\tilde{\mathbf{s}}_0 \quad \tilde{\mathbf{s}}_1 \quad \cdots \quad \tilde{\mathbf{s}}_{N-1}]^T$ is an NX N matrix .The dual expansion functions of its rows are transposed. When the function of expansion and its duals are identical, i.e. $\tilde{\mathbf{s}}_u = \mathbf{s}_u$, reduces the Equal Eq(47) and Eq(48) respectively to Eq. (35) and Eq(36) are

$$\mathbf{t} = \tilde{\mathbf{A}}\mathbf{f}$$ eq-49

$$\mathbf{f} = \mathbf{A}^T\mathbf{t}$$ eq-50

EXAMPLE 4 : A biorthonormal transform.

Consider the real biorthonormal transformation matrices

$$\mathbf{A} = \begin{bmatrix} 0.5 & 0.5 & 0.5 & 0.5 \\ -1 & -1 & 1 & 1 \\ -0.5303 & 0.5303 & -0.1768 & 0.1768 \\ -0.1768 & 0.1768 & -0.5303 & 0.5303 \end{bmatrix} \text{ and } \tilde{\mathbf{A}} = \begin{bmatrix} 0.5 & 0.5 & 0.5 & 0.5 \\ -0.25 & -0.25 & 0.25 & 0.25 \\ -1.0607 & 1.0607 & 0.3536 & -0.3536 \\ 0.3536 & -0.3536 & -1.0607 & 1.0607 \end{bmatrix}$$

A and $\tilde{\mathbf{A}}$ are biorthonormal .The transform of 1-D column vector f=[30 11 210 6] ^{T} is

$$\mathbf{t} = \tilde{\mathbf{A}}\mathbf{f} = \begin{bmatrix} 0.5 & 0.5 & 0.5 & 0.5 \\ -0.25 & -0.25 & 0.25 & 0.25 \\ -1.0607 & 1.0607 & 0.3536 & -0.3536 \\ 0.3536 & -0.3536 & -1.0607 & 1.0607 \end{bmatrix} \begin{bmatrix} 30 \\ 11 \\ 210 \\ 6 \end{bmatrix} = \begin{bmatrix} 128.5 \\ 43.75 \\ 51.9723 \\ -209.6572 \end{bmatrix}$$

$$\langle \mathbf{f},\mathbf{f} \rangle = \mathbf{f}^T\mathbf{f} = \begin{bmatrix} 30 & 11 & 210 & 6 \end{bmatrix} \begin{bmatrix} 30 \\ 11 \\ 210 \\ 6 \end{bmatrix} = 45,157$$

And <t,t>=t^Tt =65,084 which is unequal to <f,f> the transformation have not preserve inner products. but, reversible:

$$\mathbf{f} = \mathbf{A}^T\mathbf{t} = \begin{bmatrix} 0.5 & 0.5 & 0.5 & 0.5 \\ -1 & -1 & 1 & 1 \\ -0.5303 & 0.5303 & -0.1768 & 0.1768 \\ -0.1768 & 0.1768 & -0.5303 & 0.5303 \end{bmatrix}^T \begin{bmatrix} 128.5 \\ 43.75 \\ 51.9723 \\ -209.6572 \end{bmatrix} = \begin{bmatrix} 30 \\ 11 \\ 210 \\ 6 \end{bmatrix}$$

The forward and inverse transforms are calculated by using Eqs. (49) and (50) , respectively.

Finally, we mention that the majority of the concepts in this area can be used to extend the form continuously.

$$f(x) = \sum_{u=-\infty}^{\infty} \alpha_u s_u(x)$$

The coefficients and base vectors for the expansion of the internal product space C([a, b]) by α_u and S_u for u= $0, \pm 1, \pm 2, \dots$ for given f(x) and basis s_u(x) for u= $0, \pm 1, \pm 2, \dots$ .The appropriate coefficients of expansion may be calculated by defining of Integral Internal Product of the C([a, b])-i.e., Eq's -3 and all internal product spaces' general properties, that is, Eqs. (10) by (15). Thus, for e.g. s_u(x) for u= $0, \pm 1, \pm 2, \dots$. are orthonormal basis vectors of C([a, b])

$$\alpha_u = \langle s_u(x), f(x) \rangle \qquad \text{eq-52}$$

EXAMPLE 5: The Fourier series and discrete Fourier transform

Consider the representation in linear expanding of orthonormal base vectors of the form of the continuous periodic function T

$$s_u(x) = \frac{1}{\sqrt{T}} e^{j2\pi ux/T} \quad \text{for } u = 0, \pm 1, \pm 2, \dots$$

eq-53

In accordance with Eqs. (51) and (52)

$$f(x) = \sum_{u=-\infty}^{\infty} \alpha_u \left[\frac{1}{\sqrt{T}} e^{j2\pi ux/T}\right]$$

$$= \frac{1}{\sqrt{T}} \sum_{u=-\infty}^{\infty} \alpha_u e^{j2\pi ux/T}$$

eq-54

And

$$\alpha_u = \langle s_u(x), f(x) \rangle$$

$$= \int_{-T/2}^{T/2} \left[\frac{1}{\sqrt{T}} e^{j2\pi ux/T}\right]^* f(x)dx$$

$$= \frac{1}{\sqrt{T}} \int_{-T/2}^{T/2} f(x)e^{-j2\pi ux/T}dx$$

eq-55

yields the following discrete counterparts of Eqs. (53) through (55)

$$s(x,u) = \frac{1}{\sqrt{N}} e^{j2\pi ux/N} \quad \text{for } u = 0, 1, \ldots, N-1$$

eq-56

$$f(x) = \frac{1}{\sqrt{N}} \sum_{u=0}^{N-1} T(u)e^{j2\pi ux/N}$$

eq-57

And

$$T(u) = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} f(x)e^{-j2\pi ux/N}$$

eq-58

The discrete complex basis vectors of Eq. (56) are an orthonormal basis of inner product space C^N Equations (58) and (59)

Consider, Eqs (55) and (58), in the calculation of Fourier and the Fourier discrete transform of f(x)=2sin(2πx) of period of T=1 are now used . In accordance with Eq. (55)

$$\alpha_1 = \int_{-1/2}^{1/2} \left[\frac{1}{\sqrt{1}} e^{j2\pi(1)x/1}\right]^* \sin(2\pi x)dx$$

$$= \int_{-1/2}^{1/2} e^{-j2\pi x} \sin(2\pi x)dx = \int_{-1/2}^{1/2} [\cos(2\pi x) - j\sin(2\pi x)]\sin(2\pi x)dx$$

$$= \frac{1}{4\pi}\sin^2(2\pi x) - j\left[\frac{x}{2} - \frac{1}{8\pi}\sin(4\pi x)\right]\Bigg|_{-1/2}^{1/2} = -j0.5$$

and, in the same way ,α_{-1}= j0.5 hence entire another coefficients are zero, the resultant Fourier series is

$$f(x) = j0.5e^{-j2\pi x} - j0.5e^{j2\pi x} \quad \text{eq-59}$$

5.3 Correlation

The correlation of f and g indicated by two continual functions f(x) and g(x) are denoted by $f \star g(\Delta x)$, is defined as

In fact, the term cross correlation should be used when f(x) ≠ g(x) and auto-correlation when f(x)=g(x) Equation (61) is valid for both cases

$$f \star g(\Delta x) = \int_{-\infty}^{\infty} f^*(x)g(x+\Delta x)dx$$
$$= \langle f(x), g(x+\Delta x) \rangle \quad \text{eq-61}$$

Here the final step are follows from Eq. (3) with α =-∞ and α=∞ Sometimes related to as the sliding inner product f and g, the correlation tests the similarity of f(x) and g(x) according to their relative displacement △x . if △x =0.

As the name suggests, sliding inner product, display slide between functions, multiply them and compute the area. The functions become increasingly identical as the area increases.

$$f \star g(0) = < f(x) ,g(x) > \quad \text{eq-62}$$

And Eq. (52) defining continuous orthonormal expansion coefficients in Eq. (51) can be written as alternatively

$$\alpha_u = <f ,S_u> =f \star s_u (0)$$

Thus, the expansion coefficients are correlations of one point, with zero displacement △ . The similarity of f(x) and su(x) is measured by α_x

The discrete equivalents of Eqs. (61) through (63) are Integers n and m, fn indicates the nth element in f, and g_{n+m} implies the element (n+m) in g. Equation (66) is the result of Equation (65) and (23)

$$\mathbf{f} \star \mathbf{g}(m) = \sum_{x=-\infty}^{\infty} f_n^* g_{n+m}$$

eq-64

$$f \star g(0) = \langle f,g \rangle \quad \text{eq-65}$$

and

$$T(u) = \langle S_u, f \rangle = S_u \star f(0)$$

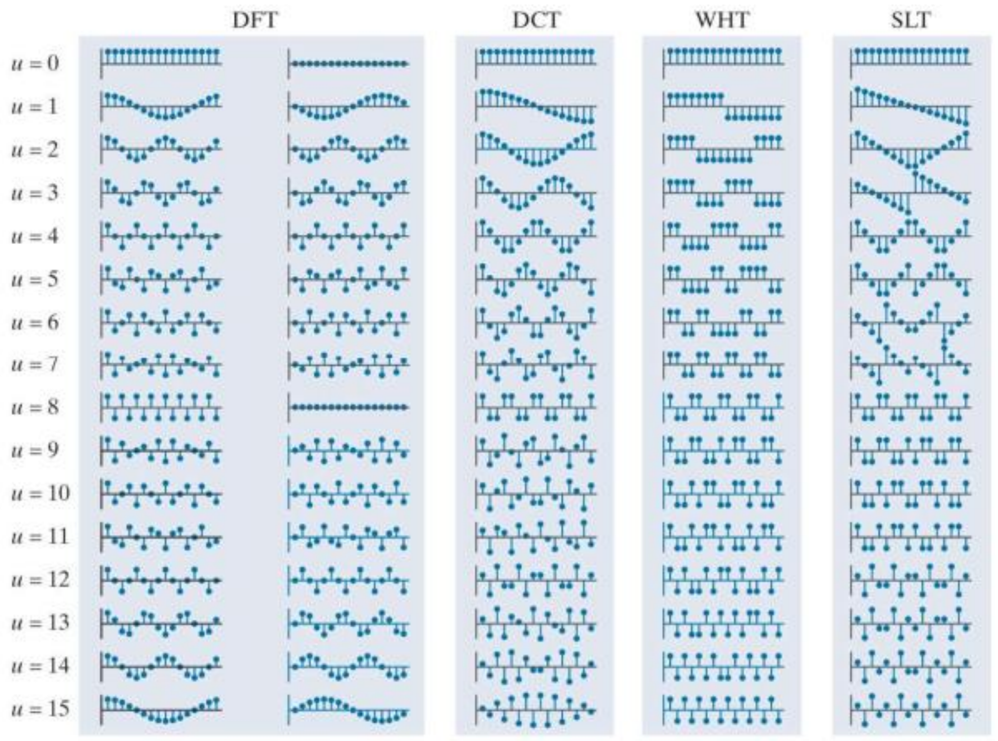
Similar comments on Eq (63) and continuous series expansions and continuing transforms in Eq (66) may be made. Every orthogonal transformation element and orthogonal transformations are discrete. Each orthogonal transformation element.

5.4 Basis Functions in the Time-Frequency Plane

As Fig.5.2 , which shows the basis vectors of certain commonly encountered transformations, most orthogonal basis sets of sinusoids, square waves, ramps and other small waves called wavelets are associated with mathematical activity. If h(t) is a vector base and g(t) is the function transformed, the transform coefficient is a measure of the similarity between g and h as noted in the previous section. The large g ☆ h(0) values show that g and h have significant time and frequency characteristics (shape, bandwidth, etc.). Therefore, if h is the ramp shaped base function in figure (d), then transform coefficient could be utilized to identify linear gradients of brightness in a line of the image. If h is a sinusoidal function, like that of Fig. (a), on the other hand it is possible to use g ☆ h(0). Plots such as those in Fig. 5.2 and a similarity measure like this can reveal much about transforming the function's time-frequency features

Independent variables T and F instead of spatial variables x and u are used in the introduction to the time frequency plane. g(t) and h(f) continuous functions are replaced by f(x) and $S_u(x)$. Although concepts use continuous functions and variables, they also apply to distinct functions and variables

a b c d
e f g h


DFT
DCT
WHT
SLT
u = 0
u = 1
u = 2
u = 3
u = 4
u = 5
u = 6
u = 7
u = 8
u = 9
u = 10
u = 11
u = 12
u = 13
u = 14
u = 15

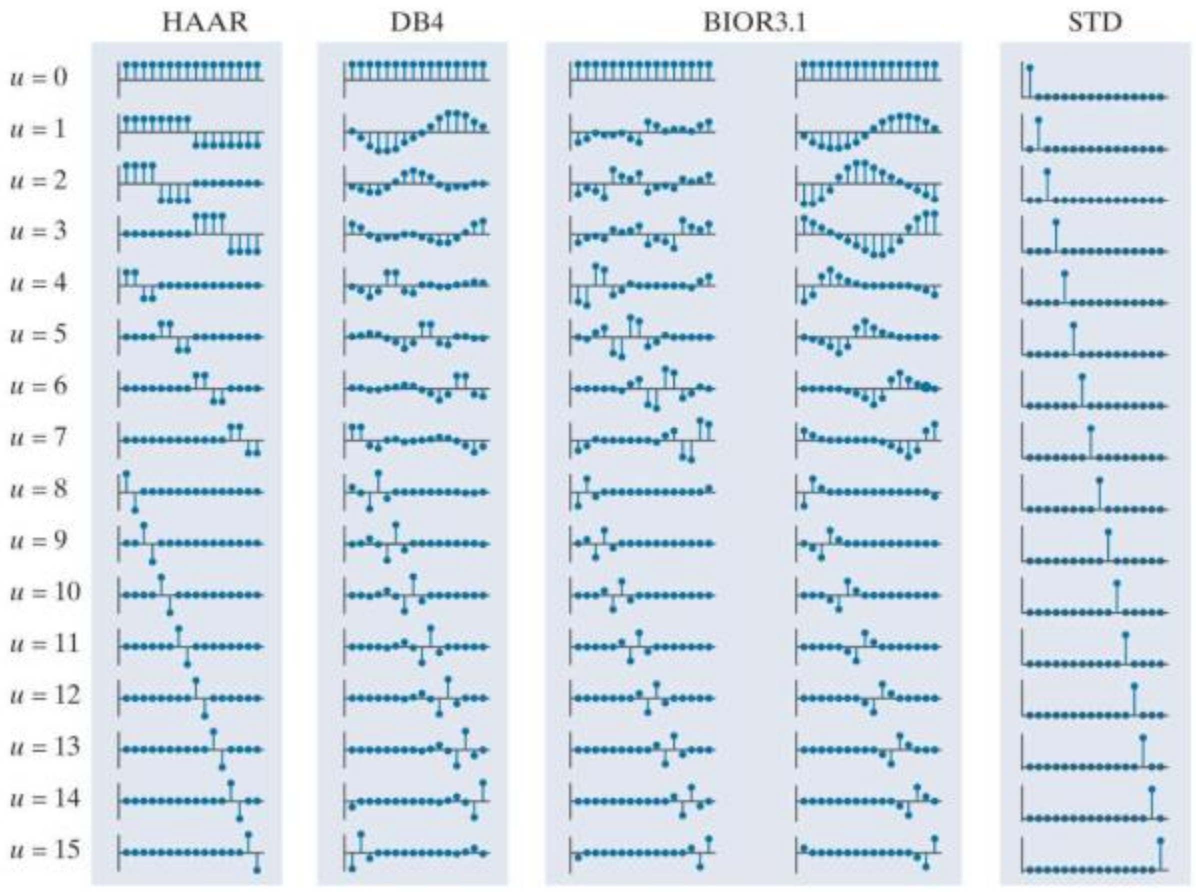

HAAR
DB4
BIOR3.1
STD
u = 0
u = 1
u = 2
u = 3
u = 4
u = 5
u = 6
u = 7
u = 8
u = 9
u = 10
u = 11
u = 12
u = 13
u = 14
u = 15

FIGURE 5.2 Basis vectors (for) of some commonly encountered transforms: (a) Fourier basis (real and imaginary parts), (b) discrete Cosine basis, (c) Walsh-Hadamard basis, (d) Slant basis, (e) Haar basis, (f) Daubechies basis, (g) Biorthogonal B-spline basis and its dual, and (h) the standard basis, which is included for reference only (i.e., not used as the basis of a transform).

(Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

The position of h on the plane of time-frequency Fig. 5.3 is a strictly objective descriptor of h and therefore g for large values of (a). let $p_h = |h(t)|^2 / ||h(t)^2||^2$ be a probability density function with mean

In Eq. (67) , every value of t is weighted by p_h to calculate a weighted mean with respect to coordinate t

$$\mu_t = \frac{1}{\|h(t)\|^2}\int_{-\infty}^{\infty} t|h(t)|^2 dt$$

eq-67

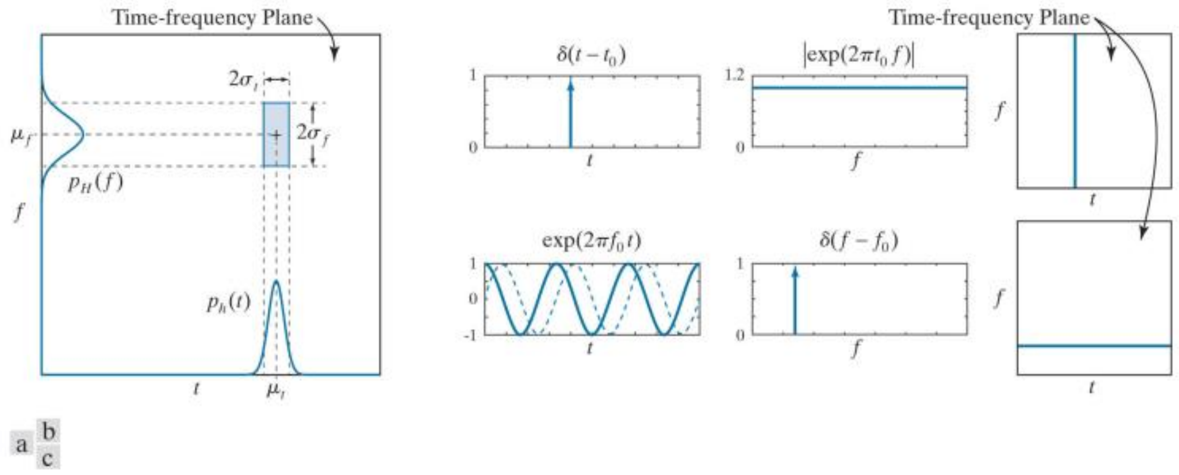


FIGURE 5.3

(a) Basis function localization in the time-frequency plane. (b) A standard basis function, its spectrum, and location in the time frequency plane. (c) A complex sinusoidal basis function (with its real and imaginary parts shown as solid and dashed lines, respectively), its spectrum, and location in the time-frequency plane.

(Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

and variance

$$\sigma_t^2 = \frac{1}{\|h(t)\|^2}\int_{-\infty}^{\infty}(t-\mu_t)^2|h(t)|^2dt \quad \text{eq-68}$$

Let $p_H(f) = |H(F)|^2 / \|H(F)^2\|^2$ be a probability density function with mean

$$\mu_f = \frac{1}{\|H(f)\|^2}\int_{-\infty}^{\infty} f|H(f)|^2 df \quad \text{eq -69}$$

and variance

$$\sigma_f^2 = \frac{1}{\|H(f)\|^2}\int_{-\infty}^{\infty}(f-\mu_f)^2|H(f)|^2df$$

The Fourier transform of H is where f denotes frequency. H(f) (t). Then, as shown in Fig. 5.3(a), the energy of base function h is concentrated at (μ_t ,μ_f) on the time-frequency plane. In a rectangular region called the Heisenberg box or a cell, the majority of the energy comes from such an area $4\sigma_t\ \sigma_f$

The energy of continuous function h(t) is $\int_{-\infty}^{\infty}|h(t)^2|$dt

So if indicated in angular frequency, the constant on the right side of Eq. (71) even with a Gaussian base function, whose transform is also a Gaussian feature, equal treatment is feasible.

$$\sigma_t^2\sigma_f^2 \geq \frac{1}{16\pi^2} \quad \text{eq-71}$$

Because a function support can be described as the series of points where the function is non-zero, the principle of uncertainty in Heisenberg tells us that a function can't be supported on time and on frequency at all. Reveal so that both σ_t and σ_f it can't be very small. Hence the basis function $\delta(t-t_0)$ illustrated in Fig. 5.3(b) is correctly localized with the time[i.e $\sigma_t = 0$, hence the width of $\delta(t-t_0)$ is equal to zero], its spectrum is also nonzero on the whole f-axis. i.e, hence $\xi\delta(f-f_0) = \exp(-j2\pi f t_0)$ and $|\exp(-j2\pi f t_0)|$=1 for entire f , $\sigma_f = \infty$, The output is an infinitesimal small, infinitely large Heisenberg cell on the time-frequency plane. Basis function $\exp(2\pi f_0 t)$ of Fig. 5.3(c) , on the other hand, is primarily, nonzero on the broad time axis, nonetheless is clearly localized in frequency. Because $\xi\{\exp(-j2\pi f t_0) = \delta(f-f_0)\}$ spectrum $|\delta(f-f_0)|$ is zero at fully frequencies other than The resultant Heisenberg cell is infinitely comprehensive and infinitesimally small in height $\sigma_{\omega=0}$ As Figs. 5.3(b) and (c) illustrate, perfect localization in time is accompanied by a loss of localization in frequency and vice versa.

Repeating once more to Fig. 5.2, notice the basis for the DFT of Fig. 5.2(a) and the standard basis of Fig. 5.2(h) in Fig. 5.3(c) and (b), accordingly, are discrete examples (for N=16) of the impulses and complex exponential functions. In the top half of Fig. 5.2, the other basis is both the index u frequency and the width or support 16. For a given u, their locations are similar in the time frequency plane. This is especially obvious when u is 8 and the basic functions of the Heisenberg cells are the same. For all the other u parameters $\mu_{t,}$ $\sigma_{t\,.}$ μ_t and $\sigma_{t\,.}$ of the heisenberg cell and their value is similar, where there are slight variations between cosine, ramp or square wave forms. Similarly, with exception of the already described standard basis, the basis function of the bottom half of Fig. 5.2 is also analogous for a given u. These basis functions are small waves called wavelets of form that are scaled and shifted

The DFT basis functions do not seem to be ordered frequency due to aliases.

$$\psi_{s,t}\ (t) = 2^{s/2}\ \psi(2^{s}\ t - \tau\,)$$

In which s and τ are integers and mother wavelet $\psi(t)$ is a real, square-integrable function having a bandpass-like spectrum. Parameter decides the position of $\psi_{s,t}\ (t)$ on the t-axis, s decided its width—that is, how broad or narrow it is along the t-axis, and 2^{s} manage its amplitude.

, the functions analogous to in Fig. 5.2 have lowpass spectra and are known as scaling functions. Eq (72) generates a basis which is defined by the cells Heisenberg on the right side of Fig. 5.4 along with a properly constructed mother wavelet

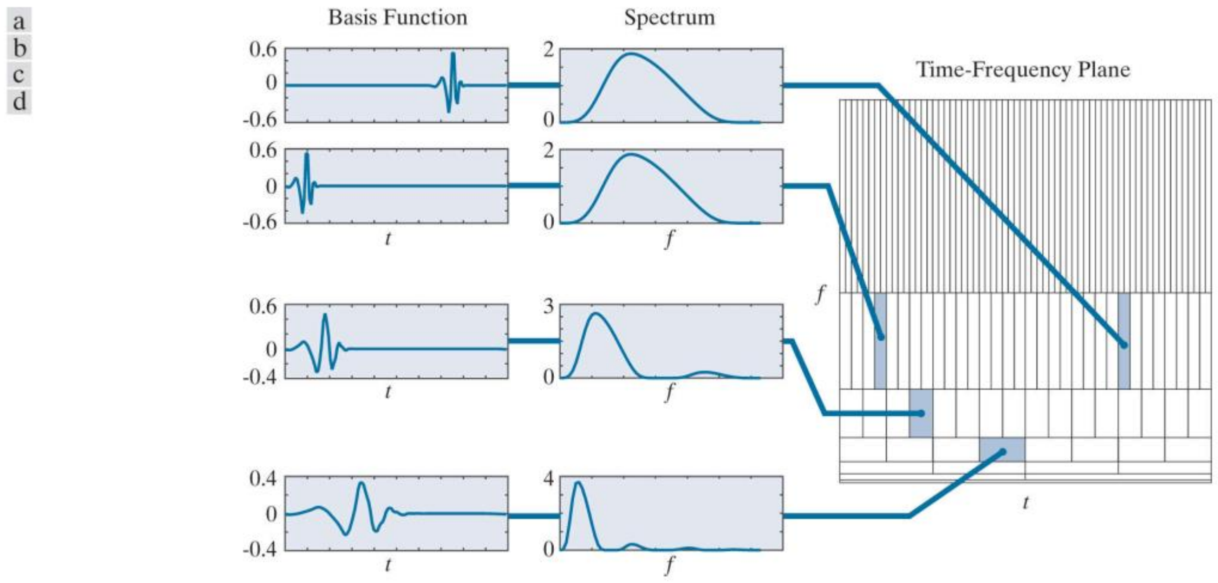


FIGURE 5.4 Time and frequency localization of 128-point Daubechies basis functions.

(Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

The proof of Eq. (73)

$$\Im\{\psi(2^t\ t)\} = \frac{1}{2^s}\ \Psi\left(\frac{f}{2^s}\right) \text{ Eq -73}$$

And the spectrum is spread for positive s values — each component of the frequency is shifted by more than one by factor of 2^S. The compressing time expands the spectrum as was the case for a rectangular pulse. It is shown in Figs 5.4(b) – (d) graphically. Note that in Fig. 5.4(c), the width of the base function is half as in (d) whereas its width of its spectrum is twice that of (d). It is shifted by a factor of two higher frequency. The same could be explained in Fig. 5.4(b) in comparison to (c) the base function and spectrum. Halving time support and doubling frequency support, Heisenberg cells of different widths and heights are produced with the same area. In addition, each cell row to the right of Fig. 5.4 represents a single scale s and frequency range. The cells in a row are moved over time in relation to each other.

$$\Im\{\psi(t-\tau) = e^{-j2\pi\tau f}\ \Psi(f) \qquad \text{eq-74}$$

Thus $|\Im\{\psi(t-\tau)| = |\Psi(f)|$ and the spectra of the time-shifted wavelets are identical.

5.4 Basis Images

As the inverse transformation kernel s(x,y,u,v) in Eq. (32) only retains the indices x,y,u,v, rather than f(x,y) or T(u,v), Eq. (32) can also be used as the matrix sum of the kernel

$$\mathbf{F} = \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} T(u,v)\mathbf{S}_{u,v} \qquad \text{eq-75}$$

where F is an matrix having the elements of f(x, y) and

$$\mathbf{S}_{u,v} = \begin{bmatrix} s(0,0,u,v) & s(0,1,u,v) & \dots & s(0,N-1,u,v) \\ s(1,0,u,v) & \vdots & \dots & \vdots \\ \vdots & \vdots & \dots & \vdots \\ \vdots & \vdots & \dots & \vdots \\ \vdots & & & \\ s(N-1,0,u,v) & s(N-1,1,u,v) & \dots & s(N-1,N-1,u,v) \end{bmatrix}$$

eq-76

For $u\,,v = 0, \dots.,N-1$ F is then clearly defined as a linear combination of N^2 matrices having size NXN , that is , the $S_{u.v}$ for $u\,,v = 0, \dots.,N-1$, when the underlying s(x,y,u,v) are real-valued, independent, and symmetric

$$S_{u,v=}\ S_u\ S_v^T$$

Eq-77

Where S_u and S_v defined by Eq. (20) are previously. F is a 2-D image and is called basic images in the context of digital image processing. As shown in Fig. 5.5(a), it can be arranged in a array to give a concise view of 2-D basis functions they represent.

EXAMPLE : The basis images of the standard basis.

In figure 5.2(h), $e_n$the basis of the NX1column vector, which is 1 in nth and all other elements are 0, would be a certain instance (N=16) of a standard basis $e_o\,, e_{1\,,\dots.}\,, e_{N-1}$. Since it is real orthonormal, and the corresponding orthogonal transformation matrix is during the corresponding 2-D transform. In other words, the transformation of F on the standard base is F—confident that a discrete function is implicitly represented in respect of the standard basis if it is written in vector form.

The basis image of the 2-D size standard base Figure 5.5(b) Just as the 1-D-basis vectores, that are non zero at just one moment (or x-value), are not zero at just one point on a xyplane, the basis images of Fig. 5.5(b) are nonzero. This is a result of Eq (77) , hence $s_{u,v}\ = e_u\, e_v^T\ = E_{u,v}$ where $E_{u,v}$ is a matrix with zeros having a 1 in the uth row and vth column . in similar way , the DFT basis shown in fig 5.6. which follow from Eq. (77) , Eq. (22) , and the defined equation of the 1-D DFT expansion functions [i.e., Eq. (56)]. Note the DFT basis image of maximum frequency occurs when u and v are 4, just as the 1-D DFT basis function of maximum frequency occurred at in Fig. 5.1

a b

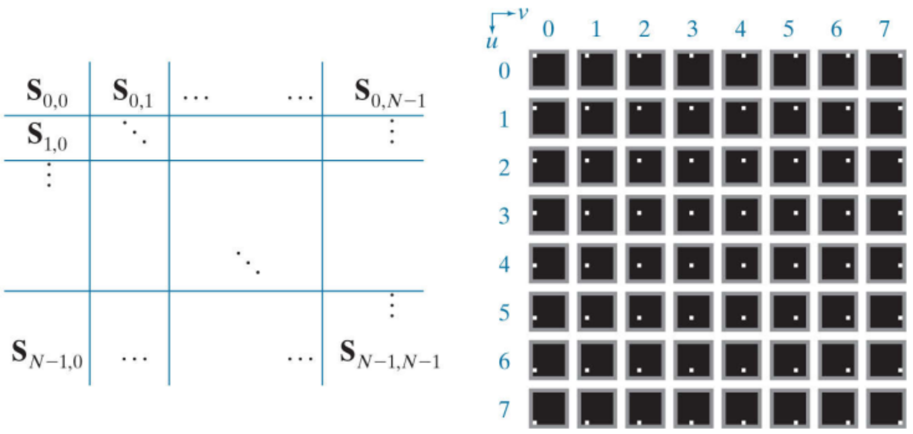
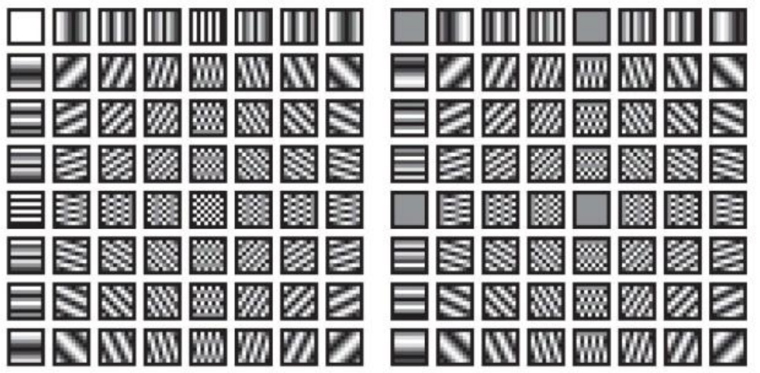


FIGURE 5.5 (a) Basis image organization and (b) a standard basis of size For clarity, a gray border has been added around each basis image. The origin of each basis image (i.e., $x = y = 0$) is at its top left

(Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

$$\frac{1}{\sqrt{8}}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & \omega & -j & -j\omega & -1 & -\omega & j & j\omega \\ 1 & -j & -1 & j & 1 & -j & -1 & j \\ 1 & -j\omega & j & \omega & -1 & j\omega & -j & -\omega \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & -\omega & -j & j\omega & -1 & \omega & j & -j\omega \\ 1 & j & -1 & -j & 1 & j & -1 & -j \\ 1 & j\omega & j & -\omega & -1 & -j\omega & -j & \omega \end{bmatrix}$$



a b c

FIGURE 5.6 (a) Tranformation matrix A_F of the discrete Fourier transform for N=8 where $\omega = e^{-j2\pi/8}$ *or* $(1-j)/\sqrt{2}$ or (b) and (c)

(Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

The real and imaginary parts of the DFT basis images of size 8X8 For clarity, a black border has been added around each basis image. For 1-D transforms, matrix A_F is used in conjunction with Eqs. (43) and (44) ; for 2-D transforms, it is used with Eqs. (6-41) and (6-42).

5.6 Fourier-Related Transforms

The Fourier real function transformation is highly complex. In this section we discuss three transforms associated to Fourier which are real rather than complex: the discrete Hartley transform, discrete cosine transform, and discrete sine transform. All three transforms handle unnecessary numbers' computational complexity and can be carried out via fast FFT-like algorithms.

The Discrete Hartley Transform

A discrete Hartley Transform (DHT) is a Fourier-related transformation of discrete, periodic data analogous to the discrete Fourier Transform (DFT), with analogue applications in signal processing and associated fields .Its primary difference from DFT would be that it transforms real inputs to real outputs without involving complex numbers intrinsically. Much like DFT is the discreet analogue of the continuous Fourier transformation (FT), DHT is the discrete analogue of the Hartley transform, which Ralph V. L. Hartley introduced in 1942.

The transformation matrix of the discrete Hartley transform (DHT) is obtained by replacing the inverse transformation kernel.

The function cas, the terminology for the cosine-and-sin function, is defined as $cas(\theta) = \cos(\theta) + \sin(\theta)$

$$s(x,u) = \frac{1}{\sqrt{n}}\, cas\left(\frac{2\pi ux}{N}\right)$$

$$= \frac{1}{\sqrt{n}}\left[cos\left(\frac{2\pi ux}{N}\right) + sin\left(\frac{2\pi ux}{N}\right)\right] \quad \text{eq -78}$$

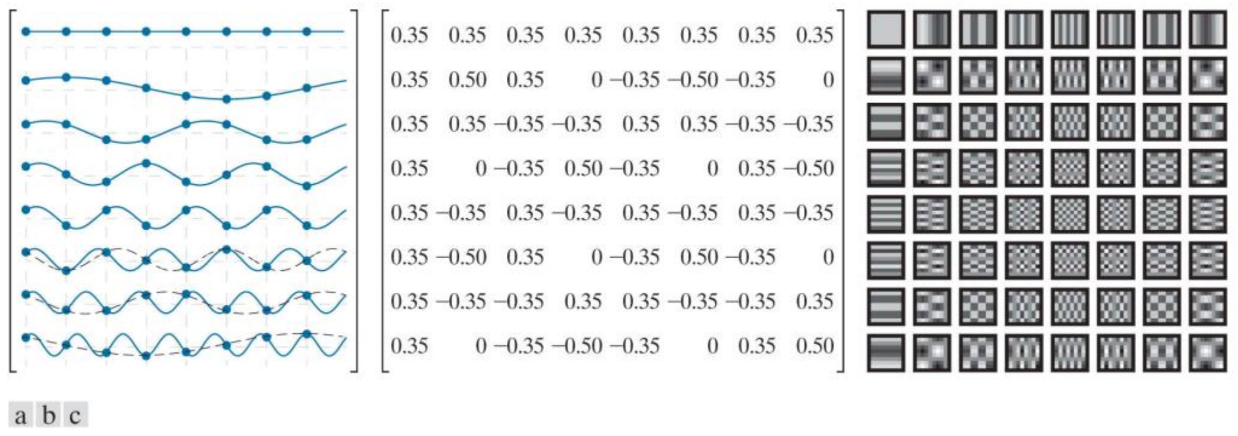
Whose divisible 2-D complement is

We shall not consider the non- divisible form $s(x,y,u,v) = \frac{1}{N}\, cas\left(\frac{2\pi(ux+vy)}{N}\right)$

$$s(x,y,u,v) = \left[\frac{1}{\sqrt{n}}\, cas\left(\frac{2\pi ux}{N}\right)\right]\left[\frac{1}{\sqrt{n}}\, cas\left(\frac{2\pi vy}{N}\right)\right]$$

eq -79

Meanwhile the resultant DHT transformation matrix—represented $\boldsymbol{A_{HY}}$ in Fig. 5.7 —is real, orthogonal, and symmetric, $\boldsymbol{A_{HY} = A_{HY}^{T} = A_{HY}^{-1}}$ ***and*** $\boldsymbol{A_{HY}}$ should be used in the calculation of both forward and inverse transform. For 1-D transforms, A_{HY} is used in combination with Eqs. (6-28) and (29) for 2-D transforms, Eqs. (35) and (36) are used. Since A_{HY} is symmetric, the forward and inverse transforms are same.



$$\begin{bmatrix} 0.35 & 0.35 & 0.35 & 0.35 & 0.35 & 0.35 & 0.35 & 0.35 \\ 0.35 & 0.50 & 0.35 & 0 & -0.35 & -0.50 & -0.35 & 0 \\ 0.35 & 0.35 & -0.35 & -0.35 & 0.35 & 0.35 & -0.35 & -0.35 \\ 0.35 & 0 & -0.35 & 0.50 & -0.35 & 0 & 0.35 & -0.50 \\ 0.35 & -0.35 & 0.35 & -0.35 & 0.35 & -0.35 & 0.35 & -0.35 \\ 0.35 & -0.50 & 0.35 & 0 & -0.35 & 0.50 & -0.35 & 0 \\ 0.35 & -0.35 & -0.35 & 0.35 & 0.35 & -0.35 & -0.35 & 0.35 \\ 0.35 & 0 & -0.35 & -0.50 & -0.35 & 0 & 0.35 & 0.50 \end{bmatrix}$$

FIGURE 5.7

The transformation matrix and basis images of the discrete Hartley transform for (a) Graphical representation of orthogonal transformation matrix A_{HY} (b) A_{HY} rounded to two decimal places, and (c) 2-D basis images.

(Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

For 1-D transforms, matrix A_{HY} is used in conjunction with Eqs. (28) and (29) ; for 2-D transforms, it is used with Eqs. (35) and (36)

Note the resemblance of the harmonically associated DHT basic functions in Figure. 5.7(a) and the real part of the DFT basic functions in Fig. 5.1. It's simple to demonstrate that

$$\boldsymbol{A_{HY} = Real\{A_F\} - Imag\{A_F\}}$$

$$\boldsymbol{= Real\{1 + j\}\{A_F\}} \qquad \textbf{eq-(80)}$$

Where $\boldsymbol{A_F}$ is the unit transformation matrix of the DFT. In addition, given the real part of the DFT kernel.

In Eqs. (81) and (82) , subscripts **HY** and **F** are used to represent the Hartley and Fourier kernels, respectively.

$$Re\{s_F\,(x,u)\} = Re\left\{\frac{1}{\sqrt{N}}\,e^{\frac{j2\pi ux}{N}}\right\} = \frac{1}{\sqrt{N}}\,cos\left(\frac{2\pi ux}{N}\right)$$

eq-(81)

The discrete Hartley kernel can be rewritten using triginometric identity $cas(\theta) = \sqrt{2}\cos\,(\theta - \frac{\pi}{4})$[see Eq. (78)] as

$$s_H\ (x,u) = \sqrt{\frac{2}{n}}\ \cos\left(\frac{2\pi ux}{N} - \frac{\pi}{4}\right)$$ **eq-(82)**

The basic functions of the Fourier and Hartley discrete transformations will be scaled $\sqrt{2}$ and shifted by $\pi/4$, i.e. scaled and shifted by one another. When comparing Figures 5.1 and 5.7, the shift is evident (a)

The Discrete Cosine Transform

The most frequently found form discrete cosine transform (DCT) is achieved by replacing the inverse transformations kernel. There are 8 standard DCT variations and various symmetry conditions are assumed. For example, even about a sample or about a point between two samples could be assumed to be the entry

$$s(x,u) = \alpha(u)\cos\left(\frac{(2x+1)u\pi}{2N}\right)$$

eq-83

Where

$$\alpha(u) = \begin{cases} \sqrt{\frac{1}{N}} & \text{for } u = 0 \\ \sqrt{\frac{2}{N}} & \text{for } u = 1, 2, \ldots, N-1 \end{cases}$$

eq-84

5.7 Walsh-Hadamard Transforms

An orthogonal transformation technique, the Walsh-Hadamard transform breaks down a signal into a set of basis functions. It is non-sinusoidal and orthogonal in nature. Walsh functions, that are rectangular or square waves with values of +1 or –1, represent as the basis functions for these equations. Walsh-Hadamard transforms are also referred to as Hadamard transforms Walsh transforms, or Walsh-Fourier transforms, depending on the context.

$$s(x,u) = \frac{1}{\sqrt{N}}(-1)^{\sum_{i=0}^{n-1} b_i(x)b_i(u)}$$

(eq-85)

This corresponds to the modulo 2 arithmetic operation performed in the exponent of Eq. (85) and is the kth bit in the binary representation of z. For example, if n=3 and z=6 and (110 in binary), and If $b_0(2) = 0$, $b_1(z) = 1$ and $b_z(z) = 1$ if N=2 the resulting Hadamard-ordered transformation matrix is

$$\mathbf{A}_W = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

WHTs with Hadamard or natural ordering are denoted by the letter $\mathbf{A}_w$**,** which stands for Hadamard or natural order transformation matrix. Although it is of size 2X2 in this case, it is more commonly of size N X N, where N X N is the dimension of the discrete function being transformed.

where the matrix on the right (without the scalar multiplier) is called a Hadamard matrix of order 2. Letting $\mathbf{H}_N$ denote the Hadamard matrix of order N, a simple recursive relationship for generating Hadamard-ordered transfomation matrices is

$$\mathbf{A}_W = \frac{1}{\sqrt{N}}\mathbf{H}_N$$

Where

$$\mathbf{H}_{2N} = \begin{bmatrix} \mathbf{H}_N & \mathbf{H}_N \\ \mathbf{H}_N & -\mathbf{H}_N \end{bmatrix}$$

And

$$\mathbf{H}_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

$$\mathbf{H}_4 = \begin{bmatrix} \mathbf{H}_2 & \mathbf{H}_2 \\ \mathbf{H}_2 & -\mathbf{H}_2 \end{bmatrix}$$

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

And

$$\mathbf{H}_8 = \begin{bmatrix} \mathbf{H}_4 & \mathbf{H}_4 \\ \mathbf{H}_4 & -\mathbf{H}_4 \end{bmatrix}$$

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

5.9 Slant Transform

The N x N Slant transform matrices are defined by the recursion

$$\mathbf{S}_n = \frac{1}{\sqrt{2}} \left[\begin{array}{cc|c|cc|c} 1 & 0 & \mathbf{0} & 1 & 0 & \mathbf{0} \\ a_n & b_n & & -a_n & b_n & \\ \hline \multicolumn{2}{c|}{\mathbf{0}} & \mathbf{I}_{(N/2)-2} & \multicolumn{2}{c|}{\mathbf{0}} & I_{(N/2)-2} \\ \hline 0 & 1 & \mathbf{0} & 0 & -1 & \mathbf{0} \\ -b_n & a_n & & b_n & a_n & \\ \hline \multicolumn{2}{c|}{\mathbf{0}} & I_{(N/2)-2} & \multicolumn{2}{c|}{\mathbf{0}} & -\mathbf{I}_{(N/2)-2} \end{array}\right] \left[\begin{array}{c|c} \mathbf{S}_{n-1} & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{S}_{n-1} \end{array}\right]$$

where $N = 2^n$, IM, denotes an M x M identity matrix, and

$$\mathbf{S}_1 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

The parameters a_n and b_n are defined by the recursions

$$b_n = (1 + 4a_{n-1}^2)^{-1/2}, \qquad a_1 = 1$$
$$a_n = 2b_n a_{n-1}$$

which solve to give

$$a_{n+1} = \left(\frac{3N^2}{4N^2-1}\right)^{1/2}, \qquad b_{n+1} = \left(\frac{N^2-1}{4N^2-1}\right)^{1/2}, \qquad N = 2^n$$

Using these formulas, the 4 x 4 Slant transformation matrix is obtained as

$$\begin{array}{cc} & \text{Sequency} \\ \mathbf{S}_2 = \dfrac{1}{2} \begin{bmatrix} 1 & 1 & 1 & 1 \\ \frac{3}{\sqrt{5}} & \frac{1}{\sqrt{5}} & \frac{-1}{\sqrt{5}} & \frac{-3}{\sqrt{5}} \\ 1 & -1 & -1 & 1 \\ \frac{1}{\sqrt{5}} & \frac{-3}{\sqrt{5}} & \frac{3}{\sqrt{5}} & \frac{-1}{\sqrt{5}} \end{bmatrix} & \begin{array}{c} 0 \\ 1 \\ 2 \\ 3 \end{array} \end{array}$$

Properties of the Slant Transform

(i) The slant transform is real and orthogonal.

$S = S^*$

$S^{-1} = S^T$

(ii) The slant transform is fast, it can be implemented in (N log2N) operations on an N x 1 vector.

(iii) The energy deal for images in this transform is rated in very good to excellent range.

(iv) The mean vectors for slant transform matrix S are not sequentially ordered for n ≥ 3.

5.9 Haar Transform

In general, the Haar functions **$h_k(x)$** are described on the continuum interval x ∈ [0, 1] for k = 0,...,N—1 for which N = 2n. The integer k could be partitioned in a unique way as follows:

$$k = 2^P + q - 1$$

Where 0≤p≤n-1; q = 0, 1 for p = 0 and $1 \le q \le 2^P$ for p≠ 0. For example, when N = 4, we have

k	0	1	2	3
p	0	0	1	1
q	0	1	1	2

Representing k by (p ,q), the Haar functions are defined as

$$h_0(x) \triangleq h_{0,0}(x) = \frac{1}{\sqrt{N}}, \qquad x \in [0,1].$$

$$h_k(x) \triangleq h_{p,q}(x) = \frac{1}{\sqrt{N}} \begin{cases} 2^{p/2}, & \frac{q-1}{2^p} \le x < \frac{q-\frac{1}{2}}{2^p} \\ -2^{p/2}, & \frac{q-\frac{1}{2}}{2^p} \le x < \frac{q}{2^p} \\ 0, & \text{otherwise for } x \in [0,1] \end{cases}$$

In order to produce the Haar transform, it is necessary to allow x take discrete values at m/N, where m =0, 1,..., N-1. The Haar transform is defined as follows for N = 8.

$$\mathbf{Hr} = \frac{1}{\sqrt{8}} \begin{array}{c} \\ \left[\begin{array}{cccccccc} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ \sqrt{2} & \sqrt{2} & -\sqrt{2} & -\sqrt{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sqrt{2} & \sqrt{2} & -\sqrt{2} & -\sqrt{2} \\ 2 & -2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & -2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & -2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & -2 \end{array}\right] \end{array} \begin{array}{c} \text{Sequency} \\ 0 \\ 1 \\ 2 \\ 2 \\ 2 \\ 2 \\ 2 \\ 2 \end{array}$$

Properties of the Haar Transform

1. The Haar transform is real and orthogonal. Therefore,

$$Hr = Hr^*$$

$$Hr^{-1} = Hr^T$$

2. In computing, the Haar transform is an extremely rapid transformation. If you have a N x 1 vector, you can do it in O (N) operations, which is quite fast.

3. The basis vectors of the Haar matrix are ordered in a frequency-ordered manner.

4. For images, the energy compaction of the Haar transform is weak.

5.10 Wavelet Transforms

The wavelet transform is quite close to the Fourier transform (or much more similar to the windowed Fourier transform), but it has an entirely different merit function than the Fourier transform. The most significant distinction is as follows: On one hand, the Fourier transform breaks down the input signal into sines and cosines, i.e., functions that are localised in Fourier space; on the other hand, the wavelet transform employs functions that are localised in both real and Fourier space. For the most part, the wavelet transform may be stated mathematically using the following equation:

$$F(a,b) = \int_{-\infty}^{\infty} f(x)\, \psi^{*}_{(a,b)}(x)\, \mathrm{d}x$$

where * denotes the complex conjugate symbol and function ψ denotes a function of some kind. This function can be chosen at random, as long as it complies with a few rules.

Because of this, the Wavelet transform is actually an infinite collection of different transforms that can be computed depending on the merit function that is employed in its computation. This is the primary reason why we can hear the word "wavelet transform" used in a variety of various settings and applications, including computer science. There are a variety of methods for categorizing the many types of wavelet transformations. We just present the division based on the wavelet orthogonally. We can employ orthogonal wavelets for discrete wavelet transform development and non-orthogonal wavelets for continuous wavelet transform development while developing discrete wavelet transforms, for example. The following are the characteristics of these two transformations:

After performing the discrete wavelet transform, a data vector of the same length as the input is returned. In most cases, even in this vector, many data points are close to zero. This corresponds to the fact that it decomposes into a set of wavelets (functions) that are orthogonal to the translations and scaling of the original image data. As a result, we decompose such a signal into a wavelet coefficient spectrum with the same or smaller number of wavelet coefficients as the number of signal data points. Because there is no redundant information in a wavelet spectrum of this type, it is particularly well suited for signal processing and compression applications.

Instead, the continuous wavelet transform yields an array that is one dimension bigger than the input data. We can obtain an image of the time-frequency plane from a single-dimensional data set. We can clearly see the progression of the signal's frequencies over the course of the signal's duration and compare the spectrum to the spectra of other signals. Because the non-orthogonal collection of wavelets is utilized in this analysis, the data is strongly correlated, resulting in a significant amount of redundancy. This makes it easier to see the outcomes in a more humane light.

Discrete Wavelet Transform

With the discrete wavelet transform (DWT), you can achieve a more precise implementation of the wavelet transform by employing a discrete subset of wavelet scales and translations that adhere to certain constraints. For lack of a better term, this transform decomposes the signal into a set of wavelets that are mutually orthogonal to one another, which is the primary difference between it and the continuous wavelet transform (CWT), or its discrete time series implementation, which is sometimes referred to as discrete-time continuous wavelet transform (DT-CWT).

The wavelet can be generated from a scaling function that explains the scaling qualities of the wavelet's scaling properties. The requirement that the scaling functions must be orthogonal to their discrete translations imposes some mathematical requirements on them, which are referenced throughout the text, for example, the dilation equation, which can be found here.

$$\phi(x) = \sum_{k=-\infty}^{\infty} a_k \phi(Sx - k)$$

Color Image Processing

The utilization of color for image processing is inspired by two key factors. First, color is an important descriptor that also simplifies the identification and extraction of objects from the image. Color image processing is divided into two main parts: full color and pseudo-color processing. In the initial group, the images in question are usually taken with a full-color sensor, such as a color TV camera or a color scanner. In the second group, the issue is to assign a color to a specific monochrome intensity or set of intensities. Until recently, much of the digital color image processing was performed at the pseudo-color level. Even so, over the last decade, color sensors and color image processing hardware have become available at affordable prices. As a result, full-color image processing techniques are now used in a wide variety of applications, including publishing, visualization and the Internet.

- After this chapter is complete, learners can understand color fundamentals and the color spectrum.
- Know many of the color models used in digital image processing.
- Understand how to implement fundamental techniques, including intensity slicing and intensity-to-color transformations in pseudo-color image processing.
- You know how to decide whether a gray-scale approach can be extended to color pictures.
- Know how to work with full color images, including color transformations, color add-ons and tone/color corrections.

- Have to know the effect of noise in the processing of color images.
- Know how to perform spatial filtering on color images.
- Understand the benefits of using color in the segmentation of images.

5.11 Color Fundamentals

Introducing and discussing the visible light spectrum. A series of component colors includes the visible light, also known as white light. The light passes through triangular prism and these colors are also observed. Figure : The colors of the white light on the prism – red, orange, yellow, green, blue and violet – are separated. Dispersion is defined as the isolation of visible light into its various colors.

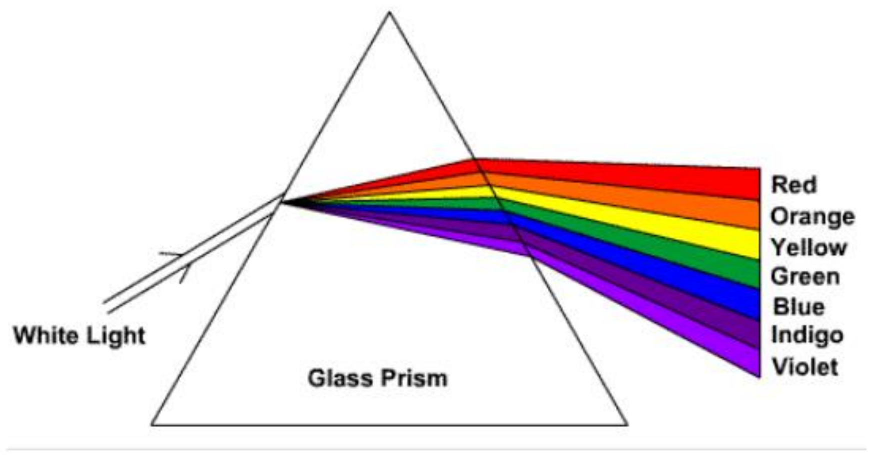


Figure 5.8 Color spectrum seen by passing white light through a prism.

James Clerk Maxwell demonstrated that electromagnetic radiation is a form of light. Radio waves, visible light and rays contain this radiation. As a radiation range, which extends beyond visible radiation to include one side of the radio wave and the other side of the gamma rays, Figure 2 indicates electra-magnetic radiation. A very small part of the electromagnetic spectrum occupies the visible light field. The sunlight is visible and stretches beyond the red (infraround IR) and ultraviolet (UV) with a maximum of yellow intensity.

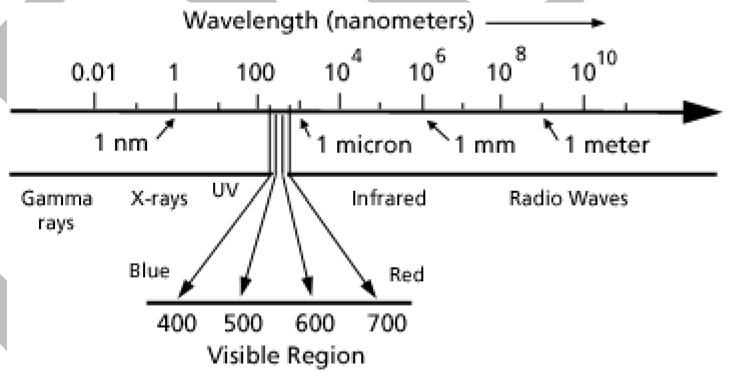


Figure 5.9. The electromagnetic spectrum

Fig.5.9. The electromagnetic spectrum, which encompasses the visible region of light, extends from gamma rays with wave lengths of one hundredth of a nanometer to radio waves with wave lengths of one meter or greater.

In regard to light as an electromagnetic wave, the spectral signature of a colour can be recognised by its wavelength. We perceive the waves as colour, the shorter the wavelength violet and the longest wave is red. Visible light is the range of electromagnetic wavelengths to which the eye reacts. The human eye is unable to adapt to radiation of longer or shorter wave lengths.

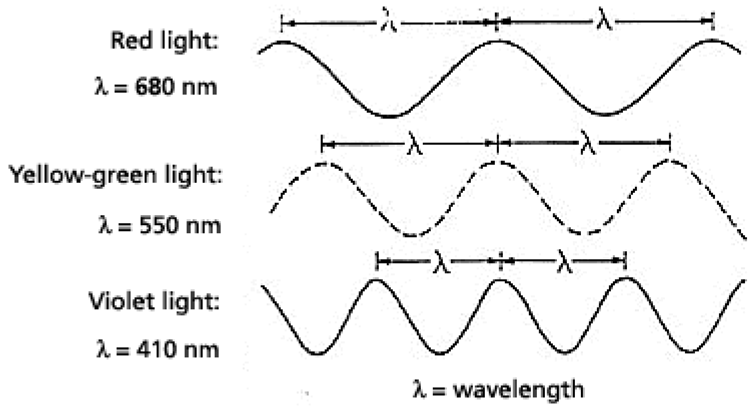


Figure 5.9. A wave representation of three different light hues: red, yellow-green and violet

Fig 5.9. A wave representation of three different light hues: red, yellow-green and violet, each with a different wavelength, which represents the distance between wave crests.

Fig.5.9 shows three standard waves of visible light. The longitude of the wave is the distance from the crest to the next and the lambda,λ, is indicated in Greek. Violet light is a 410 nanometer wavelength electromagnetic radiation with 680 Nanometers of red light.

Hundreds of thousands of different colors and intensities can be distinguishable from the human visual system, but just about 100 grey shades. Therefore, a lot of additional information can be found within an image and this additional information can be used in order to simplify image analyses, e.g. object identification and extraction based on color.

To define a specific color, three separate quantities are used. The color of the wavelength is the dominant one. The colors visible on the electromagnetic spectrum range from approximately 410nm (violet) to 680nm (red), as shown in Figure 3.

The saturation depends on the purity of excitation and on the amount of white light combined with the hue. There is a pure hue completely saturated, i.e. no mixed white light. Hue and saturation combined decide the chromaticity of a specified color. Ultimately, the actual amount of light determines the intensity with a more intense colors corresponding to the more light.

Achromatic light has no color - only quantity or intensity is its attribute. Grey level is an intensity measurement. The intensity of the energy is determined by the physical quantity. Brightness or luminance, on the other hand, is defined by color perception and is also psychological. Blue is perceived to be as darker than green, given the intensity of blue and green. Note also that our perception of intensity is nonlinear, as normalized intensity variations are considered to be the same changes in brightness between 0.1 and 0.11 and 0.5 to 0.55.

Color mainly depends on the object's reflecting properties. We see the reflected rays, while some are absorbed. The color and function of the human visual system must also be taken into consideration. For instance, when green but no red light lights it, the object which reflects red or green, appears green, and in the absence of green light, the object appears red. In pure white light, it will appear yellow (= red + green).

Tristimulus theory of color perception

There are 3 kinds of cones in the human retina. Figure 4 shows the response of each type of cone in accordance with the wavelength of the incident light. At 440nm (blue), 545nm (green) and 580nm the peaks for each curve are (red). Note that the last two in the yellow range of the spectrum actually peak.

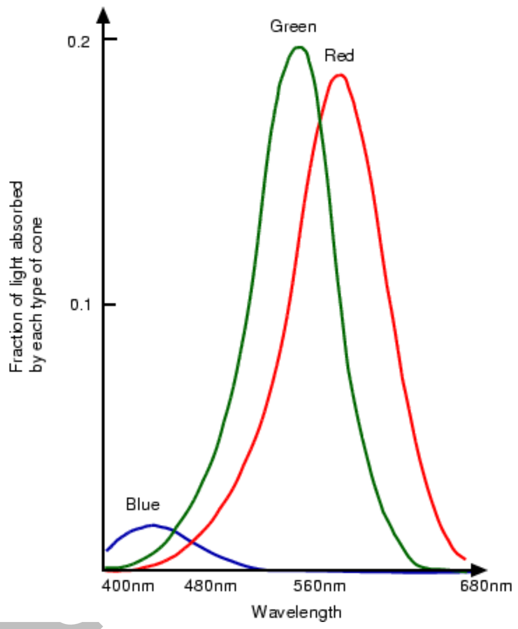


Figure 5.10: Spectral response curves for each cone type. The peaks for each curve are at 440nm (blue), 545nm (green) and 580nm (red).

(Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

CIE primaries

The tristimulus color perception theory seems to suggest that a combination of three primaries – red, green and blue – can produce any color, while almost all visible colors can be corresponded with one another, but some cannot. However, when one of the primary colors, it can be balanced by a combination of the other two, and thus a negative weighting of the primary can be considered for this particular one.

In 1931, the Commission Internationale de l'Éclairage (CIE), which could be applied to the shape of all visible colors, specified three basic primaries, the names X, Y and Z. The primary Y is selected such that the color match function matches the luminous efficiency function of the human eye precisely, as shown in figure 2 for the sum of the three curves.

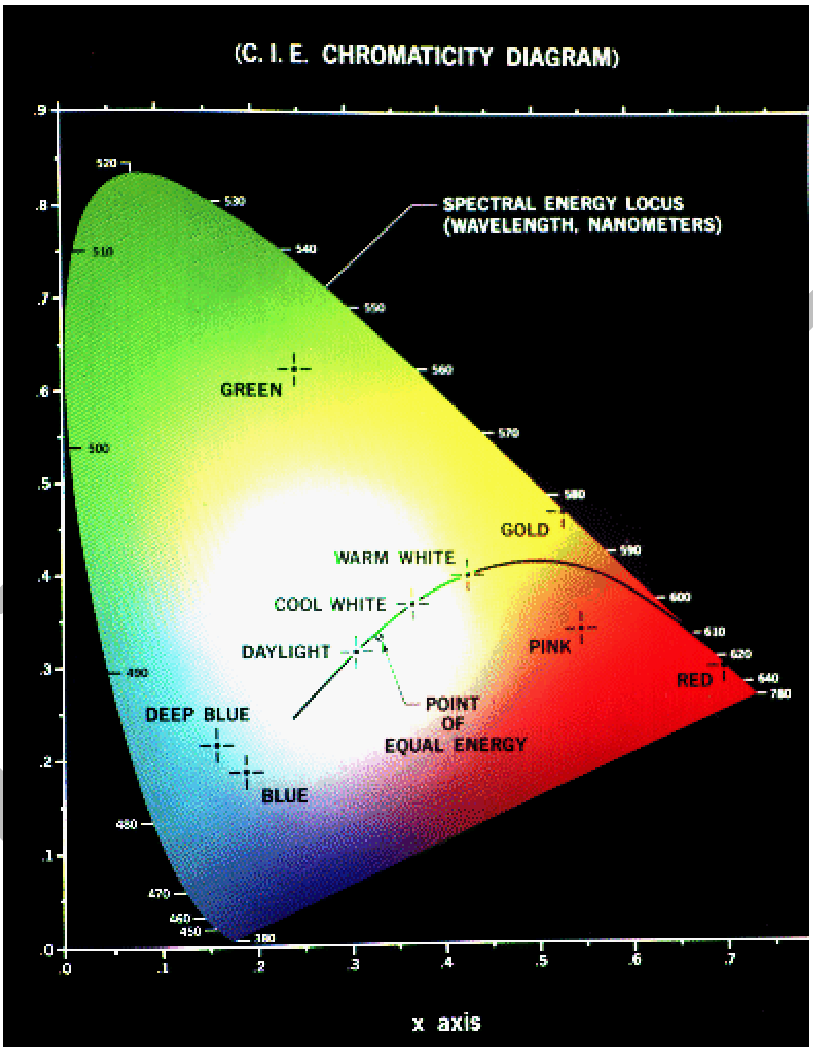


Figure 5.11: The CIE Chromaticity Diagram showing all visible colors. x and y are the normalized amounts of the X and Y primaries present, and hence z = 1 - x - y gives the amount of the Z primary required.

(Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

Both visible colors appear in the CIE Chromaticity Diagram (see Figure 5.11). The x and y axis provide normal X and Y primary quantities for a certain color and, therefore, z = 1 – x – y provides the necessary amount of Z primary. Chromaticity is independent of the luminous energy and depends on dominant wavelength and saturation. Colors of the same chromaticity but differing luminance all map in that area to the same point.

The pure spectrum colors lie on the curved border and a regular white light is determined to be close to equivalent energy point x = y = z = 1/3. The endpoints of a line at this point are followed by additional colors, that is, colors that add white. Both colors in any line of the chromaticity diagram can be achieved by combining the colors of the end points of the line as shown in Figure 5.12. In addition, the vertices will form all colors in a triangle by mixing colors.

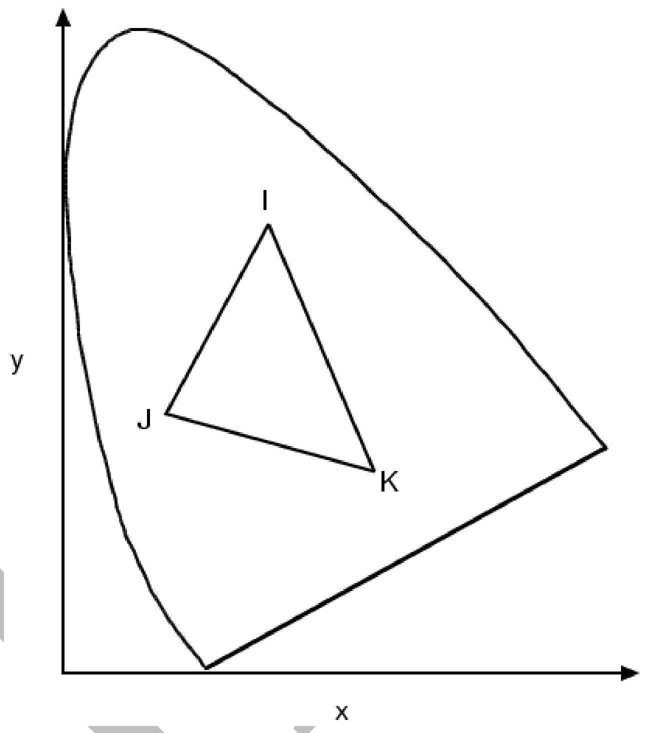


Figure 5.12: Mixing colors on the chromaticity diagram.

(Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

All colors on the line IJ can be obtained by mixing colors I and J, and all colors in the triangle IJK can be obtained by mixing colors I, J and K.

7.2 Color Models

By specifying a 3D coordinate system and a subspace that includes all constructible colors within a particular model, color patterns provide a standard way to specify a particular color. Every color that can be described by a model is a single point within the subspace it defines. The basic hardware (RGB, CMY, and YIQ) or image processing applications (HSI) are oriented to each color model.

The RGB Model

In the RGB model, an image is composed of 3 independent image levels: red, green and blue. One is in each of the primary colors. (As seen in figure 2, the regular wavelengths are for the three primaries). The sum of each primary component present is specified in a particular color. The geometry for color defined with a Cartesian coordination system for RGB color model is shown in Figure 5.13. In the line between the black and white vertices is the greyscale spectrum, i.e. these colors made of equal quantities of each primary.

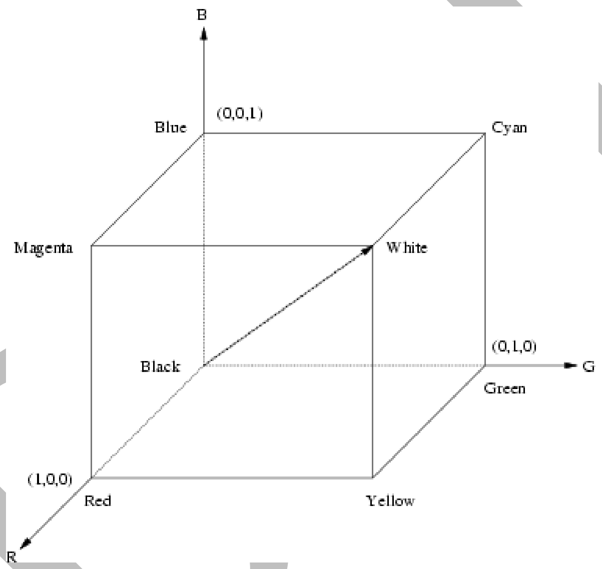


Figure 5.13: The RGB colour cube. The greyscale spectrum lies on the line joining the black and white vertices.

(Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

It is an additive model, which is to say that the colors present in the light add to new colors and is suitable, for example, for mixing colored light. The additive color mix of red, green, and blue primaries to form three secondary colors yellow (red + green), cyan (blue + green) and magenta (red + blue) and white ((red + green + blue) in the picture on the left of Figure 5.13.

For the most color displays and video cameras the RGB model is used.

The pixel depth is termed the number of bits used to display each pixel in RGB space. Consider an RGB image, which contains 8-bit images of each red, green, and blue image. Each pixel RGB [that is, triplets of values (R, G, B)] is 24 bits in depth under these situations (3 image planes times the number of bits per plane). A 24-bit RGB image is sometimes referred to by the term full-color image. The total number of possible colors in a 24-bit RGB image is $(2^8)^3$= 16, 777, 216 . Figure

8 illustrate the 24-bit RGB color cube analogous to the diagram in Figure 5.14 Note that the range of values in the cube are also scaled to the numbers that represent the number bits in the images for digital images. If the key images are 8-bit, the limits of the cube would be [0, 255]. For instance, at the point [255, 255, 255] white will be in the cube.

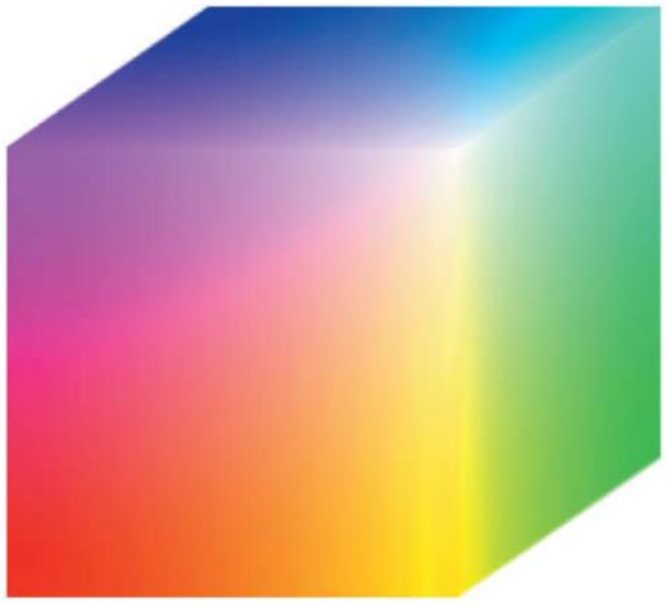


Figure 5.14 A 24-bit RGB color cube.

(Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

The CMY and CMYK Color Models

A subtractive model for absorption of the color, for example because of colored pigments in paints, is the CMY (cyan-magenta-yellow) model. While the RGB model determines what's added to black to obtain a particular color, the CMY model determines what is subtracted from white. The primaries are cyan, magenta and yellow in this case, the secondary color being red, green and blue.

When the surface is illuminated with a cyan pigment with white light, no red light, including magenta, yellow and blue, is reflected. The RGB and CMY model relationship is defined by:

$$\begin{bmatrix} C \\ M \\ Y \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} R \\ G \\ B \end{bmatrix} \text{ eq-1}$$

Most devices, such as color printers and copiers, that store colored pigments on paper need CMY data input or make an RGB internal conversion to CMY. The conversion is carried out with simple operation (1) where all color values are assumed to be standardized into the [0, 1] range. Equation

(1) shows that the light reflected by the pure cyan surface does not contain any red (i.e., C = 1 — R in the equation). Likewise, pure magenta is not green and pure yellow is not blue. Equation (1) also shows that a collection of CMY values allows RGB values to be easily obtained by deducting from 1. Per CMY value. As previously stated, this color model is used for the generation of a hardcopy output while image processing, so that it is usually of little practical interest to reverse CMY to RGB operation.

The CMY model is used by printing devices and filters.

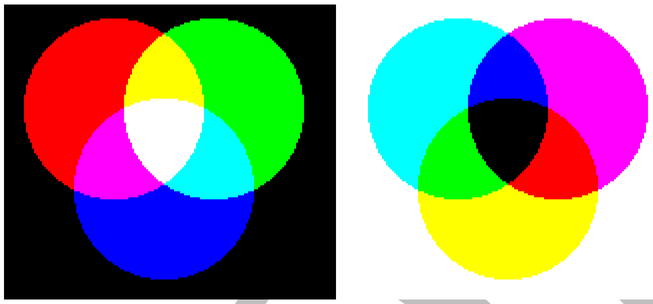


Figure 5.15: The figure on the left shows the additive mixing of red, green and blue primaries to form the three secondary colors yellow (red + green), cyan (blue + green) and magenta (red + blue), and white ((red + green + blue). The figure on the right shows the three subtractive primaries, and their pairwise combinations to form red, green and blue, and finally black by subtracting all three primaries from white.

(Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

The same quantities can contain black in the pigments primaries, cyan, magenta, and purple. The combination of these colors in practice makes a black look muddy.

Thus, a fourth color, black, denoted by the K, will be added to create true black (which then in print is the dominant color) which will lead to the color of the CMYK model. The black is applied to the measurements needed for true black. So when publishers speak of 'four-color printing,' they refer to the 3 CMY colors, plus a slice of black.

To convert from CMY to CMYK, use the following formula:

$$K = \min(C, M, Y) \quad \text{eq-2}$$

$$C = C\ K$$

$$M = M\ K$$

$$Y = Y - K$$

If then we have pure black, with no color contributions, from which it follows that

C=0 eq-3

M=0 eq-4

Y=0 eq-5

The C, M, and Y on the right side of Eqs. (2) -(5) are in the CMY color system. The C, M, and Y on the left of Eqs. (6) -(9) are in the CMYK system.

Otherwise

C = (C − K)/(1 − K) eq-6

M = (M − K)/(1 − K) eq-7

Y=(Y-K)/(1-K) eq-9

Where all values are assumed to be in the range [0, 1]. The conversions from CMYK back to CMY are:

C = C * (1 − K) + K eq-10

M = M * (1 − K) + K eq-11

Y = Y * (1 − K) + K eq-12

The C, M, Y, and K on the right side of Eqs. (10) - (12) are in the CMYK color system. The C, M, and Y on the left of these equations are in the CMY system

The HSI Color Model

The HSI model would improve the RGB model. The color model of the hue saturation intensity is closely similar to the color sensing properties of human vision. The method, which transforms from RGB to HSI or back, is harder than other color models. I denote the intensity of light, H refers to the hue indicating the measure of the purity of colors, S refers to the saturation. If the saturation value of a color is high, it indicates that the color is the low white color.

Three quantities of hue, saturation and intensity may be defined for color. This is the HSI model and the whole color space that can be specified is shown in **Figure 5.16**

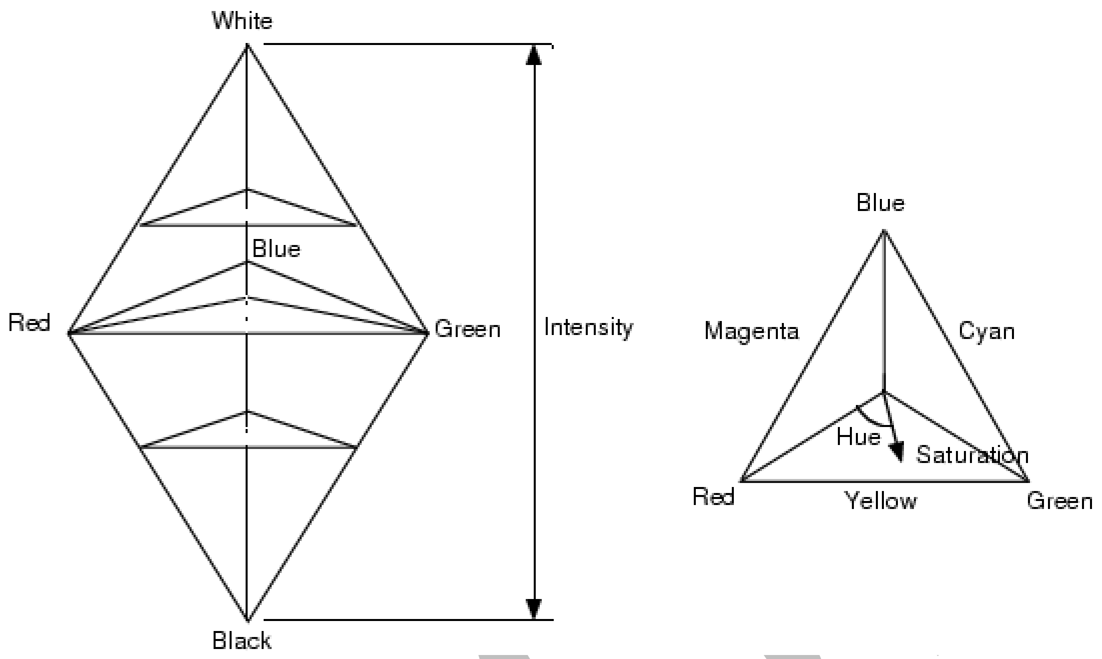


Figure 5.16: The HSI model with the left HSI, with the right HSI triangle, formed by a horizontal slice with a particular intensity through the HSI solid. The color is measured by a red, and the distance from the axis is saturated. The colors, i.e. pure colors, on the solid surface are fully saturated, and the grey spectrum on the solid axis. Hue is undefined for these colors.

(Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

Conversion between the RGB model and the HSI model is quite complicated. The intensity is given by

$$I = \frac{R+G+B}{3}$$

where the quantities R, G and B are the amounts, normalized to the range [0,1] in the red, green and blue components. Therefore the intensity is only an average of the components red, green and blue. The saturation is provided through:

$$S = 1 - \frac{\min(R,G,B)}{I} = 1 - \frac{3}{R+G+B}\min(R,G,B)$$

where the min(R,G,B) term is really just indicating the amount of white present. If any of R, G or B are zero, there is no white and we have a pure color.

Certainly this quantity is easy to measure and interpret. The color model of HSI (hue, saturation, intensity) separates the intensity component into a chromatic image from the data that conveys colors (hue and saturation). The HSI model is therefore an ideal tool for developing image processing algorithms that are natural and intuitive to humans, based on color descriptions. The primary colors are divided by 120°. The secondary colors are 60° from the primaries, which means that the angle between secondaries is also 120°. Figure 10 shows the same hexagonal shape and an arbitrary color point (shown as a dot)

An angle from a reference point determines the hue of the point. In general a 0° angle of the red axis is 0 hue, and the hue from there increases in reverse. The vector length from origin to point is the saturation (the distance from the vertical axis). The origin is defined by the vertical intensity axis intersection of the color plan. High intensity axis, length of the /vector up to a color point and the angle this vector creates with the red axis are the important components of the HSI color spaces.

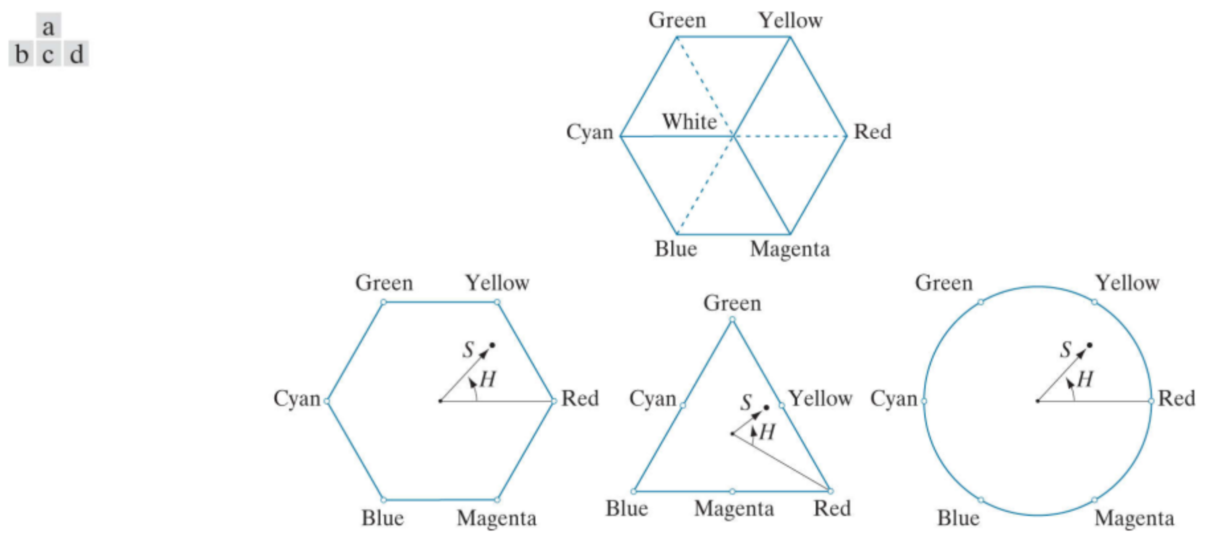


Figure 5.17 Hue and saturation in the HSI color model. The dot is any color point. The angle from the red axis gives the hue. The length of the vector is the saturation. The intensity of all colors in any of these planes is given by the position of the plane on the vertical intensity axis.

(Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

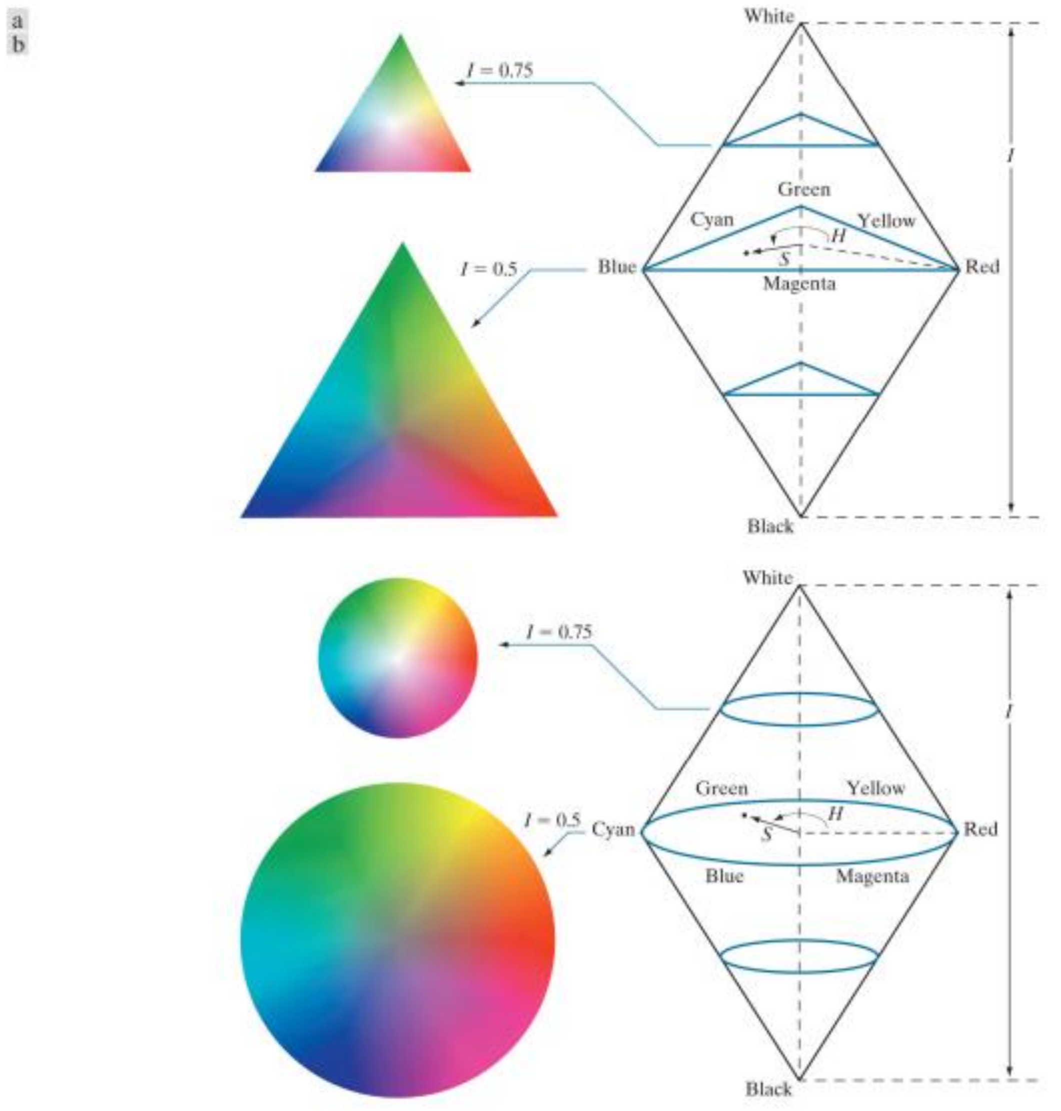


Figure 5.18 The HSI color model based on (a) triangular, and (b) circular color planes. The triangles and circles are perpendicular to the vertical intensity axis.

(Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

Converting colors from RGB to HIS

Consider RGB values normalized to the range [0; 1].

Given an RGB value, H is obtained as follows:

$$H = \begin{cases} \theta & if\ B \le G \\ 360 - \theta & if\ B > G \end{cases}$$

It should be normalized to the range [0; 1] by dividing the quantity computed above by 360

θ is given by

$$\theta = cos^{-1} \left\{ \frac{1/2[(R-G)+(R-B)]}{[(R-G)^2 \;\; + (R-B)(G-B)]^2]} \right\}$$

θ is measured with respect to red axis of HSI space

Saturation is given by

$$S = 1 - \frac{3}{R+G+B} \; [\min{(R, G, B)}]$$

Intensity component is given by

$$I = \frac{R+G+B}{3}$$

Converting colors from HSI to RGB

Consider the values of HSI in the interval [0; 1]

H should be multiplied by 360 (or 2π) to recover the angle; further computation is based on the value of H

RG sector – 0◦ ≤ H < 120◦

$$B = I(1-S)$$

$$R = I\left[1 + \frac{S \; cos \; H}{cos(60^0 \; - H)}\right]$$

$$G = 3I - (R+B)$$

GB sector – 120◦ ≤ H < 240◦

$$H' \;\; = H - 120^0$$

$$R = I(1-S)$$

$$G = I\left[1 + \frac{S \; cos \; H'}{cos(60^0 \; - H')}\right]$$

$$B = 3I - (R+G)$$

BR sector – 0◦ ≤ H < 360◦

$$H' \;\; = H - 240^0$$

$$G = I(1 - S)$$

$$B = I\left[1 + \frac{S \cos H'}{\cos(60^0 - H')}\right]$$

$$R = 3I - (G + B)$$

A Device Independent Color Model

Color model CMYK and RGB are device-dependent, that is to say, on the way a color is translated. They point to a particular device, but don't have information about the final person's perception of the color. The color with the identical RGB parameters is viewed by us in various ways based on the brightness contrast and the sharpness of your computer monitor, ambient light and the angle we are looking at the monitors. In a 'CMYK' color model, a person's perception of color depends on an even higher variety of conditions (for example glossy paper is more vivid and rich, than a matte color), particularly paint, the humidity of the papers dried up and the attributes of the printed printing press, etc.

We append (such as) color profiles to hardware-reliant color patterns to transmit extra dependable color information to a person. So each profile includes information about a specific way of human color transmission and adapts the last color to the original color settings by adding or removing any constituents. For instance, a colored profile is used for printing on glossy foils, 10% Cyan cleaning and 5% Yellow adding to the original color, because of the unique attributes of the printing press, the film itself and other conditions. However, not all the transfer color problems can be solved, even attached profiles.

Device-independent color models have no information on a person's color transfer. They describe color that people with normal colored vision perceive mathematically.

A device independent color space is one in which the coordinates used to define the color, wherever they are applied, produce the same color. The color space CIE L*a*b* is an example of a device-dependent color space (known as CIELAB and based on the human visual system).

The color components are given by the following equations:

$$L* = 116.h\left(\frac{Y}{Y_w}\right) - 16$$

$$\alpha * = 500\left[h\left(\frac{X}{X_w}\right) - h\left(\frac{Y}{Y_w}\right)\right]$$

and

$$b * = 200\left[h\left(\frac{Y}{Y_w}\right) - h\left(\frac{Z}{Z_w}\right)\right]$$

where

$$h(q) = \begin{cases} \sqrt[3]{q} & q > 0.008856. \\ 7.787q + \frac{16}{116} & q \leq 0.008856 \end{cases}$$

The reference white tri stimulus values of Xw, YW and Zw are usual white of a diffuser reflective of CIE D65 illumination standard (defined by x = 0.3127 and y = 0.3290 in CIE chromaticity diagram of Figure 5.11). Figure 5.11: The colors of the L*a*b are colorimetric (i.e. colors that match are identically encoded), are perceptually uniform (i.e. the color differences among various colors are uniformly perceived - see Mac Adams class paper [1942]). Although the format cannot be viewed directly (it is required to convert to another color space), its spectrum covers the whole spectrum of viewers and can precisely display, print or print, or input device. Like the L*a*b system, it is an excellent intensity decoupled (presented in lightness L*) from the color (presented with a* for red minus green and b* for green minus blue), which makes it useful both for the manipulation of images (tone and contrast editing) and for image compression application. Calibrated imagery systems benefit primarily from interactive, independent correction of tonal and color imbalances in two sequence operations. Before irregularities such as colors over and under saturated, the problems of the tonal range of the images are resolved. The tonal range of an image, also known as its key type concerns its overall color intensity distribution. Most information on high-key images are concentrated at high (or light) intensities; colors for low-key images are mainly at low intensities; middle-key images lie among them. As in the monochrome case, the intensities or the image of color between the highlights and the shadows are often equally desirable. The following examples show a range of color transformations for tonal and color balance correction.

5.13 Pseudocolor Image Processing

Pseudo color processing (also known as false color means assigning colors to grey values, on the basis of a given criterion. The name pseudo or false color is applied to distinguish the color assignment process from the process associated with true color for human visualization and interpretation of gray-scale events in an image or image sequence. The fact that human beings distinguish thousands of colors shades and intensities from only two dozen or so shades of the grey is one of the main reasons for coloring.

Intensity Slicing and Color Coding

This is a straightforward case for pseudocolor image processing. It is also known as color coding or density slicing.

Imagine a 3D function greyscale image with the intensity of the third dimension. The pixel position coordinates would "slice" the image into two parts when placing a plane parallel to the horizontal plane. Then different colors can be assigned to different levels. Figure 5.8 shows an example of using a plane at f(x, y) = l_i to slice the image function into two levels.

If each side of the plane shown in Figure 5.19 is assigned a different color, a pixel whose grey level is over the plane is coded in one color, whilst any pixel underneath it is coded in one color. One of the two colors may be assigned arbitrarily to the levels on the plane itself. The result is a two-colored picture that can control its relative appearance by moving the sliding plane upward and downward.

The technique can be formulated as continues to follow in particular. The grey scale is [0, L-1], and let lo is indicate as black [f(x,y),y) = 0], and level l $_{L-1}$the represented white is [f(x,y) =L-1]. Assume that P planes are defined in levels l_1, l_2,....,l_p are perpendicular to the intensity axis. If 0 < P < L –1, the grey scale is divided into I_1, I_2... I_{p+1}.

$$iff(x,y)\epsilon I_k \ letf(x,y) = c_k$$

Where c_k is the color accompanying with the kth intensity interval I_k defined by the partitioning planes at l = k - 1 and l = k.

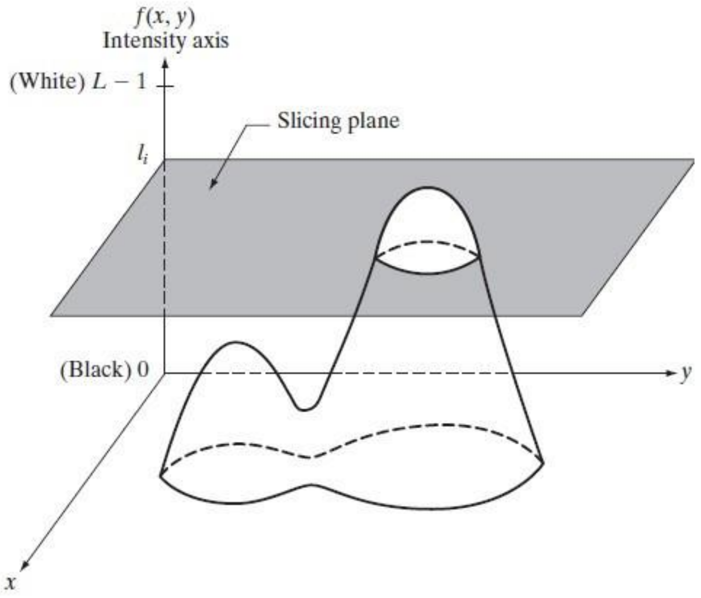


Figure 5.19 Geometric interpretation of the intensity slicing technique.

(Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

The concept of planes is particularly useful to interpret the intensity-slicing technique in geometric terms. Figure displays an alternative image defining the same mapping as in Fig. Depending on whether the mapping function in the figure is above or below the li value, any grey input level is assigned one of two colors. The mapping function takes on a staircase form when more levels are used.

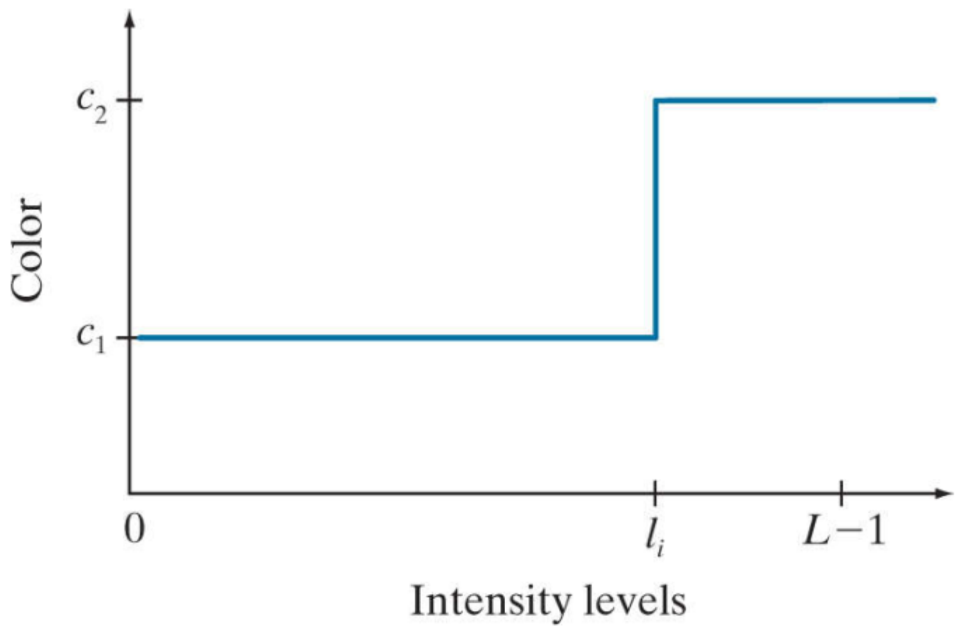


Figure 5.20 An alternative representation of the intensity-slicing technique.

(Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

Figure shows a simple yet practical application of intensity slicing. The grayscale image of the radiation testing pattern Picker Thyroid Phantom is **Figure 5.21** (a) and the intensity slicing of this image is divided into eight colors, as shown in **Figure 5.21** (b). Regions that occur in the grayscale are in fact very variable, as the different colors in the sliced image show.

The left lobe is a dull grey in the grayscale picture, for example, which makes it difficult to pick out changes in intensity. In contrast, 8 regions of constant intensity are clearly visible in the picture, one for each of the colors used. With different color numbers and intensity intervals, the characteristics of intensity variations in a grayscale image are quickly determined. In situations such as the one shown here, the object of interest has a uniform texture with variations in intensity which are difficult to analyze visually.

a b

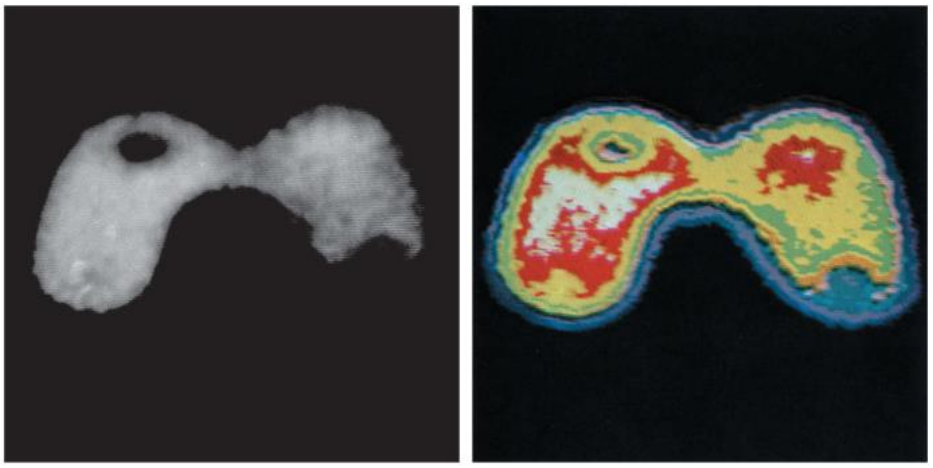


Figure 5.21 (a) Grayscale image of the Picker Thyroid Phantom. (b) Result of intensity slicing using eight colors.

(Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

The above simple example divided the grayscale into intervals and each was given a different color, regardless of the significance of the grayscale. In this case, the interest was simply to view the various grey levels that make up the image. Intensity slicing plays a much more meaningful role when the grayscale is divided according to the physical characteristics of the image Figure 5.22.(a) shows, for example, the X-ray image of a weld (wide dark horizontal area) with multiple cracks and porosities (the bright streaks running horizontally through the middle of the image). If porosity or crack is present in a weld, the full force of X-rays through the object saturates the image sensor across the object. Thus, an 8-bit image of intensity values of 255 in such a system implies a welding problem automatically. If visual analysis is used to inspect welds (a widely accepted method still now), the inspector can make the work significantly easier with the simple color coding which allocates one color to level 255 or another to all other intensity levels. The result is displayed in Figure 5.22 (b). The conclusion that if images were presented in the form of Figure 5.22.(b) no explanation was required that human error rates would be lower in other words, if you are looking for an intensity or range of values, intensity slicing is a simple but effective visualization aid, especially if many images need to be checked regularly.

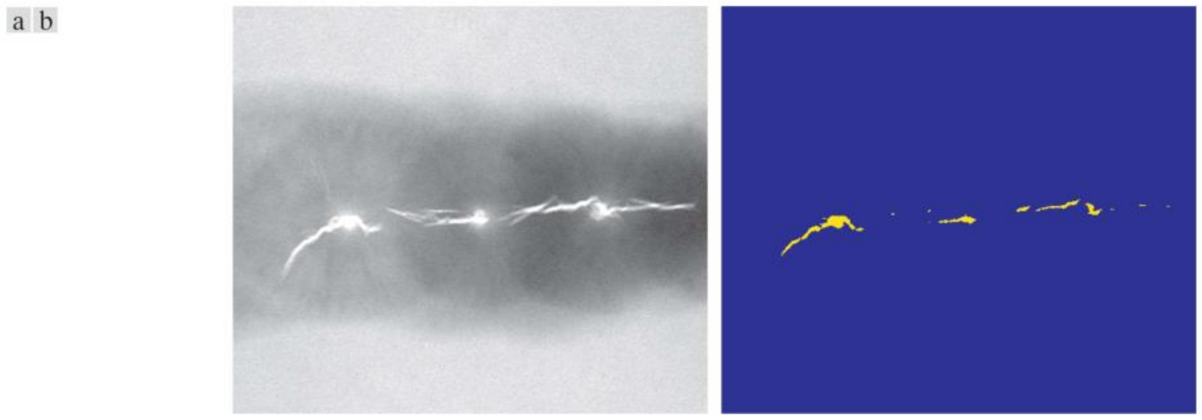


Figure 5.22 (a) X-ray image of a weld. (b) Result of color coding.

(Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

Intensity to Color Transformations

The idea behind this approach is to carry out three independent grey pixel transformations of any input pixel. The three outcomes are then separately fed into a color Television screen's red, green and blue channels. This technique generates a composite image whose color contents are amplified by the nature of the transformations functions. Mention that the gray-level image values are transformations and are not position functions.

In the intensity slicing, piecewise linear functions of the grey level generate linear functions to produce colors. This method can, however, be focused on smooth, nonlinear functions, which give significant versatility as might be expected.

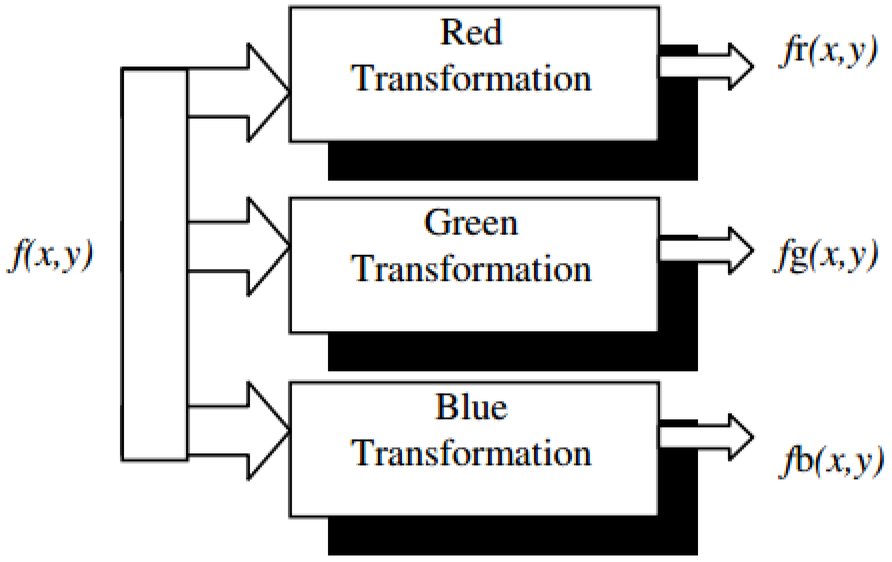


Figure 5.23 Functional block diagram for pseudocolor f_R ,f_G and f_B image processing. Images and are fed into the corresponding red, green, and blue inputs of an RGB color monitor.

The type of processing shown just now is very powerful for visualizing events of interest for complex images, in particular when they go beyond our normal sensing capabilities. **Figure 5.24** is a very good example. These are Jupiter Moon Io images, shown in pseudo-color, when several sensor photos of the Galileo spacecraft are combined, some of which cannot be seen on the eye in spectral regions. However, it is possible to combine the sensed image with a meaningful pseudo-color map by understanding physical and chemical processes that can influence sensors' responses.

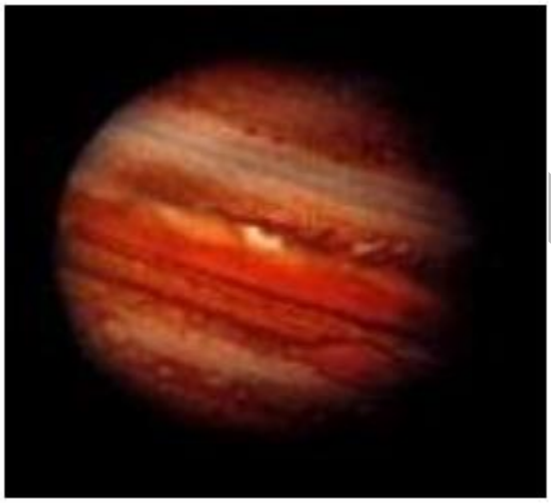


Figure 5.24: pseudo color rendition Jupiter Moon Io.

(Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

One way of combining sensed image data is by showing differences in the composition of surface chemicals or by modifying the reflected surface sunlight. The pseudo color image, for example, in **Figure 5.25** shows bright red representing the newly expelled material from Io's active volcano. This image provides these features much more easily than would have been possible through individual analysis of the component images

Figure 5.25: A close-up (Courtesy of NASA)

(Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods

5.14 Basics of Full-Color Image Processing

Techniques for full color image processing fall under two categories. In the first category, each component image is constructed independently and a composite color image from the individual elements is then established. One work directly with color pixels in the second category. Due to the at least three color images, color pixels are actually vectors. In the RGB system for example, every color point could be interpreted as a vector in the RGB coordinate system extending from the source to that point.

Let c represent an arbitrary vector in RGB color space:

$$c = \begin{bmatrix} C_R \\ C_G \\ C_B \end{bmatrix} = \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

This equation shows that the components of c are at one point simply the RGB components of a color image. When the color components are a coordinate function (x, y), the notation is used

$$c = \begin{bmatrix} C_R(x,y) \\ C_G(x,y) \\ C_B(x,y) \end{bmatrix} = \begin{bmatrix} R(x,y) \\ G(x,y) \\ B(x,y) \end{bmatrix}$$

MN vectors such as c(x, y), x = 0,1, 1, 2,...,M- l; y = 0,1,2,,...,N- 1, are present on an image of the size M X N.

It should be kept in mind clearly that Eq.(2) represents a vector with spatial variables of x and y components. Two conditions have to be met for the equivalence of each color component and vector-based processing: First, both vectors and scalars must be covered by this process. Second, it must be independent of all other components for each component of the vector.

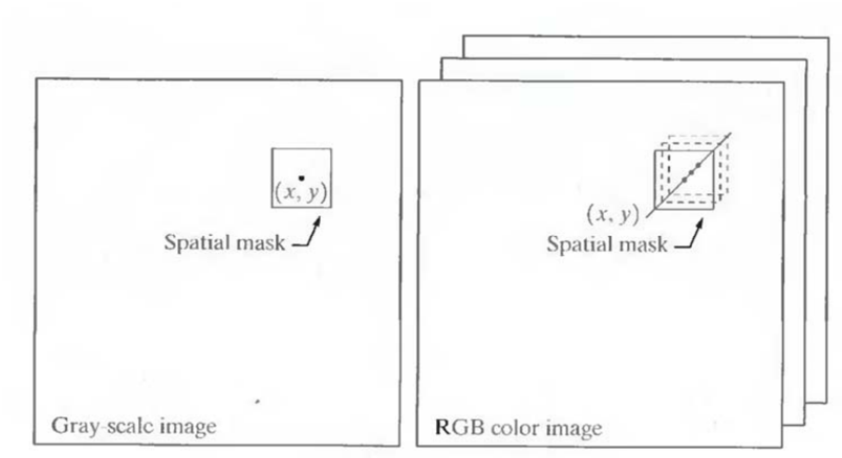


Figure 5.26 Spatial masks for gray-scale and RGB color images.

(Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

Figure 5.26 shows grey scale and full-color images for the spatial processing in the neighborhood. Assume the process is an average neighborhood. The average grey level of all pixels in the neighborhood would be summarized and divided by the total number of pixels in the neighborhood in **Figure 5.26** (a). In **Figure 5.26 (**b), an average of all vectors in the neighborhood would be summed up and the total number of vectors in the neighborhood would be divided by each component. So each component of the average pixel is the sum of the image pixel of the same component, which is the same as the result if the average is per color component.

5.15 Color Transformations

In the context of a single color model, the color transformation processes the color image components.

Formulation

As with the techniques of transformation at gray-level of model colour with the expression

$$g(x, y) = T[f(x, y)]$$

Where f(x,y) is an color input image, g(x,y) is a color output image that's transformed or processed, and T has become an operator in f over a space area of f (x,y).

The pixel values here are three-fold or quartets from the color space selected to represent the images (e. g. groups of three or four values).

We introduced the basic grayscale transformations similarly to the approach. In this section we will focus only on the color changes in the form.

$$si = T(r1, r2, \ldots\ldots, rn), i = 1,2, \ldots\ldots\ldots, n$$

Where ri and si are variables for notation simplicity which are f(x,y) and g (x,y) at any point (x,y) color components, n is the number of color components, and {T1,T1,....,Tn} is a set of functions for transformation and color map that operates at the time of ri to produce si. Notice that n transformations, Ti, integrate in equation to implement the single transformation function T . The color space selected to describe f and g pixels defines the value of n. The red, green, and blue components for the image input respectively indicate when the RGB color space is selected, e.g. n=3, r1,r2, and r3. When selecting CMYK or its color spaces, n=4 or n=3.

Figure above shows a color image of a large-size (4' x 5 ') high-resolution color image of a bowl of strawberries and cup of coffee that's been digitized. The initial CMYK scan components are in the second row of the figure. In these images, 0 is black and 1 color component in each CMYK. This is shown by the strawberries, because the images that correspond to the two CMYK components are the brightest of the large quantities of magenta and yellow. Black is spare and generally contained in the bowl of strawberries in coffee and shadows. The last row or Fig. 7 shows the components of **Figure 5.27**— equated, (2) (4). As expected, a monochrome rendering of the original color is the intensity component. Figure 7: components computed using equation, (2) up to (4)are shown in this last row . The intensity component is presumed to be a monochrome version of the original full-colour. Moreover the strawberries are relatively pure in colour; they have the highest saturation or a minimum dilution of the hues in the image by white light, so finally we find it difficult to describe the hue component.

The challenge is exacerbated by the fact that in a model where 0 and 360o meet, there is a discontinuity and that the saturation of the hue is uncertain (— for example for white, black, and pure grey). The discontinuity is quite obvious around the strawberries, which are displayed in both black (0) and white grey level values (1).

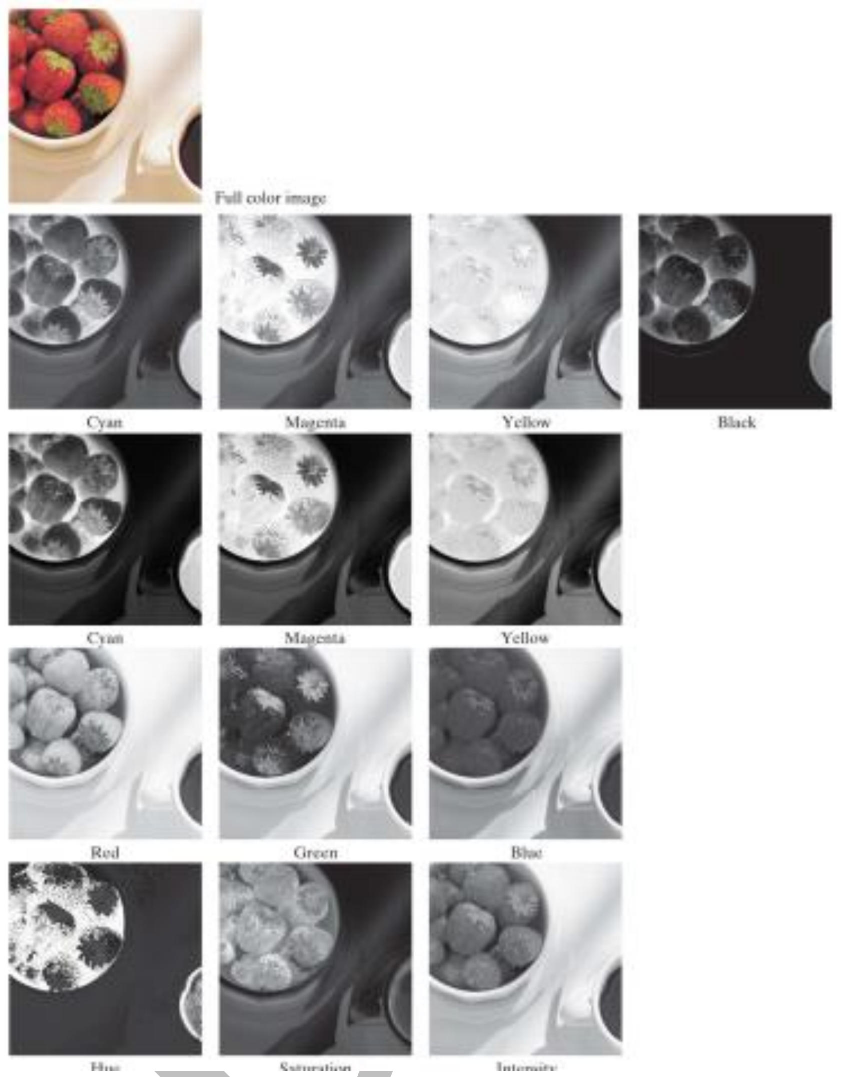


Figure 5.27: A full-color image and its various color-space components (Original image courtesy Med-data Interactive.)

(Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

Equation (2) can be used with any colour space components of **Figure 5.27** (2). Theory shows that any changes in any colour model can be carried out. However, a few other operations are more suitable for specific models in practice. The cost of transferring representations into a decision on the colour space to implement this must be considered for a given transformation. Suppose, for example, we want to change the image intensity using Fig. 7.

$$g(x, y) = kf(x, y)$$

Where $0 < k < 1$, the simple transformation can be done in the colour space.

$$s_3 = kr_3$$

Where s1 = r1, and s2 = r2. Only its component intensity r3 is altered. Three components have to be changed in the RGB colour space:

$$s_i = kr_i \quad i = 1,2,3$$

Similar linear transformations are required in CMY space:

$$s_i = kr_i + (1-k) \quad i = 1,2,3$$

In the same way, the transformations needed to change the CMYK image intensity are given

$$s_i = \begin{cases} r_i & i = 1,2,3 \\ kr_i + (1-k) & i = 4 \end{cases}$$

This equation tells us that we only modify the fourth (K) component to modify the intensity of a CMYK image.

Figure 5.28 shows that the transformations are applied to the full color image . In **Figure 5.28** (c) through (h) the mapping functions are shown graphically. The CMYK mapping function consists of two components, the same is the case with HSI; one component is handled by the transformations, the other by the rest. As most various transformations have been made, the net outcome of modifying the color intensity by a fixed value for everyone has been the identical

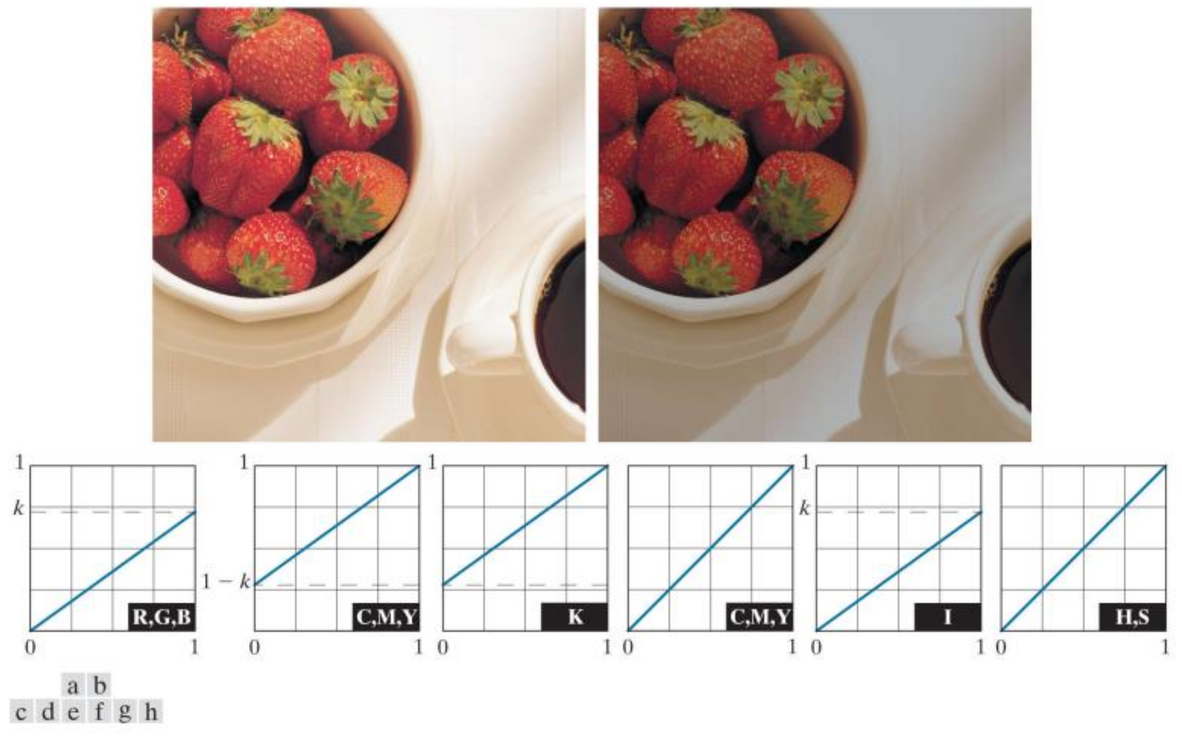


Figure 5.28 Adjusting the intensity of an image using color transformations. (a) Original image. (b) Result of decreasing its intensity by 30% (i.e., letting). (c) The required RGB mapping function. (d)–(e) The required CMYK mapping functions. (f) The required CMY mapping function. (g)–(h) The required HSI mapping functions.

(Original image courtesy of MedData Interactive.)

(Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

It must be noted that every transformation represented throughou) varies depending on only one component within the space of its colors. The red outcome component, varies depending on only the red input and is autonomous of the green and blue inputs. Straightforward and regularly used tools for color processing include transformations of this type. As mentioned at the start of our discussion, it can be done on a per-color basis.

We would also evaluate numerous transformations of this type in the remainder of this section and discuss the matter wherein the functions for component transformation depend on all the color components of the input image and therefore cannot be performed on a single color component basis.

Color Complements

The colour circle (also known as the colour wheel) shown in **Figure 5.29** came from Sir Isaac Newton who developed his first position at the end of the colour spectrum in the seventeenth century. The colour circle is an image of the colours organized as per their chromatic connection. The circle consists of the primary colours being equally distant. The secondary colours are then positioned in an equal distance arrangement between both the primary colours. The net outcome is that the colour circle is supplemented by hues directly opposite one another. Our interest in supplements is due to their analogy with the grayscale negative.

Color complements are, as with the gray-scale particular instance, effective to enhance the detail embedded in the dark regions of a color image notably if the regions dominate. Some of these concepts are explained in the following example.

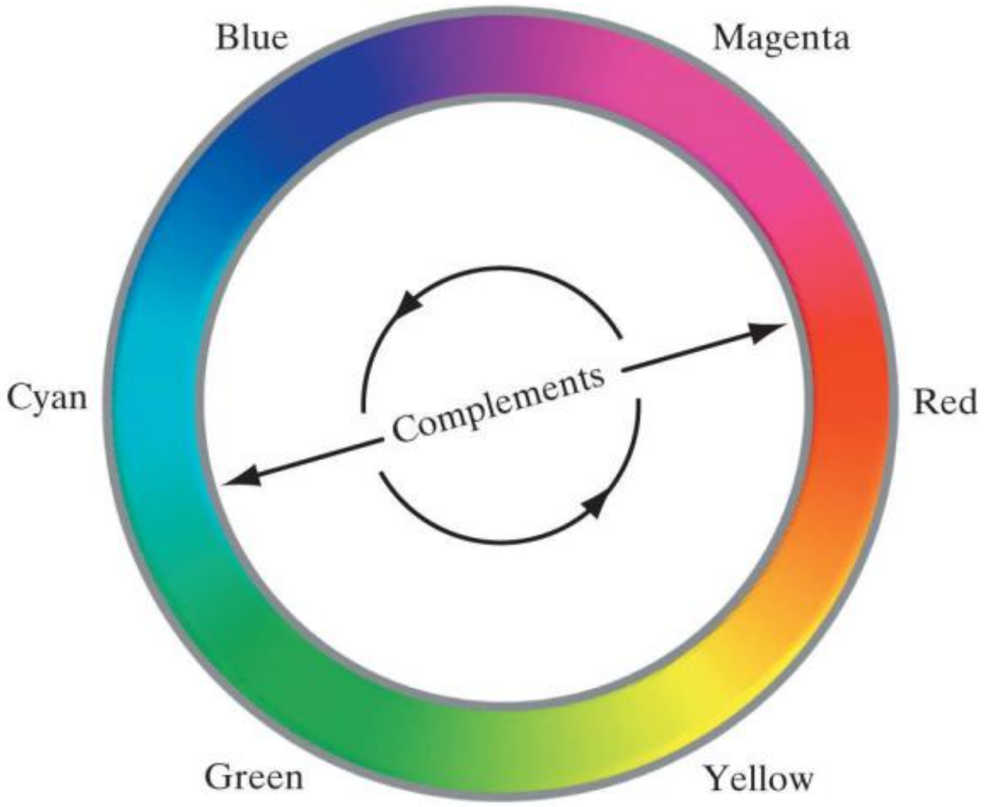


Figure 5.29 Color complements on the color circle.

(Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

EXAMPLE: Computing color image complements.

The full picture of **Figure 5.30** and its complement to colour is displayed in **Figure 5.30** (a) and (c). **Figure 5.30** (b) illustrates the RGB transformations used to calculate a complement. They correspond to the negative grayscale process. The complement recalls conventional colour film negative photographic elements. The cyans in the complement replace the reds of the original image. The compliment is white when the original image is black, etc. The colour circle of **Figure 5.30**. allows the original image to be predicted for each hue in the supplement image, and only a corresponding input colour functions for each RGB component that is involved in the calculation of a complement.

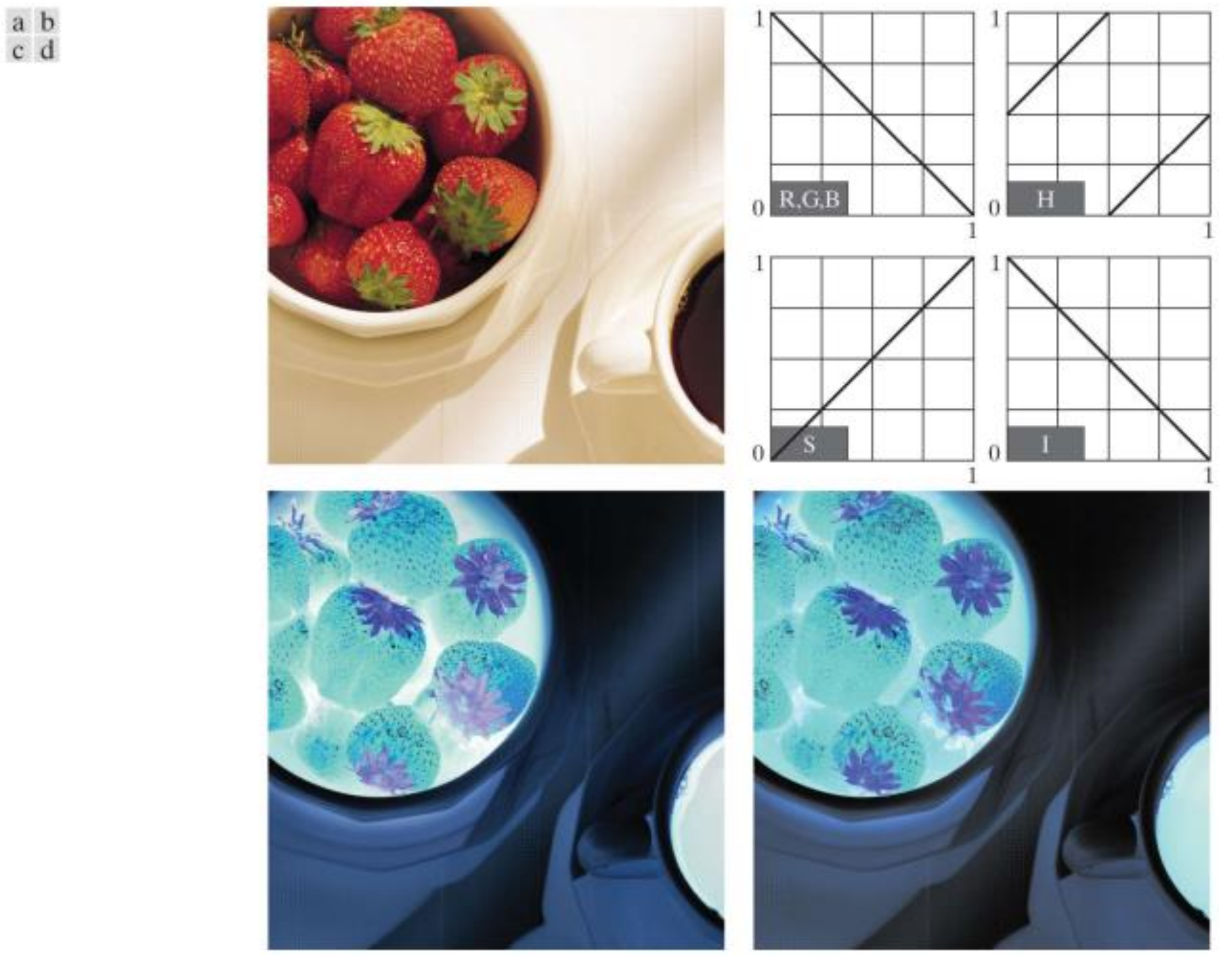


Figure 5.30 Color complement transformations. (a) Original image. (b) Complement transformation functions. (c) Complement of (a) based on the RGB mapping functions. (d) An approximation of the RGB complement using HSI transformations.

(Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Wood

Color Slicing

It is useful to highlight a special range of colours in an image to separate objects from their environment. The basic idea: (1) demonstrates the colours of interest so they can distinguish themselves from the background; or (2) uses a color-defined area as a mask. The simplest approach is to extend the techniques of intensity slicing. Even so, since the colour pixel is an n-dimensional quantity, the consequent colour transformation functions are more difficult than their counterparts in grey scale in figure 3.11. In essence, the necessary transformations seem to be quite complicated than that of the originally thought colour component transformations. It is because every practical colour slicing method includes the transformed colour components of every pixel as a function of all the colour components of the original pixel.

One way of "slicing" a color image is to map colors into a nonprominent neutral color after a certain range of interest. When a Cube of width W (or hypercube for) is contained in interest's colors, the required transformation set shall be determined through prototypes (e.g. average) of color with components.

$$s_i = \begin{cases} 0.5 \;\; if \left[\left|r_j - a_j\right| > \dfrac{W}{2}\right] & any\ 1 \le j \le n\ i = 1,2,\dots,n \\ r_i \qquad otherwise \end{cases}$$

Such transformations illustrate the colours surrounding the prototype by forcing all other colours to the centre point of the colour space (this is an arbitrarily chosen neutral point). For example, the colour space for the RGB is middle gray or colour (0.5, 0.5, 0.5) with an appropriate neutral point.

Eq. has become whenever a sphere is used to indicate the colours of interest.

$$s_i = \begin{cases} 0.5 \;\; if \displaystyle\sum_{j=1}^{n} (r_j - a_j)^2 > R_2^0 & i = 1,2,\dots,n \\ r_i \qquad otherwise \end{cases}$$

The enclosed sphere radius (or hypersphere for) and its centre is component (i.e., a prototypical colour). Other beneficial variations of Eqs. and include the implementation of multiple colour prototypes and a reduction of the intensity of colours, rather than a neutral constant.

5.16 Color Image Smoothing and Sharpening

It will be illustrated in this section how the fundamentals of this type of neighbourhood processing are applied to the task of smoothing and sharpening colored images.

Color Image Smoothing

Using the grayscale picture smoothing technique, one can view of spatial filtering as a spatial filtering operation in which the coefficients of the filtering mask are all 1. As the mask is moved across the image to be smoothed, each pixel is replaced by the average of the pixels in the neighborhood indicated by the mask, resulting in a smoothed image with fewer pixels. The application of this principle to the processing of full-color photographs is straightforward. The most significant distinction is that, rather than dealing with scalar gray-level values, we must deal with component vectors of the form given in Equation (2).

Assume Sxy is the set of coordinates that define a neighbourhood in an RGB colour image that is centred at (x, y). In an RGB colour image, The average of the RGB component vectors in this neighbourhood is calculated as follows:

$$\bar{\mathbf{c}}(x,y) = \frac{1}{K} \sum_{(s,t) \in S_{xy}} \mathbf{c}(s,t)$$

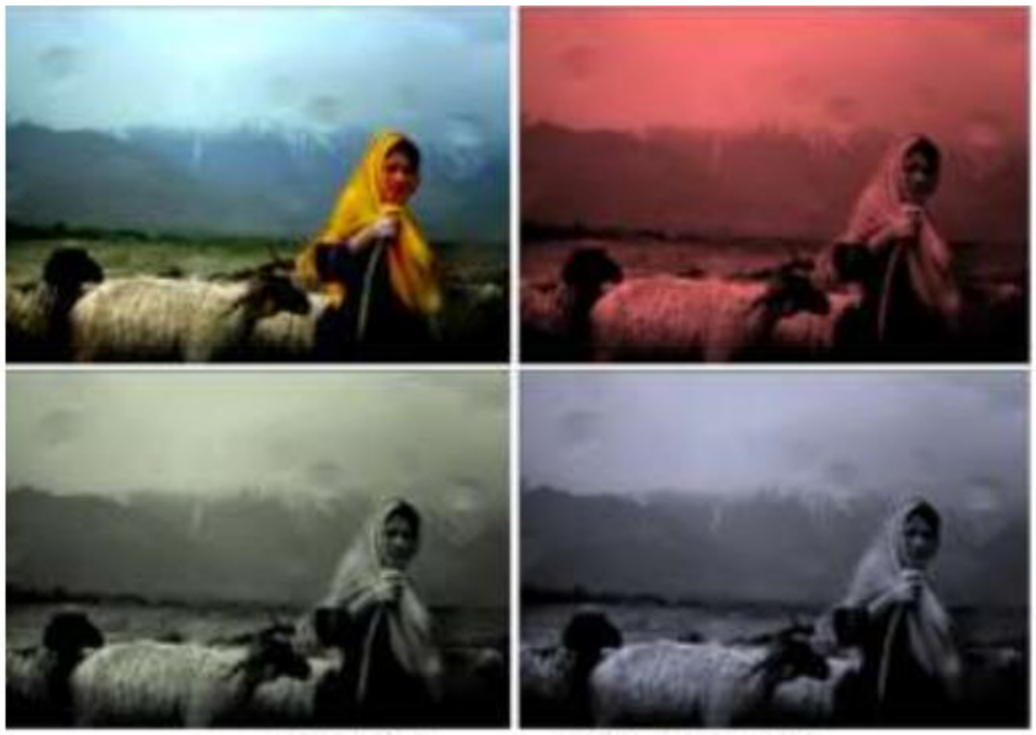
Equation (1)

It follows from Equation (2) and the properties of vector addition that

$$\bar{\mathbf{c}}(x,y) = \begin{bmatrix} \frac{1}{K} \sum_{(s,t) \in S_{xy}} R(s,t) \\ \frac{1}{K} \sum_{(s,t) \in S_{xy}} G(s,t) \\ \frac{1}{K} \sum_{(s,t) \in S_{xy}} B(s,t) \end{bmatrix}$$

Equation (2)

Identify the components of this vector as scalar images that would be created by independently smoothing each plane of the beginning RGB image using standard gray-scale neighbourhood processing, as shown in the example below. Consequently, we conclude that smoothing by neighbourhood averaging can be performed on a per-color plane basis using neighbourhood averaging. In this case, the result is identical to the one obtained when the averaging is conducted using RGB colour vectors.

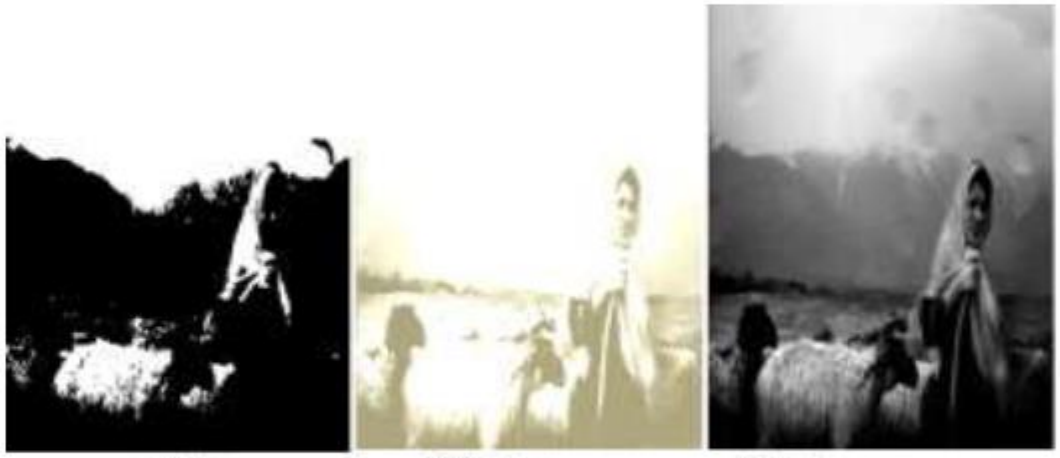


(a) RGB image (b) Red component image
(c) Green component (d) Blue component.

Figure 5.31 Color Image Smoothing

Eg:- Consider the color image shown in **Figure 5.31** (a). The red, green, and blue planes of this image are depicted in Figs, **Figure 5.31** (b) through (c)

The HSI components of the image are depicted in Figures **Figure 5.31** (a) through (d). We simply smooth each of the RGB colour planes on its own, and then merge the smoothed planes to generate a smoothed full-color result by combining the processed planes.



(a) Hue (b) Saturation (c) Intensity

Figure 5.32 HSI component of the RGB color image

Undoubtedly, the most significant advantage of the HSI colour model is that it decouples intensity (which is closely related to grey scale) from colour information. The fact that it is suited for various gray-scale processing techniques suggests that smoothing simply the intensity component of the HSI representation shown in **Figure 5.32** may be more efficient than smoothing the entire representation. This approach's benefits and/or drawbacks are demonstrated next by smoothing only the intensity component, while keeping the hue and saturation components unaffected, and converting the result to an RGB image for display.

Color Image Sharpening

In this section, we will look into image sharpening with the Laplacian function. Knowing the Laplacian of an input vector, we can define it as a vector with components that are identical to the Laplacian of each individual scalar component of the input vector. This definition comes from the field of vector analysis. The Laplacian of vector c in Equation I corresponds to the RGB colour scheme.

$$\nabla^2[\mathbf{c}(x,y)] = \begin{bmatrix} \nabla^2 R(x,y) \\ \nabla^2 G(x,y) \\ \nabla^2 B(x,y) \end{bmatrix}$$

In the same way that we learned in the previous section, we can compute the Laplacian of a full-color image by computing the Laplacian of each component image individually.

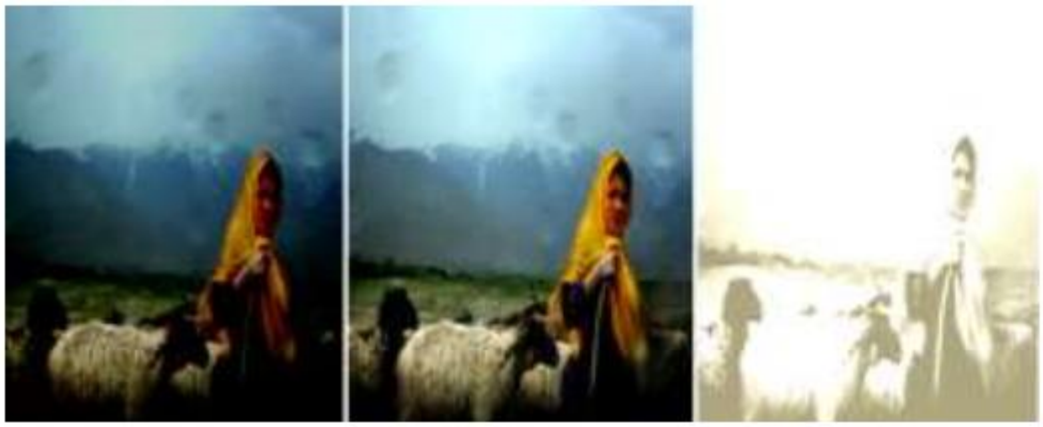


Figure 5.33 Image sharpening with the Laplacian.

Using Equation to compute the Laplacians of the RGB component pictures in **Figure 5.30** and merging them to generate the sharpened full-color result in **Figure 5.33**, .This result was obtained by multiplying the Laplacian of the intensity component by the hue and saturation components, which remained constant.

5.17 Using Color in Image Segmentation

Segmentation is a process that partitions an image into regions.

Segmentation in HIS color Space

If we want to segment an image depending on colour, for example, and in this case, Furthermore, we wish to carry out the procedure on an individual basis. When it comes to planes, it is reasonable to think of the HSI space because the hue picture provides an accessible representation of colour. Typically, using saturation as a masking image, you can separate off more information. In the hue image, look for areas of interest. The image of intensity is as follows: Because of this, it is employed less frequently for colour picture segmentation. It does not contain any colour information. The following is an illustration: This is an example of how segmentation is carried out in the HSI system.

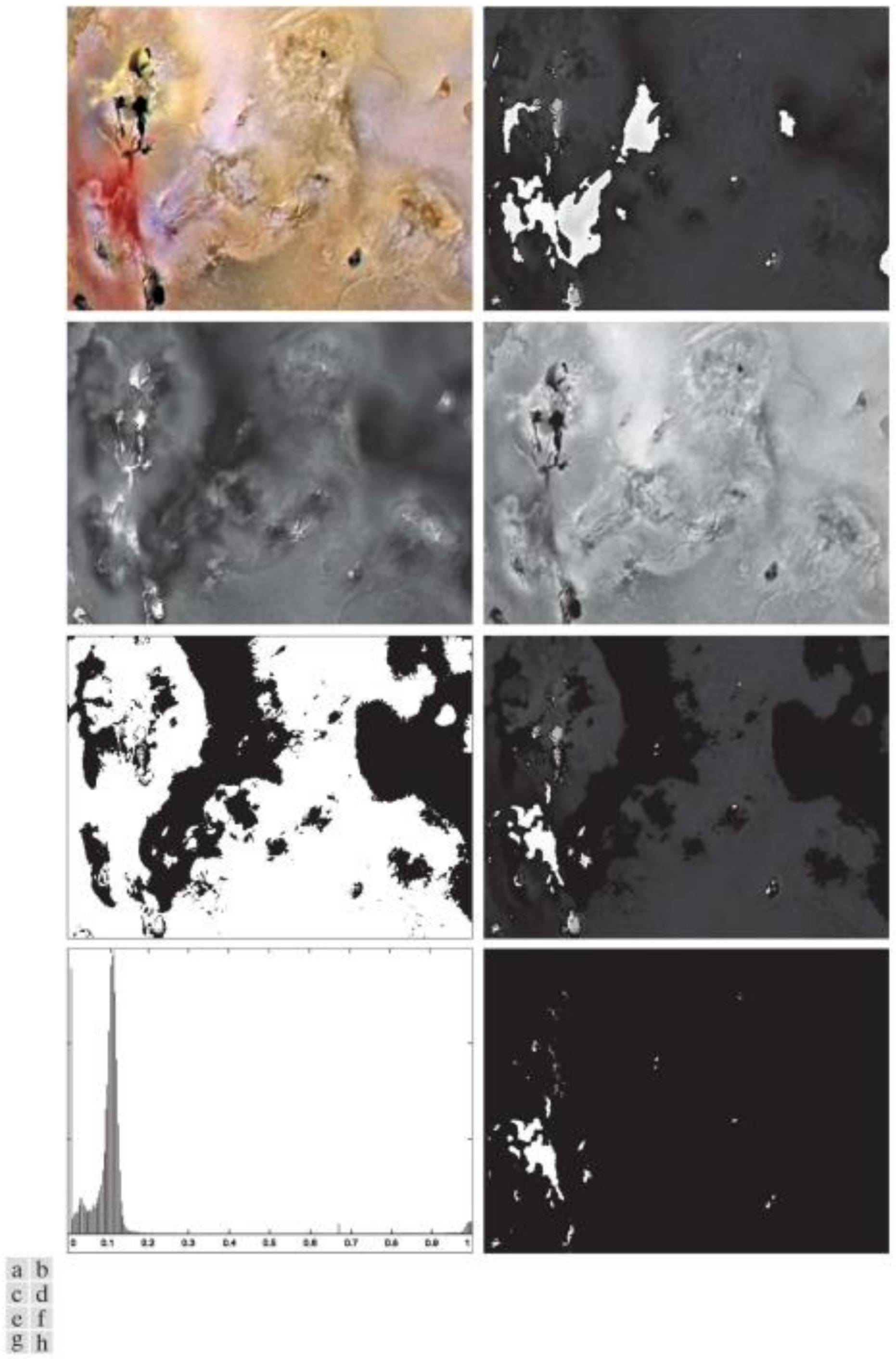


Figure 5.34 Image segmentation in HSI space. (a) Original. (b) Hue. (c) Saturation. (d) Intensity. (e) Binary saturation mask (f) Product of (b) and (e). (g) Histogram of (f). (h) Segmentation of red components from (a).

(Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

Figure 5.34 (f) depicts the product of the mask and the hue image, and **Figure 5.34** (g) depicts the histogram of the product image (note that the grayscale is in the range [0, 1]). We can observe in the histogram that the high values (which correspond to the values of interest) are grouped at the very high of the grayscale, close to the value of one. **Figure 5.34** (h) depicts the binary image produced by thresholding a product image with a threshold value of 0.9 as the outcome of the thresholding operation. Using the spatial location of the white spots in this image, we can determine which points in the original image have the reddish hue that we are looking for. The segmentation approach used here was far from ideal, as there are areas in the original image where we would definitely state there is a reddish hue, but which were not identified by this segmentation method. The regions depicted in white in **Figure 5.34** (h) are, however, the best that this approach can accomplish in detecting the reddish components of the original image, as determined by trial and comparison. The segmentation method detailed in the following section has the potential to produce better outcomes than the previous method.

Segmentation in RGB vector Space

Working in HSI space is more intuitive, segmentation is one area in which better results generally are obtained by using RGB color vectors. The approach is straightforward. Suppose that the objective is to segment objects of a specified color range in an RGB image. Given a set of sample color point's representative of the colors of interest, we obtain an estimate of the "average" color that we wish to segment. Let this average color be denoted by the RGB vector a. The objective of segmentation is to classify each RGB pixel in a given image as having a color in the specified range or not. In order to perform this comparison, it is necessary to have a measure of similarity. One of the simplest measures in the Euclidean distance. Let z is similar to a if the distance between them is less than a specified threshold, D0. The Euclidean distance between z and a is given by

$$\begin{aligned} D(\mathbf{z},\mathbf{a}) &= \| \mathbf{z}-\mathbf{a} \| \\ &= \left[(\mathbf{z}-\mathbf{a})^T(\mathbf{z}-\mathbf{a})\right]^{\frac{1}{2}} \\ &= \left[(\mathbf{z}_R-\mathbf{a}_R)^2+(\mathbf{z}_G-\mathbf{a}_G)^2+(\mathbf{z}_B-\mathbf{a}_B)^2\right]^{\frac{1}{2}} \end{aligned}$$

In this case, the subscripts R, G, and B represent the RGB components of vectors a and z, respectively. As shown in the illustration, the locus of points such that D (z, a) D0 is a solid sphere with radius D0, and this is known as the locus of points. **Figure 5.34** (a). Points that are contained within or on the surface of the object. Points outside the sphere satisfy the provided colour condition; spheres that satisfy the specified colour criterion sphere od does not exist. It is necessary to code these two groups of points in the picture. Using, for example, black and white, one can create a binary segmented image.

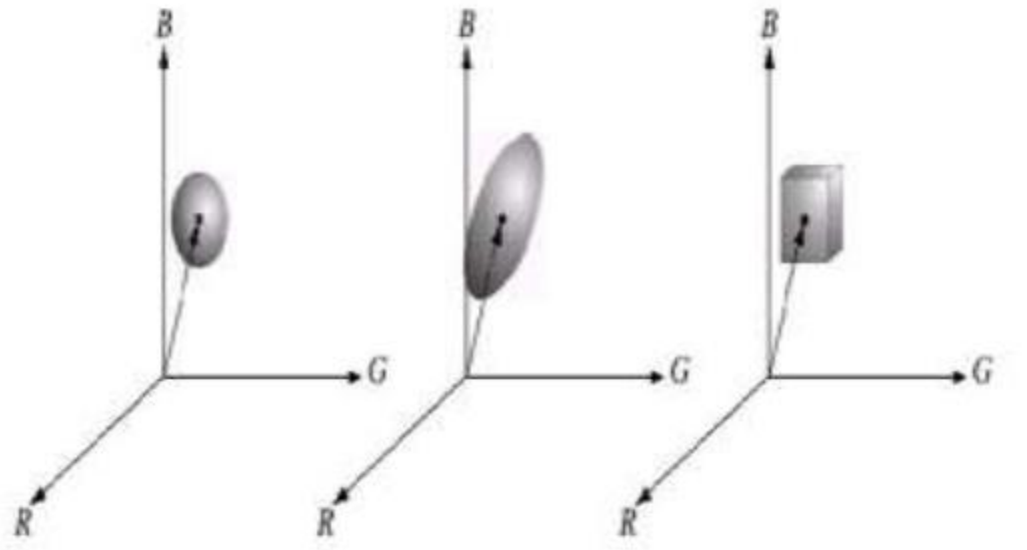


Figure 5.34 Three approaches for regions for RGB vector segmentation.(a->b->c)

(Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

5.18 Noise in Color Images

It is possible to use color images with the noise models. While the noise content of a color image typically has the same properties in each color channel, it is possible for separate color channels to be affected differentially by noise in some cases. If the electronics of a particular channel malfunction, this is one possibility to consider. Different noise levels, on the other hand, are more likely to be produced by changes in the relative amounts of noise. A measure of the amount of illumination supplied to each individual color channel using a red filter in a CCD camera, for example, can significantly weaken the image. because of the amount of illumination detected by the red sensor elements Because CCD sensors are noisier at lower levels of illumination, the resulting red light is more intense. In this circumstance, the noise component of an RGB image would tend to be louder than the noise components of the other two component images.

In this example, we explore the impact of noise in colour images. The image in **Figure 5.35** (a) through (c) depicts the three primary colours in an RGB image with added additive Gaussian noise, whereas **Figure 5.35** (d) represents the final image after the colours have been mixed together. In color images, fine grain noise, such as this, appears less obvious since it is masked by colour. The results of converting the RGB image in **Figure 5.35** to HSI are shown in **Figure 5.35** (a) through (c). Consider how much worse the hue and saturation values in the noisy image have gotten when compared to the original HSI image. Similarly, the overall image of **Figure 5.35** (c) is a little smoother and has less noise than any of the three noisy RGB images. As can be seen in Equation, the intensity image is the sum of the RGB images

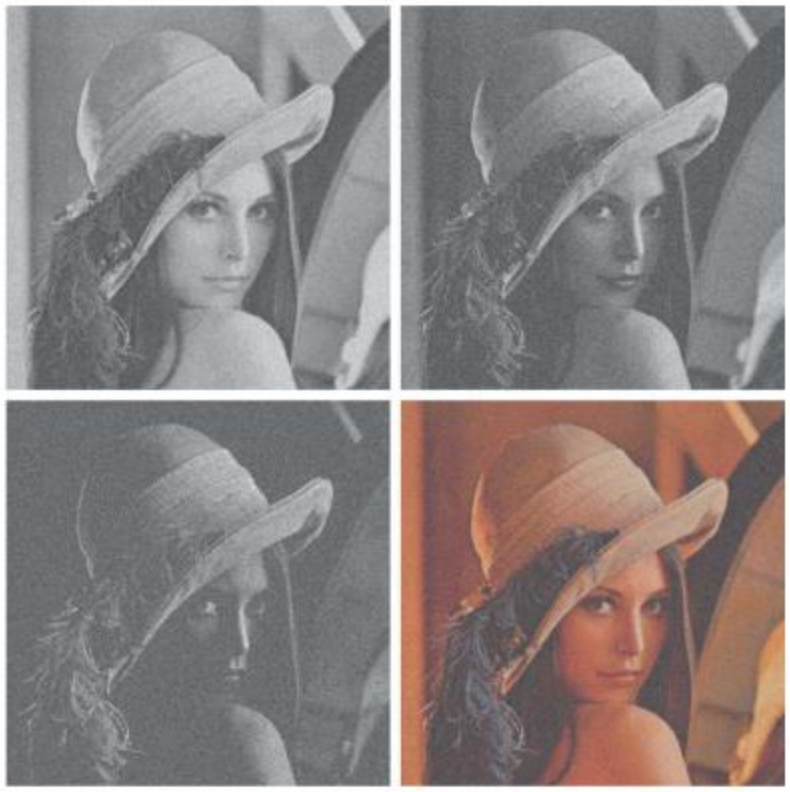


Figure 5.35 (a)–(c) Red, green, and blue 8-bit component images corrupted by additive Gaussian noise of mean 0 and standard deviation of 28 intensity levels. (d) Resulting RGB image

(Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

Figure 5.36 HSI components of the noisy color image in Figure 5.35 (d) . (a) Hue. (b) Saturation. (c) Intensity.

(Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

When a single RGB channel is damaged by noise, for example, converting to HSI distributes the noise among all HSI component images. An illustration of this is shown in Figure 5.36. Figure 5.36 (a) illustrates an RGB image with a corrupted green component caused by salt and pepper noise with a probability of either salt or pepper of 0.05. The HSI component images in Figure 5.36 (b) through (d) clearly demonstrate how the noise extended from the green RGB channel to all of the HSI images. Naturally, this is not surprising, as the HSI components are computed using all RGB components.

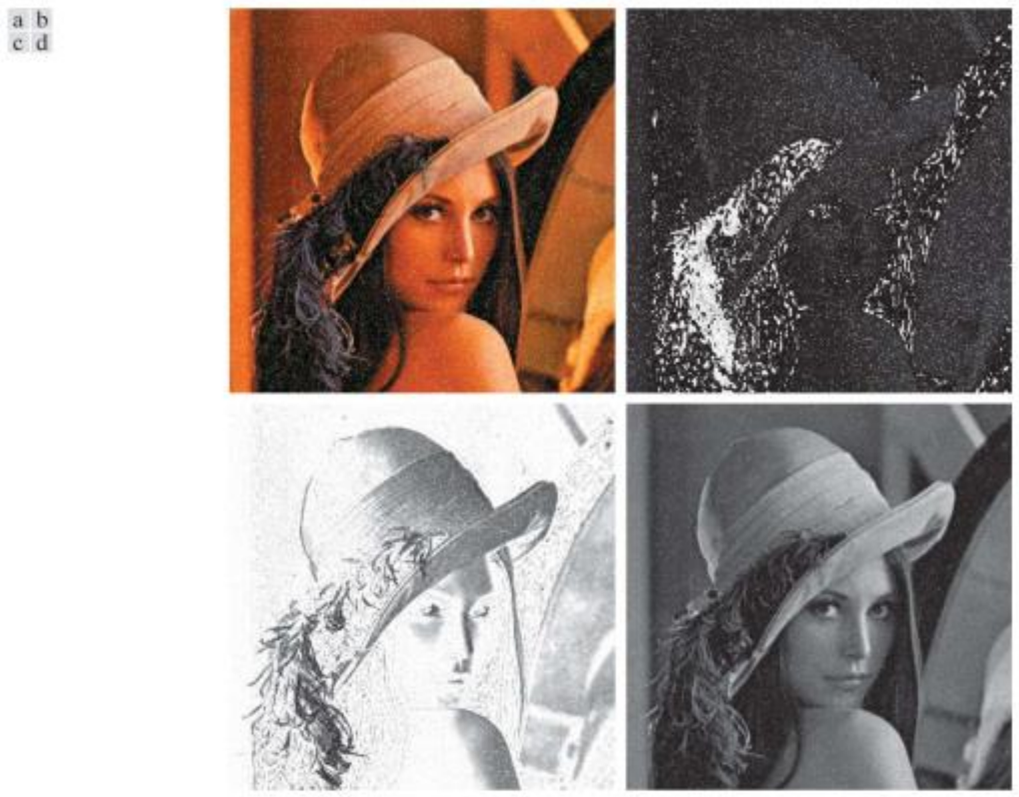


Figure 5.36 (a) RGB image with green plane corrupted by salt-and-pepper noise. (b) Hue component of HSI image. (c) Saturation component. (d) Intensity component

(Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

5.19 Color Image Compression

Due to the fact that the number of bits necessary to represent colour is often three to four times that required to represent grey levels, data compression is critical for the storing and transmission of colour images. In the previous sections' RGB, CMY(K), and HSI images, the data that are compressed are the components of each colour pixel (e.g., the red, green, and blue components of the pixels in an RGB image); these are the mechanism by which colour information is given. Compression is the process of removing duplicate and/or unneeded data.

Figure 5.37 (a) depicts a 24-bit RGB full-color representation of an iris, where each of the red, green, and blue components is represented by eight bits. Figure 5.37 (b) was created using a compressed version of the image in (a) and is thus a compressed and then decompressed approximation of it. Although the compressed image cannot be displayed directly on a colour monitor—it must be decompressed first—it comprises just one data bit (and hence one storage bit) for every 230 bits of data in the original image. Suppose that the image is of size 2000 × 3000 $=6.10^6$ pixels. The image is 24 bits/pixel, so it storage size is $144 \cdot 10^6$ bits.

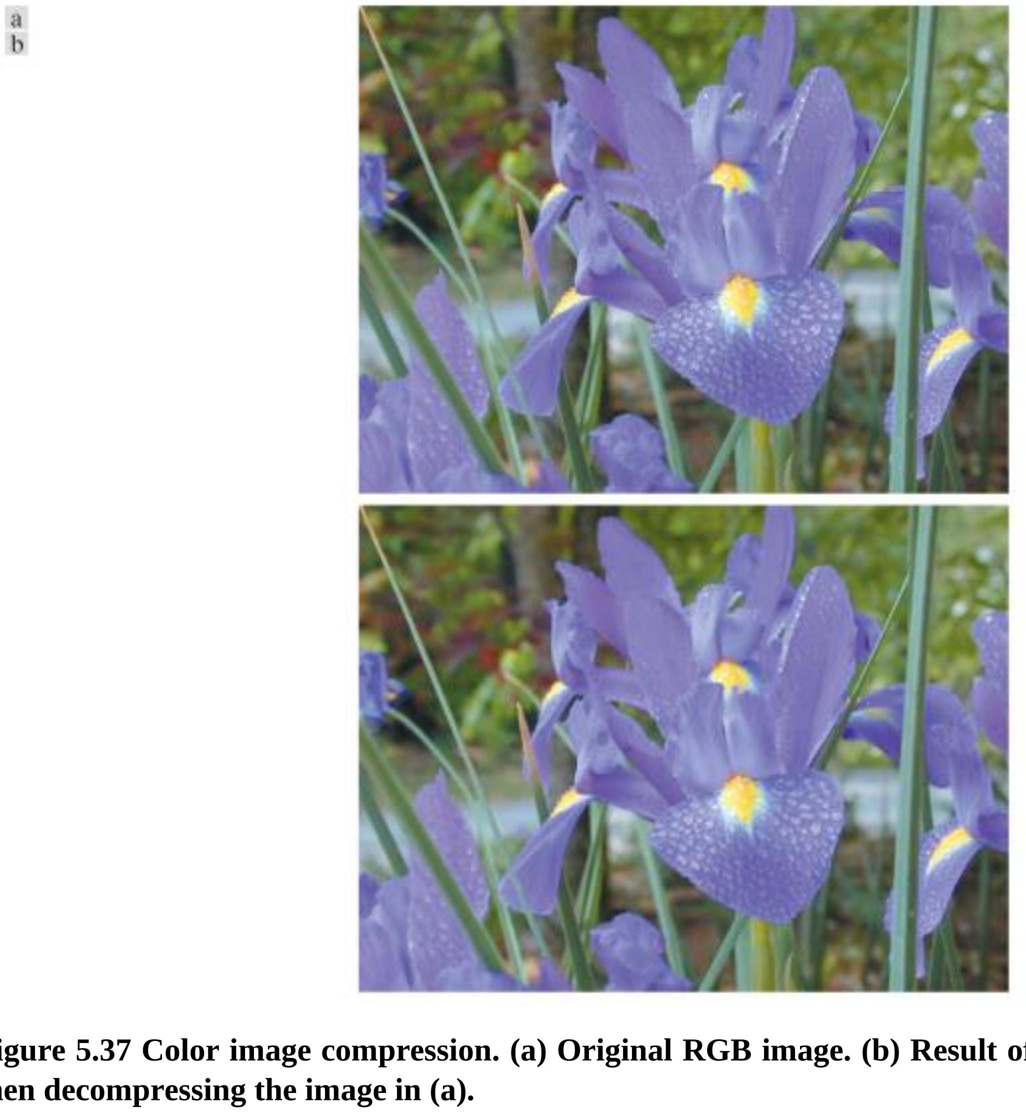


Figure 5.37 Color image compression. (a) Original RGB image. (b) Result of compressing, then decompressing the image in (a).

(Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

5.20 Unit End questions

1. Explain the Matrix-Based Transforms.
2. Explain the basic functions in the Time-Frequency Plane.
3. Write short note on Discrete Hartley Transform.
4. Write short note on Slant Transform.
5. Write short note on Haar Transform.
6. Write short note on Wavelet Transforms.
7. Explain the color fundamentals.
8. Explain the color models.
9. Explain the Pseudocolor Image Processing
10. Explain the color image Segmentation.
11. Explain the noise in the color images.
12. Explain the Color image compression.

5.21 Reference for further reading

1. Digital Image Processing Gonzalez and Woods Pearson/Prentice Hall Fourth 2018
2. Fundamentals of Digital Image Processing A K. Jain PHI
3. The Image Processing Handbook J. C. Russ CRC Fifth 2010

Chapter 6 Image Compression and Watermarking:

Introduction

In recent years multimedia product development and demand have increased more and more, contributing to an inadequate network bandwidth and memory storage device. The theory of data compression therefore becomes increasingly important in order to reduce the redundancy of data, saving more bandwidth and space in hardware. The process of encoding information using fewer bits or other information units than an unencoded representation is data compression or source-coding in computer science and information theory. It helps to reduce costly resource consumption, for example, disk space or bandwidth transmission. Theory and practice will be introduced, the most commonly used compression techniques examined, and the industry standards described, that will help us. This section concludes on the process of introducing visible and invisible data (such as information concerning copyright) to images into digital images watermarking.

6.1 Fundamentals

Data compression, also known as compaction, the process by which data is reduced, typically using encoding techniques, necessary for storage or the transfer of a given information piece. There must be a clear distinction between information and data. It's not synonymous with it. Actually, data is the means of transmitting information. The same amount of information may be represented by different amounts of data. You can see an example in order to understand image redundancy or data redundancy in digital image processing. Suppose two people told a story from Ramesh and Suresh. In the comparison with Ramesh, Suresh told the story in less words, where Ramesh had too many words to tell the same story. Either Ramesh said irrelevant information/data that isn't the part of his story, or he repeated his words several times. In other words, it includes data (or words) not providing any relevant information or simply restoring what has already been known. This means that data redundancy is included

Data redundancy is a key issue for the compression of digital images. It is not an abstract concept, but a quantifiable entity mathematically. If n1 and n2 indicate the number of information-carrying in two data sets that comprise the same information, the relative data redundancy (R_D) of the first data set (n1) can be defined as

$$R_D = 1 - \frac{1}{C_R}$$

Where C_R is commonly referred to as the compression ratio

$$C_R = \frac{n_1}{n_2}$$

In case n2 = n1, C_R = 1, R_D = 0 shows that the first representation of the information does not contain redundant data (related to the second dataset). In n2 << n1,$C_R \implies \infty \; R_D \implies 1$ which involves considerable compression and highly redundant data. Finally, the second set of data contains a lot more than the original representation of $n_2 \gg n_1 C_R \implies 0 \; R_D \implies \infty$, respectively. Naturally this is the usually unwanted case of expansion of the data. C_R and R_D are generally intervals (0, ∞) and (-∞, 1), respectively. A useful Compression Ratio, such as 10 (or 10:1), means that for each one unit in a second or compressed data set, a first data set contains ten information carrying units (say bits). The corresponding 0.9 redundancy means that 90% of the first data is redundant.

Three fundamental data redundancies could be recognized and used in digital image compression: coding redundancy, interpixel redundancy, and psychovisual redundancy. Data compression is obtained by reducing or eliminating one or more of these redundancies.

Coding Redundancy:

In this we use a wording to demonstrate how an image's grey histogram can also provide a good insight into code construction in order to minimise the amount of data used for it.

Again suppose a discrete random variable r_k in interval [0, 1] is the grey levels in the image and every r_k is likely to occur probability p_r (r_k).

$$p_r\ (r_k) = \frac{n_k}{n} \quad k = 0,1,2,\dots,L-1$$

Where L is the number of grey level, n_k is the number that is demonstrated in the image with the k^{th} grey level, and n is the total pixel number in the image. If the number of bits used for each r_k's value is l (r_k), the average number of bits needed for each pixel is

$$L_{avg} = \sum_{k=0}^{L-1} l\ (r_k) p_r(r_k)$$

In other words, the average length of the code words allocated to the different grey level values is determined by summing up each grey level's number of bits and probability of the grey level occurrence. The number of bits needed to code the M X N image is also MNL_{avg}.

Interpixel Redundancy:

Take into account the images in Fig. 6.1(a) and (b). These images show almost identical histograms as shown in Figs. 6.1(c) and (d). It is also important to note that both histograms are trimodal, and that three dominant ranges of grey level values are present. Due to the not equally likely grey values of these images, variable length coding can be applied for reducing the redundancy of coding resulting from a straight or natural binary pixel coding. Nevertheless, the coding process does not affect the correlation level between the pixels in the images. That is, the codes used to represent each image's grey levels are not related to the correlation between pixels.

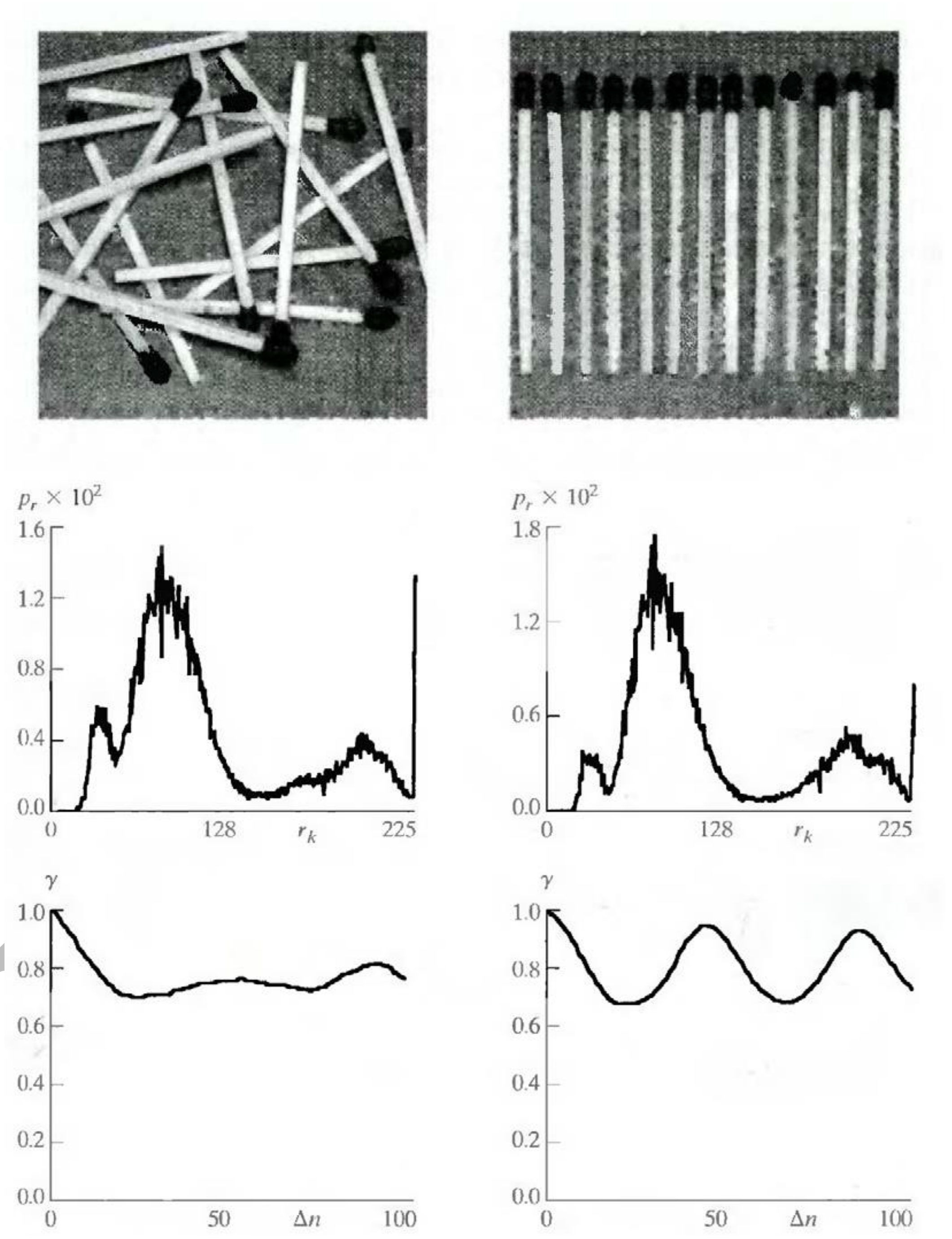


Fig.6.1 Two images and their gray-level histograms and normalized autocorrelation coefficients along one line.

(Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

The autocorrelation coefficient calculated along a line of each picture is shown in Figures 1.1(e) and (f).

$$\gamma(\Delta n) = \frac{A(\Delta \mathrm{n})}{A(0)}$$

Where

$$A(\Delta n) = \frac{1}{N - \Delta n} \sum_{y=0}^{N-1-\Delta n} f(x,y) f(x, y + \Delta n)$$

The above scaling factor is the number of sum terms for each integer value of Δn. This value is variable. Naturally, the total of pixels on a line must be Δn strictly lower than N. The variable x is the line co-ordinate for the computation used. Notice the shape of functions in Figs. 6.1(e) and (f). Their shapes can be associated objectively to the structure in Figs. 6.1(a) and (b). This relationship is especially apparent in Fig. 6.1(f), where there is a direct correlation between both the significant correlations among both pixels divided into 45 and 90 samples between the vertically oriented matches of Fig. 6.1. (b).

These images represent another significant form of data redundancy – one which is directly related to the image's interpixel correlations. The information contained in individual pixels is comparatively small because the value of any given pixel could be adequately predicted from the value of its neighbours. A great deal of a single pixel's visual contribution to a picture is redundant, based on its neighbours' values. A variety of names have been coined to refer to these interpixel dependencies, including spatial redundancy, geometric redundancy, and frame redundancy. To include all of them we use the term interpixel redundancy.

The 2D pixel array usually used for human visualization and interpretation should be changed into a more effective (but usually nonvisual) format to reduce interpixel redundancies in an image. An image may for instance be represented by the differences between adjacent pixels. This type of transformations is known as mapping (i.e. those which delete interpixel redundancy). They are called reversible mappings if you can reconstruct the original picture components from the transformed data package.

Psychovisual Redundancy:

The brightness of a region, as the eye perceives, depends not only on the light reflected by that region. For instance, variations in intensity (Mach bands) in an area of constant intensity can be perceived. Such phenomena arise because the eye does not respond to all visual information with equal sensitivity. Some information is simply less important for normal visual processing than other information. It is said that this information is psycho-visually redundant. It can be removed without adversely affecting the quality of the perception of images.

It is not surprising that psychovisual redundancies are in place, since human perception of the data in an image does not usually require a quantitative analysis of each pixel value in an image. An observer usually investigates and psychologically combines characteristics, such as edges and textures, into recognizable groups. In order to complete the process of image interpretation the brain correlate these groups with previous knowledge. Psycho-visual redundancy differs from the previously mentioned redundancies. Psychovisual redundancy is associated with actual or measurable visual information, in contrast to coding and interpixel redundancy. The removal of the information is only possible as it is not necessary for the normal visual processing. As the removal of redundant psycho-visual data causes the loss of quantitative information, quantization is commonly called.

This terminology corresponds to the ordinary use of the word, usually by mapping a wide range of input values to a limited number of output values. The quantization results in loss of data compression, since the operation is irreversible (visual information lost).

Fidelity Criteria

Realizable or quantitative information is lost due to the removal of psychovisually redundant data. Since information relevant is lost, it is highly desirable to use repeatable or reproducible methods to quantify the nature and extent of the data loss. As the basis for such an evaluation, there are two general classes of criteria:

A) Objective fidelity criteria and

B) Subjective fidelity criteria.

When the information loss level could be described as a function of the original or input image as well as the compressed and then decompressed output image, an objective fidelity criterion is stated for this. The root-mean-square error (rms) of an input to the output image is a good example. That f(x, y) is an image input, and allow f(x, y) to indicate an estimation or approximation of f(x, y) which is the result of compression and decompression of the input afterwards. e(x, y) error between f (x, y) and $\hat{f}(x,y)$for any value of x and y can be defined as

$$e(x,y) = \hat{f}(x,y) - f(x,y)$$

so that the cumulative error among the two images is

$$\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}\left[\hat{f}(x,y) - f(x,y)\right]$$

Where the images are M X N. The root-mean-square error, e_{rms} is the square root averaged over the M X N array and the errors between f(x, y) and $\hat{f}(x,y)$

$$e_{rms} = \left[\frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}\left[\hat{f}(x,y) - f(x,y)\right]^2\right]^{1/2}$$

The mean-square signal-to-noise ratio of the compressed-decompressed image is a closely linked objective fidelity criterion. If$\hat{f}(x,y)$ is considered to be the sum of the original image f(x, y) and a noise signal e(x, y), the mean-square signal-to-noise ratio of the output image, denoted SNR_{rms}, is

$$SNR_{rms} = \frac{\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}\hat{f}(x,y)^2}{\frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}\left[\hat{f}(x,y) - f(x,y)\right]^2}$$

In the square root of Eq. above, the rms value of the signal-to-noise relation, known as the SNR_{rms}, is derived.
While objective fidelity criteria provide a simple and convenient mechanism for assessing loss of information, human beings are ultimately viewed the most decompressed images. Therefore, it is often more appropriate to measure image quality with a human observer's subjective assessments. To this end, a "typical" decompressed image is shown in a corresponding cross section of viewers and the evaluations are averaged. The assessments can be carried out using an absolute evaluation scale or by comparing f(x, y) and $\hat{f}(x,y)$side-by-side.

Image compression models
Fig.6.2 illustrates that there are two different structural blocks of a compression system, one encoder and one decoder. The input image f(x,y), which produces a set of symbols from the input data, is fed into the encoder. The encoded representation is then fed into the decoder after transmission through the channel to produce the revised output $\hat{f}(x,y)$. $\hat{f}(x,y)$can be an accurate f (x, y) replica in general, or not. If so, the system is error-free or information-free; otherwise, the reconstructed image contains some degree of distortion. Two relatively independent functions or subblocks comprise both the encoder and decoder shown in Fig.6.2. The encoder consists of a source encoder that eliminates input redundancies and a channel encoder that enhances the noise immunity of the output from the source encoder. As intended, a channel decoder with a source decoder is included with the decoder. If there is noise free (not an error) channel between the encoder and decoder, the channel encoder and Decoder shall be removed and the specific encoder and Decoder shall be respectively the source encoder and decoder.

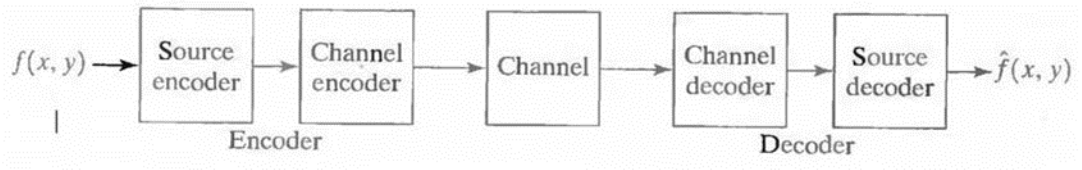


Fig.6.2 A general compression system model
Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

The Source Encoder and Decoder:

The source encoder must reduce or delete any coding, interpixel or psychovisual redundancies in the input image. In each case, the particular application and related fidelity criteria determine the best encoding method of use. Usually, a series of three independent operations can model this approach. As illustrated by Fig. 6.3 (a), one of the three redundancies is reduced by each operation. The corresponding source decoder is shown in Figure 6.3 (b). The mapper converts the input data into (usually non-visual) format in the first step of the source encoding process so that interpixel redundancies are reduced in the input image. Generally, this operation is reversible and can decrease the data needed to represent the image directly or not.

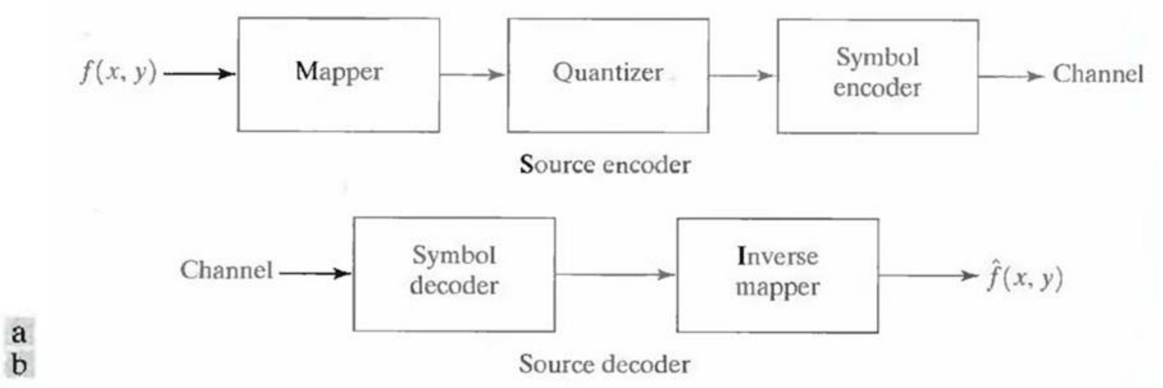


Fig.6.3 (a) Source encoder and (b) source decoder model
Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

Run-length coding is an instance of mapping that leads directly to data compression in this initial phase of the whole encoding process. An example of the opposite is the image representation by a set of transforming coefficients. This is where the mapper converts the image into a range of coefficients, rendering its interpixel redundancies available in later phases of the encoding process for compression.
The second step, or quantizer block in figure 6.3(a), decreases the precision of the output of the mapper according to a fidelity criterion pre-established. The psycho-visual redundancies of the image are reduced by this stage. This is an irreversible procedure. It is an irreversible operation. Therefore, if the compression is required, it must be omitted.

The symbol coder generates a fixed or variable-length code to constitute the quantizer output in the third and final phases of the source encoding process and maps the source output to a code. The term symbol coder differs from the entire source encoding process. This coding operation is the same. In most cases, the mapped and quantized data set is represented by variable-length code. It provides the shortest code words to the output values that most often occur and reduces the redundancy of the code. Naturally, the operation is reversible. After the symbol coding step is completed, each of the three redundancies has been removed by the Image.

Figure 6.3(a) indicates three successive operations in the source encoding process although not all three are generally contained in each compression system. Remember, for instance, that if error-free compression is required, the quantizer must be removed.

Fig. 6.3.(a). The mapper and quantizer, for example, are often represented in the predictive compression systems by a single block, which performs both operations simultaneously.Fig 6.3(b) includes only two components in a source decoder: a decoder for symbols and an inverse mapper. The inverted operations of the source encoder and mapper blocks are performed on these blocks in the reverse order. Since quantizations lead to irreversible loss of information, the general sources decoder model shown in Fig. 6.3 (b) does not include reverse quantizing blocks.

The Channel Encoder and Decoder:

In the overall encoding -decoding process, a channel encoder and decoder play a significant role when the channel in Fig. 6.3 is noisy or susceptible to error. They are intended to decrease channel noise by inserting in the source encoded data a controlled form of redundancy. Since the source encoder's output is low in redundancy, transmission noise without this controlled redundancy is highly sensible. R. W. Hamming developed one of the most effective channel coding methods (Hamming [1950]). It is based on adding sufficient bits to the encoded data to ensure that certain minimum bits are changed between valid words. (Multi-bit errors can be detected and corrected by adding extra redundancy bits.) The Hamming 7-bit number (7, 4) is a 4-bit binary number b3b2b1b0, which is h_1, h_2, h_3...., h_6, h_7.

$$h_1 = b_3 \oplus b_2 \oplus b_0$$
$$h_2 = b_3 \oplus b_1 \oplus b_0$$
$$h_4 = b_2 \oplus b_1 \oplus b_0$$
$$h_3 = b_3$$
$$h_5 = b_2$$
$$h_6 = b_1$$
$$h_7 = b_0$$

Where X refers to an exclusive OR operation. Please note: bits h_1, h_2 and h_4 are equal parity bits respectively for b_3 b_2 b_2, $b_3b_1b_0$, and $b_2b_1b_0$. (Recall that a string of binary bits has even parity if the number of bits with a value of 1 is even.) The channel decoder must check the coded value for an odd parity over the bit fields, even parity, to decode a Hamming encoded result. The non-zero word $C_4c_2c_1$ is a one-bit error.

$$c_1 = h_1 \oplus h_3 \oplus h_5 \oplus h_7$$
$$c_2 = h_2 \oplus h_3 \oplus h_6 \oplus h_7$$
$$c_4 = h_4 \oplus h_5 \oplus h_6 \oplus h_7$$

When a non-zero value is identified, the decoder will simply add the word code bit position that is indicated by the word parity. The decoded binary value is then removed as $h_3h_5h_6h_7$ from the corrected code word.

Variable-Length Coding:

The easiest way to compress images without errors is to only minimise coding redundancy. Coding redundancy is usually available in any normal grey level binary encoding of a grey scale image. The grey levels can be removed by coding. In order to do that, a varying-length code must be constructed which assigns the shortest possible code words to the most probable grey levels. Here, we look at various optimal and almost optimal methods for building such a code. The techniques are expressed in the information theory language. Practically the source symbols may be the grey image levels or the output of a mapping process at a grey level (pixel discrepancies, run-lengths and so on

6.2 Huffman coding:

Huffman is the most popular way to remove coding redundancy (Huffman [1952]). The Huffman code provides the smallest number of code symbols per source symbol when independently coding the symbols of an information source. The resultant code is optimum for a set value of n in terms of the noiseless coding theorem, provided the source symbols are coded one at a time.
The first step in Huffman's technique is to produce a series of reduced source values by arranging the probability of the symbols being considered and combining the lowest probability symbols to the next source reduction in a single symbol. This binary coding process can be seen in Figure 6.4 (K-ary Huffman codes can also be constructed). A hypothesized set of source symbols and their probabilities in terms of decreasing probability values can be ordered from top to bottom. To decrease the initial source, the 0.06 and 0.04 probabilities of the bottom of the two sources are combined with the probability 0.1 of a 'compound symbol.' The compound symbol is placed in the first column for reducing source and its associated probability, so that reduced spring probabilities are also most likely to be determined. This process is then repeated until a reduced source with two symbols (at the far right) is reached.

The second step in the process of Huffman is to code each source that is reduced, beginning with the smallest source and returning to the original source. Naturally, symbols 0 and 1. are the minimal binary code for a two symbol source. As illustrated in Fig. 4.2, the two symbols were allocated to the right (the assignment is arbitrary; reversing the order of the 0 and 1 would work just as well). Since the reduced source symbol is 0.6, combining two two symbols in the reduced source on the left, the code 0 is now allocated to these two symbols and one symbol 0 and 1 are arbitrary appended to each to differentiate between them. For each reduced source, this process was repeated until it reaches the initial source. The final code is shown in fig. 6.5 at the far left. This code has an average length.

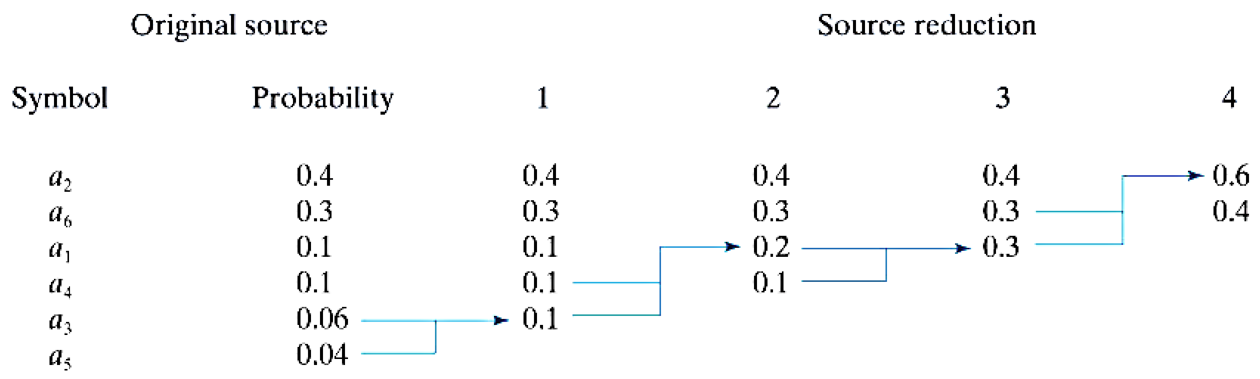
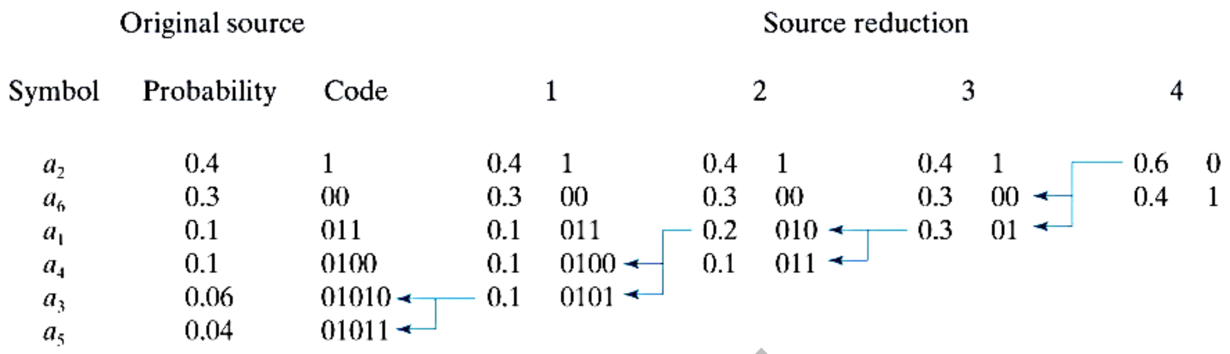


Fig.6.4 Huffman source reductions.



Original source
Source reduction
Symbol
Probability
Code
1
2
3
4
a_2 0.4 1 0.4 1 0.4 1 0.4 1 0.6 0
a_6 0.3 00 0.3 00 0.3 00 0.3 00 0.4 1
a_1 0.1 011 0.1 011 0.2 010 0.3 01
a_4 0.1 0100 0.1 0100 0.1 011
a_3 0.06 01010 0.1 0101
a_5 0.04 01011

Fig.6.5 Huffman code assignment procedure.

L_{avg} = (0.4)(1) + (0.3)(2) + (0.1)(3) + (0.1)(4) + (0.06)(5) + (0.04)(5) = 2.2 bits/pixel

and the entropy of the source is 2.14 bits/symbol. The resulting Huffman code efficiency is 0.973.

The procedure of Huffman generates optimum code for a number of symbols and probabilities that are restricted from the coding of the symbols one at a time. Once the code is generated, it is easy to encode and/or decode in a lookup table way. The code as a whole is a block code that can be decoded instantly. A block code is named as the code is mapped to a fixed sequence of code symbols for each source symbol. It's immediate because every word of code can be decoded into a series of symbols without referring to the following symbols. It is unique in that every code string can be decoded in one way only. Thus any Huffman string of encoded symbols can be decoded by examining left-to-right at the individual string symbols. The left to the right scan of the 010100111100 encoded string shows that the initial word code of the Fig. 6.5 binary code is 01010, which is the symbol code a3. The next valid code is 011, which is symbol a1. The completely decoded message to be a3a1a2a2a6 is revealed in this way.

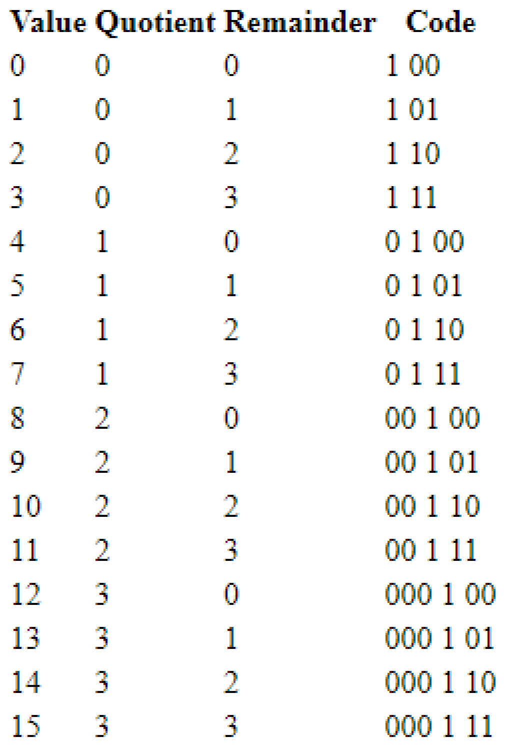
6.3 Golomb Coding

A Golomb code is a variable-length code similar to Huffman; but, unlike Huffman, it is based on a simple concept of the likelihood of the values (which are explicitly treated as natural numbers rather than abstract symbols): small values are more likely than large ones. The precise relation between size and probability is captured in a parameter, the divisor.

To Golomb-code a number, find the divisor's quotient and remainder. The quotient should be written in unary notation, followed by the remainder in truncated binary notation. In practise, a stop bit is required following the quotient: if the quotient is expressed as a succession of zeroes, the stop bit is a one (or vice versa - and people do seem to prefer to write their unary numbers with ones, which is Wrong). The remainder's length can be calculated by the divisor.

A Golomb-Rice code is a Golomb code with a power of two as the divisor, allowing for more efficient implementation via shifts and masks rather than division and modulo.

For example, here's the Golomb-Rice code with divisor 4, for numbers up to 15:



Value	Quotient	Remainder	Code
0	0	0	1 00
1	0	1	1 01
2	0	2	1 10
3	0	3	1 11
4	1	0	0 1 00
5	1	1	0 1 01
6	1	2	0 1 10
7	1	3	0 1 11
8	2	0	00 1 00
9	2	1	00 1 01
10	2	2	00 1 10
11	2	3	00 1 11
12	3	0	000 1 00
13	3	1	000 1 01
14	3	2	000 1 10
15	3	3	000 1 11

The source of information A generates the symbols {A0, A1, A2, A3 and A4} with the corresponding probabilities {0.4, 0.3, 0.15, 0.1 and 0.05}. Encoding the source symbols using binary encoder and Golomb encoder gives:

Source Symbol	P_i	**Binary Code**	**Golomb Code**	**Source Symbol**
A0	0.4	000	0	**A0**
A1	0.3	001	10	**A1**
A2	0.15	010	110	**A2**
A3	0.1	011	1110	**A3**
A4	0.05	100	1111	**A4**
Lavg	H = 2.0087	3	2.05	$\mathbf{L_{avg}}$

The Entropy of the source is

$$\mathrm{H} = -\sum_{i=0}^{4} \mathrm{Pi}\ \log_2 \mathrm{Pi} = 2.0087\ \mathrm{bit/symbol}$$

Since we have 5 symbols ($5<8=2^3$), we need 3 bits at least to represent each symbol in binary (fixed-length code). Hence the average length of the binary code is\

$$\mathrm{Lavg} = \sum_{i=0}^{4} \mathrm{Pi\ li} = 3\ (0.4 + 0.3 + 0.15 + 0.1 + 0.05) = 3\ \mathrm{bit/symbol}$$

Thus the efficiency of the binary code is

$$\eta = \frac{H}{Lavg} = \frac{2.0087}{3} = 67\%$$

The average length of the Golomb code is

$$Lavg = \sum_{i=0}^{4} Pi\ li = 0.4 * 1 + 0.3 * 2 + 0.15 * 3 + 0.1 * 4 + 0.05 * 4 = 2.05 \text{ bit/symbol}$$

Thus the efficiency of the Golomb code is

$$\eta = \frac{H}{Lavg} = \frac{2.0087}{2.05} = 98\%$$

This example demonstrates that the efficiency of the Golomb encoder is much higher than that of the binary encoder.

6.4Arithmetic coding:

Unlike previously mentioned variable-length codes, nonblock codes are created by arithmetic coding. There is no single correspondence between the source symbols and the code words of arithmetic coding that can be traced back to Elias's work. A single arithmetic code word is instead assigned to a whole sequence of source symbols (or message). The word code describes an interval between 0 and 1 of real numbers. With the increase in the number of symbols in the message, the range for representing it becomes smaller and the number of units of information necessary for representing the range is larger. The size of the interval is reduced by each message symbol according to its probability. Since the technique requires that each source symbol does not translate into an integral number of coding symbols (that is, coding the symbols one by one at a time) as is the approach of Huffman, it reaches (but only in theory) the boundary of noiseless coding theorem.

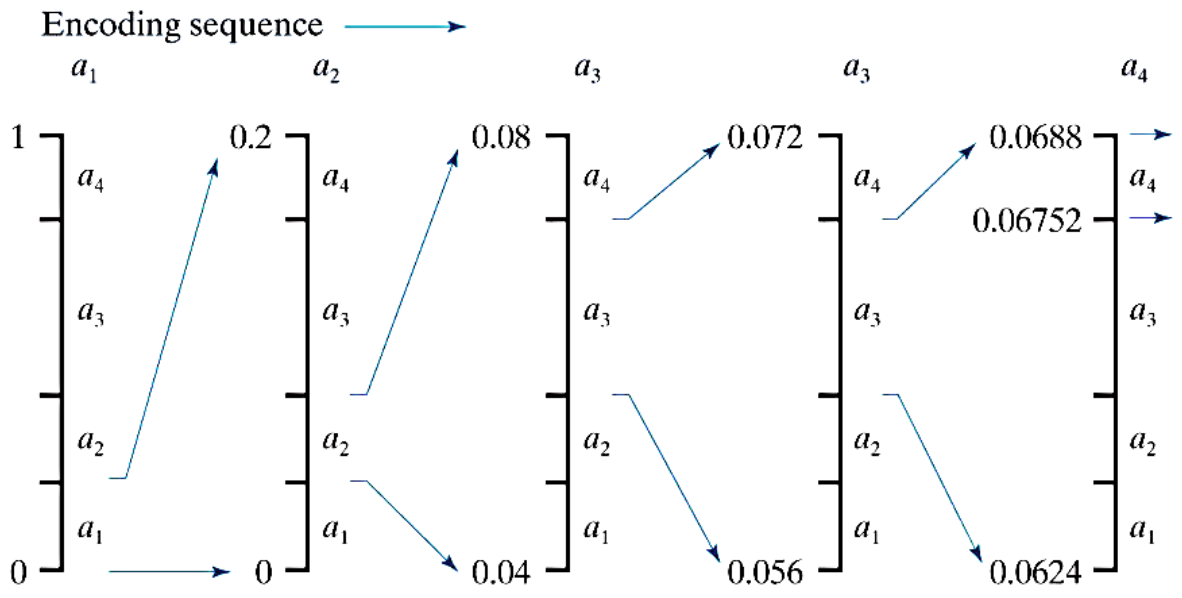


Fig.6.6 Arithmetic coding procedure
Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

The fundamental arithmetic coding procedure is illustrated in Fig.6.6. A five-symbol or message sequence, a1a2a3a3a4, is coded in this section from a four-symbol source. The message is supposed to occupy the complete half-open interval [0, 1) at the beginning of the coding process. As shown in table 5.2, this interval was originally divided into four regions, depending on each source symbol's probabilities. For instance, the symbol ax, is related to the sub-interval [0, 0.2). The message interval is originally restricted to [0, 0.2] because the message is the first symbol of the coded message. Thus Fig.6.6 [0, 0.2] extends the figure to its full height, with the end points marked with the narrowing range values.

Source Symbol	Probability	Initial Subinterval
a_1	0.2	[0.0, 0.2)
a_2	0.2	[0.2, 0.4)
a_3	0.4	[0.4, 0.8)
a_4	0.2	[0.8, 1.0)

Table 6.1 Arithmetic coding example

Symbol a2 reduces the subinterval by [0.04, 0.08), and a3 closes it by [0.056, 0.072), respectively. A special end-of-message indicator should be reserved for the final message symbol, and the range is narrowed by [0.06752, 0.0688). Naturally, any number can be used to represent the message within this subinterval—for example, 0.068.

Three decimal digits represent the five-symbol message in the arithmetically coded message of Fig.6.6. It is evaluated in 3/5 or 0.6 decimal digits by source symbol with entropy, which is 0.58 decimal digits or 10-ary units/symbol.
The resulting arithmetical code reaches the limit introduced by the noiseless coding theorem, as the length of the coded sequence increases.
In practice two factors lead to the coding performance being less than limited (1) by adding the end-of-message indicator required to separate one message from the other; and (2) by applying finite arithmetic precision. The latter issue is addressed in practical implementations of arithmetic coding with a scaling strategy and a rounding strategy (Langdon and Rissanen [1981]). The scaling strategy renormalizes every [0, 1] subinterval before splitting it into the probabilities of the symbols. The rounding strategy ensures that finite precision arithmetic truncation does not prevent correctly representing coding subintervals.

6.5 LZW Coding:

The Lempel Ziv-Welch (LZW) coding method applies fixed-long coding words to variable sequences of source symbols but does not require a priori knowledge of the probability of encoding symbols occurring. LZW compression has been built into a number of mainstream image file formats, such as the interchange format (GIF), tagged image file format (TIFF), and the portable document format (PDF).
Conceptually, LZW coding is very straightforward. A codebook or "dictionary" is created at the beginning of the coding process that contain the source symbols to be coded. The first 256 words in the dictionary are given to grey values 0, 1, 2... And 255 for 8-bit monochrome images. The gray-level sequences not included in the dictionary are algorithmically determined (for example the next unused) locations when an encoder sequences the image pixels consecutively examines. If, for example, the two first pixels of an image are white, sequence "255-255," the address below is reserved for grey levels 0 to 255, and can be assigned to position 256. The next time two white pixels of the following number are encountered, the code word 256 is used to represent them with the location address, which contains sequence 255-255. If in the code process a 9-bit, 512-word dictionary is used, a single 9-bit code word replaces the original bits (8 + 8) used to represent two pixels. Cleary, a major system parameter is the size of the dictionary. If it is too small, it is less likely to detect corresponding grey level sequences; if it is too large, the size of the code words affects the performance of the compression.

Consider the following 4 x 4, 8-bit image of a vertical edge:

39	39	126	126
39	39	126	126
39	39	126	126
39	39	126	126

The steps involved in coding 16 pixels will be described in Table 6.2. The following starting content is supposed to include a 512 words dictionary:

Dictionary Location	Entry
0	0
1	1
⋮	⋮
255	255
256	—
⋮	⋮
511	—

Initially unused are locations 256 through 511. It is encoded with a left-to-right, top-to-bottom processing of its pixels. The variable — column 1 of Table 6.2 — known as the currently recognised sequence is linked to each successive Gray Level value. This variable is null or empty at the start, as can be seen. The dictionary will be searched for each concatenated sequence and the newly concatenated and accepted sequence (i.e. located in the dictionary) will replace the dictionary as it has in the first row of the table. This has been done in row 2 column 1.

There is no generation of output codes or alteration of the dictionary. If, however, the concatenated sequence cannot be identified, the current sequence is output as the next encoded value, a concatenated but unknown sequence is added to the dictionary, and the current pixel value is started in the sequence currently recognized. This was done in table row 2. The last two columns provide a detailed gray-level sequence when the whole 4 x 4 image is scanned. There are defined nine extra words of code. The dictionary includes 265 code words and a number of repetitive gray-level sequences were successfully identified by the LZW algorithm—which uses them in order to reduce the original 128-bit image to 90 bits (i.e., 10 9-bit codes). Reading the third column from top to bottom produces the encoded result. The compression ratio resulting is 1.42:1.

One special feature of LZW coding that has just been demonstrated is that while the data is encoded, the code dictionary or book is created. Noteworthy, the LZW decoder produces the same decompression dictionary as it decodes the encoded data stream simultaneously. While this example does not require a strategy for handling dictionary overflow, most useful applications require a strategy. A convenient way is to flush or reset the dictionary when complete and continue to code with a new initialized dictionary. A more complicated way is to monitor compression and flush the dictionary in bad or unacceptable circumstances. Instead it is easy to track and replace the least used dictionary entries if necessary.

Currently Recognized Sequence	Pixel Being Processed	Encoded Output	Dictionary Location (Code Word)	Dictionary Entry
	39			
39	39	39	256	39–39
39	126	39	257	39–126
126	126	126	258	126–126
126	39	126	259	126–39
39	39			
39–39	126	256	260	39–39–126
126	126			
126–126	39	258	261	126–126–39
39	39			
39-39	126			
39–39–126	126	260	262	39–39–126–126
126	39			
126-39	39	259	263	126–39–39
39	126			
39-126	126	257	264	39–126–126
126		126		

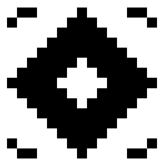
Table 6.3 LZW coding example

6.6 Run-Length Coding

Repeating intensity images on their rows (or columns) may also be compressed by showing the same intensity runs as run-length pairs, where a new intensity starts with each run-length pair and the number of consecutive pixels with that intensity. In the 1950s the technique, known as Run-Length Encoding (RLE) was developed and became the traditional compression method in facsimile coding, along with its 2-D extensions. Compression is accomplished by elimination of a simple type of spatial redundancy – the same-intensity groups. When there are few (or no) runs of identical pixels, run-length encoding results in data expansion.

Run Length Encoding (RLE) is not a technique used too commonly these days, but it is an excellent way of making a sense of some of the compression problems.

Imagine that we have a simple black and white image.



One way a computer can store this image in binary is to use the format of '0' in white, and '1' in black (this is a "bitmap," since pixels have been mapped to bits). The above picture would be shown as follows using this method:

```
011000010000110
100000111000001
000001111100000
000011111110000
000111111111000
001111101111100
011111000111110
111110000011111
011111000111110
001111101111100
000111111111000
000011111110000
000001111100000
100000111000001
011000010000110
```

The main question in compression is whether or not the same image can be represented in fewer bits so that the original image can still be reconstructed.

We can prove it. There are various ways to do this, but we concentrate on a way called run length encoding in this section.

Imagine reading the above bits to someone who copied them... you might tell things like "five zeroes" earlier on rather than "zero zero zero zero c zero." That is the basic concept behind run time encoding (RLE), used in the storage of digital images to save space. In run length encoding, each row is replaced by numbers that indicate how many consecutive pixels the same colour is, often beginning with the number of white pixels. For instance, the first line in the above image contains one white, two black and four white, one black, four white and two black pixel.

011000010000110

This could be represented as follows.

1, 2, 4, 1, 4, 2, 1

For the second row, since we have to say the number of white pixels before saying black, we have to tell specifically that there is zero at the beginning of the row.

100000111000001

You may ask why we must first say the number of white pixels that was zero in this case. The explanation is that if we had no clear rule, the computer would not know which colour was what when the picture in this form was shown.0, 1, 5, 3, 5, 1

The third row contains five whites, five blacks, five whites.

000001111100000

This is coded as:

5, 5, 5

That means we get the following representation for the first three rows.

1, 2, 4, 1, 4, 2, 1

0, 1, 5, 3, 5, 1

5, 5, 5

You can work out what the other rows would be following this same system.

Representation for the remaining rows

The remaining rows are

4, 7, 4

3, 9, 3

2, 5, 1, 5, 2

1, 5, 3, 5, 1

0, 5, 5, 5

1, 5, 3, 5, 1

2, 5, 1, 5, 2

3, 9, 3

4, 7, 4

5, 5, 5

0, 1, 5, 3, 5, 1

1, 2, 4, 1, 4, 2, 1

Converting run length encoding back to the original representation

Just to ensure that we can reverse the compression process, have a go at finding the original representation (zeroes and ones) of this (compressed) image.

4, 11, 3

4, 9, 2, 1, 2

4, 9, 2, 1, 2

4, 11, 3

4, 9, 5

4, 9, 5

5, 7, 6

0, 17, 1

1, 15, 2

The CCITT (consultative committee of the international telegraph and telephone) and run-length coding are CCITT standards for encoding a binary and gray-level image. With this method, images are scanned row by row and identifies the run . The output vector run-length pixel value and run length is defined.

$H_{run\text{-}length} = (H_0 + H_1)/ (L_0 + L_1)$

H_0 = entropy of the black run

H_1 = entropy of white run

L_0 = average value of black run

L_1 = average value of white run

6.7 Symbol-Based Coding

It is also referred to as token-based coding in some circles. A symbol is a collection of sub images that appear frequently in a given context. Each symbol is stored in a symbol dictionary, and the image is coded as an asset of the triplet (x1, y1, t1), (x2, y2, t2), etc., where the location of the symbol is specified by the triplet (xi, yi) , where ti is the token, which is a reference to the symbol in the dictionary, and each triplet represents an instance of the dictionary symbol in the image.

Consider the straightforward bilevel image shown in Fig. 6.7 (a). It includes only one word, banana, which is made up of three distinct symbols: a b, three a's, and two n's.   Given that the letter b is the initial symbol recognized during the coding process, its 9 X 7 bitmap is stored at location 0 of the symbol dictionary, as shown in the following diagram. As illustrated in Fig. 6.7 (b), the token identifying the b bitmap is the number 0. To illustrate how this works, the first triplet in the encoded image's representation [see Figure 6.8.7(c)] is (0, 2, 0,] denoting that the upper-left corner (an optional convention) of the rectangular bitmap indicating the b symbol is to be placed at the location (0, 2) in the decoded image. It is possible to encode the remaining portion of an image with five extra triplets after the bitmaps for   n symbols have been detected and added to the dictionary, as shown in the following example. Compression occurs as long as the six triplets required to locate the symbols in the image, as well as the three bitmaps required to define them, are lower in size than the original image. It is assumed that each triplet is formed of three bytes in this example, and that the starting picture has or 459 bits. In this case, the compressed representation has or 285 bits and the compression ratio c=1.61 is obtained by multiplying the starting image by (6 3 8 + [(9 7) + (6 7) + (6 6)]. For example, to decode the symbol-based representation shown in Figure 6.7 (c), you simply read the bitmaps of the symbols given in the triplets from the symbol dictionary and arrange them in the appropriate positions at the spatial coordinates specified in the respective triplets.

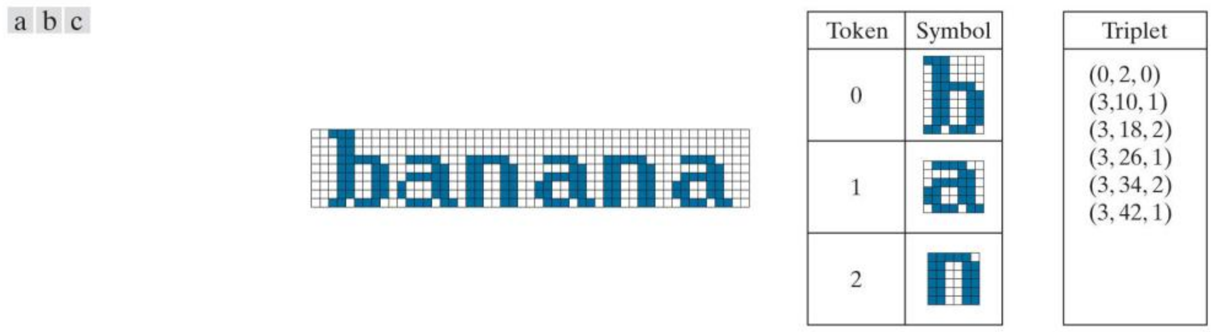


FIGURE 6.7 (a) bi-level document, (b) symbol dictionary, and (c) the triplets used to locate the symbols in the document.

Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

In the beginning of the 1970s, Symbol-based compression was suggested, but only recently became practical. Improvements in symbol matching and enhanced CPU processing speeds have enabled both dictionary symbols to be selected and to see where they can be found in an image in time. And symbol-based decoding is definitely faster than encoding, as with many other compression methods. At last, we observe that to further improve the compression performance, both the bitmaps symbols which are stored in the dictionary and the three versions used for the reference themselves can be encoded. If the resulting compression is lossless as in fig. 6.7, only exact symbol matches are permitted. If small differences are allowed, there will be some level for the reconstruction error.

6.8 8 Bit-Plane Coding:

An effective approach to reduce the redundancy of the interpixels of an image is to independently process the image bit planes. The technique, called Bit Plane Coding (bitplane coding), is based on the principle of decomposing multilevel (monochromatic or color-based) images into a series of binary images.

Bit-plane decomposition:

A m-bit grey image in the form of the base 2 polynomial could be displaying the grey levels

$$a_{m-1}2^{m-1} + a_{m-1}2^{m-1} + \cdots + a_1 2^1 + a_0 2^0$$

Based on this property, the m coefficients of the polynomial into 1-bit bits is a simple way of decomposing the image into a series of binary images. The By collecting the a0 bits of each pixel the zero-order bit plane is formed, while (m - 1) the bit bit plane contains am-1, bits or coefficients. Generally, the number of every bit plane from 0 to m-1 is determined by setting its pixels to equal the values of each pixel of the respective bits and polynomial coefficients of the original image. The inherent drawback is that small changes in the grey level can have a considerable effect on the complexity of bit planes. When the 127 (01111111) intensity pixel is neighboring to the 128 (10000000), for example, the corresponding 0 to1 (or 1 to 0) transitioning shall be included on every bit plane. As the two binary codes for 127 and 128 are different most important bits, for example, Bit plane 7 will have a zero-valued pixel next to a pixel value 1, which will create a transition between 0 to 1 (or 1 to 0) at this point.

$$g_i = a_i \oplus a_{i+1} \quad 0 \le i \le m-2$$
$$g_{m-1} = a_{m-1}$$

A different approach to decomposition (which eliminates the influence of small variations in grey levels) is to first image representation with the m-bit Gray code. From Here, X denotes the exclusive OR operation: The m-Bit Gray Code gm-1... g2g1g0 which corresponds to the polynomial in Eq. above. The only feature of this code is that the following words differ only in one bit. Small grey level changes are also unlikely to affect all mbit levels. For instance, only the 7th bit plane can contain transitions of 0 to 1 when grey level 127 and 128 are next to each other, because the Gray codes of 127 and 128, respectively, are 11000000 and 01000000.

6.9 Transform Coding:

Both predictive coding methods work directly on the image pixels and are also spatial domain methods. In this code, the techniques of compression based on modification of the image transformation are considered. A reversible, linear transform (such as a Fourier transform) is used in transform codification to map the image into a set of coefficients that are then quantified and coded. For most natural images, many of the coefficients have small dimensions and can be quantified grossly (or completely discarded) with a small distortion in the image. A number of transformations can be used to convert image data, including the discrete Fourier transform (DFT).

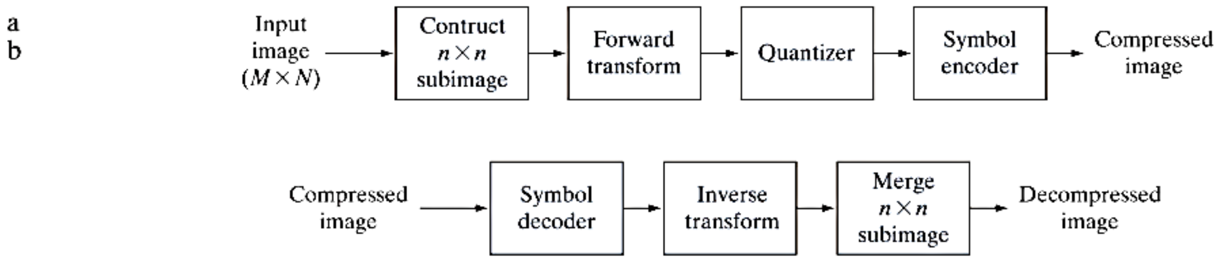


Fig. 6.8 A transform coding system: (a) encoder; (b) decoder.
Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

A standard transform coding system is shown in Figure 10. The decoder implements the inverse phase sequence of the encoder, which carries out four fairly straightforward operation (with exception of the quantification function): decomposition of the subimages, transformation, and quantification and coding. An image N X N is first divided up into sub-images of size n X n, then transformed into sub-image transforms (N/N) 2 of each size n X n. The aim of the process of transformation is to decorrelate the pixels of any subimage or to place as little information as possible in the transforming coefficients.

These coefficients have the least effect on the consistency of the subimage. The encryption process finishes the quantified coefficients by coding (usually with a variable length code). Any or all of these processing stages of transformative encoding may be adapted to local image contents or fixed for all sub-images, known as non-adaptive transform encodings.

6.10 Predictive Coding

Lossless Predictive Coding:

The error-free method of compression does not require an image to be decomposed into a group of bit planes. The technique, also known as loss-predictive coding, consists of removing the redundancy of interpixel pixel by collecting the latest information in each pixel and encoding the new information only. The new pixel information is defined as the difference between the actual and predicted pixel value**.**

The essential elements of the lossless predictive coding system are shown in Figure 8.1. The system is made up of an encoder and a decoder with the same predictor. The predictor creates the expected pixel value depend on a set of previous inputs, when each pixel of the input image, denoting fn is introduced to the encoder. The predictor's output then is rounded to the nearest integer, denoted $\hat{f}(n)$, and used to form a differential or prediction error coding the following element in the compressed data stream using variable-length (by the symbol encoder).

$$e(n) = f(n) - \hat{f}(n)$$

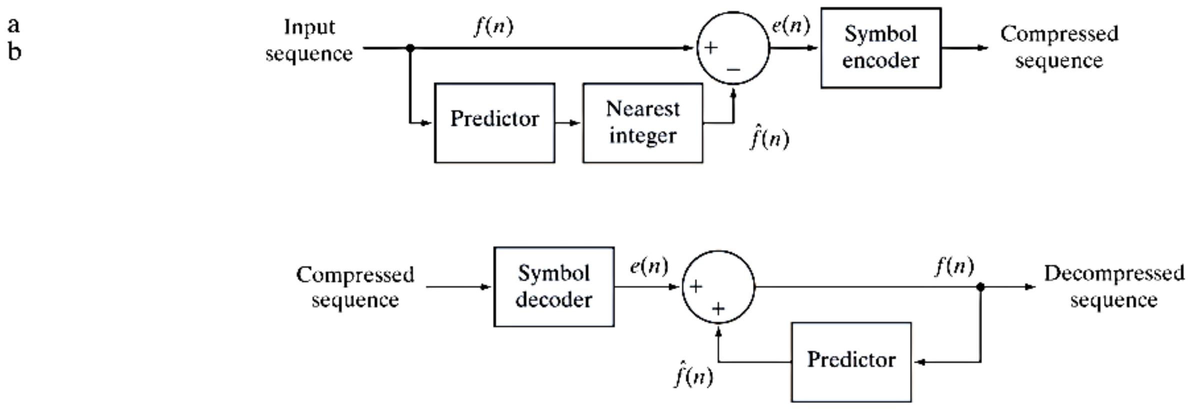


Fig. 6.9 A lossless predictive coding model: (a) encoder; (b) decoder

Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

The decoder of received variable-length code words is rebuilt from Fig. 8.1(b) and inverse operations are performed to decompress or recreate the original input sequence.

$$f(n) = e(n) + \hat{f}(n)$$

Various techniques for $\hat{f}(n)$ generation may be used local, global and adaptive. Furthermore, a linear combination of m of previous pixels forms in most cases a prediction. In other words,

$$\hat{f}(n) = round\left[\sum_{i=1}^{m} \alpha_i f(n-i)\right]$$

Where m is a linear predictor order, round is a function used to indicate an operation of the rounding or nearest integer, and αi, for i = 1,2,...,m are coefficients of prediction. The subscript n indexes predictor outputs according to their occurrence time in raster scan applications. In Eqns. above, the most explicit notations f (t), $\hat{f}(t)$ and e (t) could be replaced, where t represents time. In Eqns. In all cases, n shall is being used as an index on the spatial coordinates and/or frame number of an image (in a time sequence). For example, Eq. above can be written as 1-D linear predictive coding

$$\hat{f}(x,y) = round\left[\sum_{i=1}^{m} \alpha_i f(x, y-i)\right]$$

While each of the subscripted variable's function of spatial coordinates x and y are expressed explicitly. The Eq. says the linear 1-D prediction f(x, y) is the function alone on the current line of the previous pixels. In 2-D predictive coding, the prediction depends on the previous pixels in a top to bottom left-to-right image scan. In the 3D case, these pixels and prior pixels of earlier frames are the basis of this case. For the first m pixel of each line, the equal above cannot be evaluated, so these pixels must be coded by other methods (e.g. Huffman code) and considered as an overhead of the predictive coding process.

Lossy Predictive Coding:

within this types of coding, a quantizer is added to the lossless prediction models and the resulting balance between reconstruction precision and compression performance is evaluated. As Fig.9 illustrates, between the symbol encoder as well as the position during which the prediction error was created, the quantizer that captures the nearest integer of the error-free encoder would be inserted. It maps the prediction error into a limited range of $\hat{e}(n)$ outputs, which determine the compression and distortion associated with the lossy predictive coding.

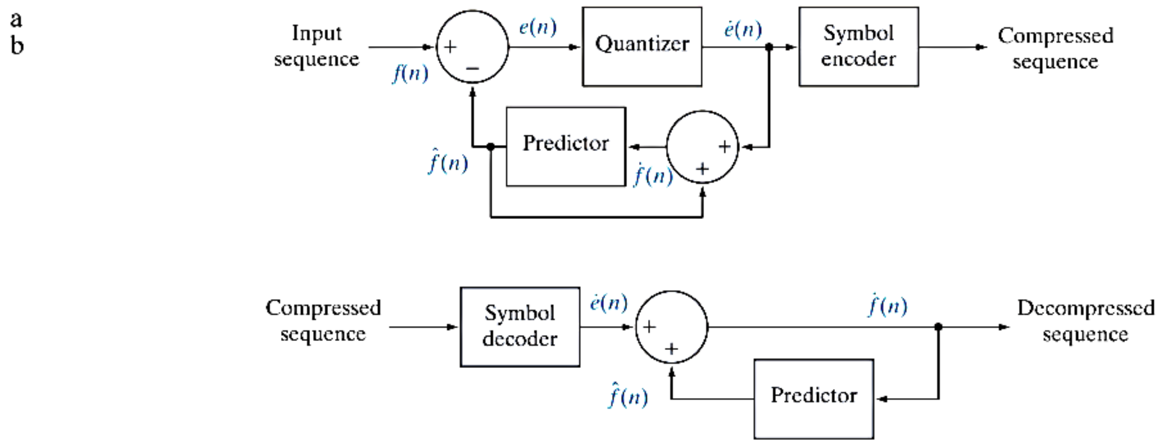


Fig. 6.10 A lossy predictive coding model: (a) encoder and (b) decoder. Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

The error-free figure encoder should be modified to accommodate the insertion of the quantization step, so that the encoder and decoder produce predictions equivalent. As shown in fig. 9 (a), the the lossy encoder's predictor is placed within a feedback loop, where its input (referred to as $\dot{f}(n)$) is produced in function of previous predictions and the corresponding quantified errors. The results are obtained.

$$\dot{f}(n) = \dot{e}(n) + \hat{f}(n)$$

This configuration of the closed loop avoids error building in the output of the decoder. Fig. 9(b) shows that the decoder output is also provided by the Eqn above.

Optimal predictors:

The optimal predictor used by most predictive coding applications limits the mean-square prediction error of the encoder . The notation $E\{.\}$denotes the statistical expectation operator.

$$E\{e^2(n)\} = E\left\{\left[f(n) - \hat{f}(n)\right]^2\right\}$$

subject to the constraint that

$$\dot{f}(n) = \dot{e}(n) + \hat{f}(n) \approx e(n) + \hat{f}(n) = f(n)$$

That is, the optimization criterion is chosen in order to minimize the mean-square prediction error, it is assumed that the quantization error is minimal $[\dot{e}(n) \approx e(n)]$ and the prediction is restricted to a linear combination of m of previous pixels. 1 These constraints are not necessary, but at the same time they greatly simplify analysis and reduce the computational complexity of the predictor. The resulting predictive coding method is called the differential pulse code modulation (DPCM)

6.11 Wavelet Coding:

The coding of the wavelet is based on the assumption that the coefficients of a transform that decorrelates an image's pixels can be coded better than the originals. If most important visual information are packaged into a small number of coefficients as the basis for the transformation, the remainder of the coefficients which be quantized coarsely or truncated to zero with a little distortion of the image.

The standard wavelet coding system is shown in Figure 11. For the encoding of a 2J X 2J image, a wavelet analysisΨ, — a minimum decomposition level (J - P) are chosen for the purpose of calculating the discrete transformation of the image wavelet. The fast wavelet transform can be used when the wavelet contains an additional scaling function⁚. In both instances, the computed transforms a significant part of the original image with a zero mean and laplacian distribution to horizontal, vertical and diagonal decomposition coefficients.

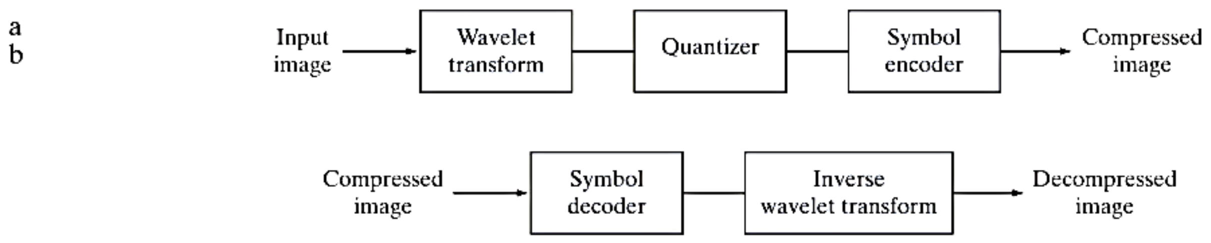


Fig. 6.11 A wavelet coding system: (a) encoder; (b) decoder.

Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

Since several of the measured coefficients have little visual detail, intercoefficient and coding redundancies can be reduced and quantified. In addition, quantization could be adapted to take advantage of any positional correlation over the P decomposition levels. One or more lossless coding techniques can be used in the final symbol coding step including run-length, Huffman coding, arithmetic and bit plane coding. Decoding is done by inverting encoding – with exception of quantization, which cannot be precisely reversed.

The major difference between the wavelet system and the transform coding system is that the subimage processing phases of the transform coder are not supported. Due to its computational efficiency as well as its intrinsically local existence (e.g. its base functions are restricted in duration), it is needless to subdivide the original image.

Wavelet Selection

In Fig. 6.11 all aspects of wavelet coding systems and performance are affected by the wavelets selected as a base for forward and reverse transformation. They have a direct impact on the complex computation of the transforms and, less directly, on the system's ability to recompress acceptable error images. The number of filter taps equal to the number of non-zero wavelet coefficients and scaling of the vector coefficients can be applied as a digital filter sequence if the transforming tap has a supplementary scaling function. The wavelet's capacity to pack information in a small number of transforming coefficients determines its performance in compression and reconstruction. Daubechies waves and biorthogonal waves are the most widely used expansion functions for wavelet-based compression. The latter allow useful analytical properties such as the number of disappearances that can be incorporated in the decomposition filters, while important synthesis properties such as smooth reconstruction are incorporated in the reconstruction filters.

The four discrete wavelengths in Figure 6 1.2 are shown. In Fig 6 1.2(a) Haar wavelets have been used as expansion or base functions, the simplest and only discontinuous wavelets considered in this example. Wavlets from Daubechies, one of the most widely used imagery wavelets, were employed in Fig. 6 1.2(b), and Daubechies waves with increased symmetry symmetry were extended to Fig. 6 1.2(c) The Cohen-Daubechies-Feauveau-wavelets in illustration of biorthogonal wavelets used in Fig. 6 1.2(d) are included. Like previous results, the subordinate structure was visible with a coefficient of intensity 128 corresponding to coefficient value 0 by scaling all detailed coefficients.

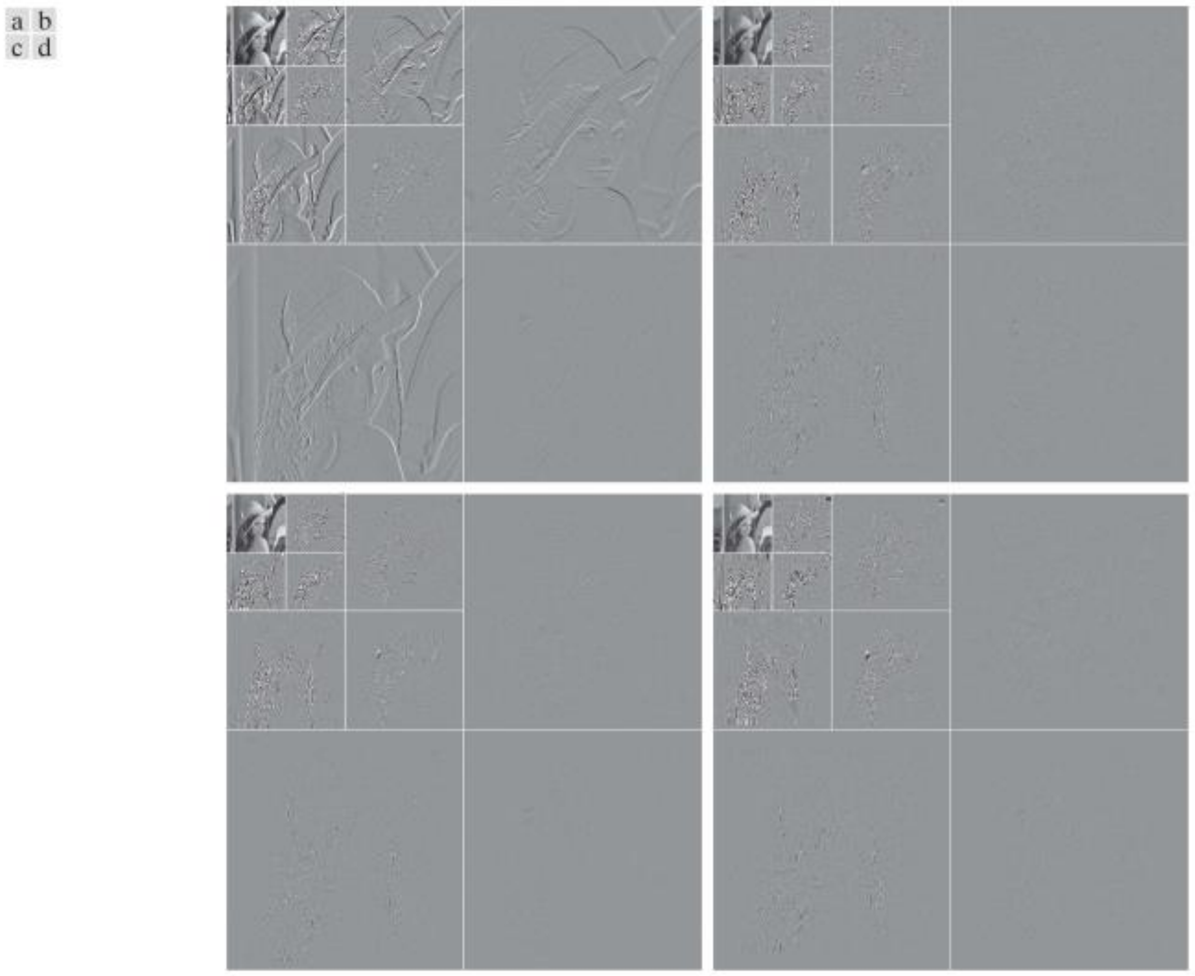


FIGURE 6.12

Three-scale wavelet transforms with respect to (a) Haar wavelets, (b) Daubechies wavelets, (c) symlets, and (d) Cohen-Daubechies-Feauveau biorthogonal wavelets.

Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

As illustrated in Table 6.2 , the number of operations required to compute the transforms in Fig. 6.12 grows from four to twenty-eight multiplications and adds each coefficient (for each decomposition level) as you move from Fig. 6.12(a) to (d). Each of the four transformations was computed using a fast wavelet transform formulation (i.e., filter bank). Notably, as computational complexity (i.e., the amount of filter taps) rises, so does the performance of information packing. When Haar wavelets are used and detail coefficients less than 1.5 are cancelled out, 33.8 percent of the entire transform is zeroed out. With more sophisticated biorthogonal wavelets, the percentage of zeroes coefficients increases to 42.1 percent, nearly doubling the possible compression.

TABLE 6.2

Wavelet	Filter Taps (Scaling + Wavelet)	Zeroed Coefficients
Haar	2+2	33.3%
Daubechies	8+8	40.9%
Symlet	8+8	41.2%
Biorthogonal	17+11	42.1%

Wavelet transform filter taps and zeroed coefficients when truncating the transforms in Fig. 8.43 below 1.5

6.12 Digital Image Watermarking

The techniques and protocols make it possible to distribute images (in the form of photographs or videos) via digital media and the Internet. Regrettably, the images distributed in this manner can be reproduced frequently and without error, jeopardizing the rights of their owners. Even when images are encrypted for distribution, they are rendered vulnerable upon decryption. To deter unauthorized duplication, one method is to embed one or more pieces of information, generally called to as a watermark, into highly vulnerable images in such a manner so the watermarks become indistinguishable from the images themselves. They defend the rights of their owners in a variety of ways as important aspects of the watermarked images, including the following:

1. Copyright identification. When an owner's rights are violated, watermarks can provide information that serves as proof of ownership.

2. Identification or fingerprinting of the user. Legal users' identities can be embedded in watermarks and used to track down the sources of unauthorized copies.

3. Authenticity determination. A watermark can serve as a guarantee that an image has not been altered, provided the watermark is designed to be destroyed upon image modification.

4. Automated monitoring. Watermarks can be tracked by systems that keep track of when and where images are used (for example, programmes that scan the Web for images embedded in Web sites). Monitoring is beneficial for collecting royalties and/or locating unauthorized users.

5. Copy protection. Watermarks can be used to set usage and copying restrictions on images (e.g., to DVD players)

In this section, we will discuss digital image watermarking, which is the technique of embedding data into an image in such a way that it may be utilized to make a assertion about the image. The methods outlined have no resemblance to the compression techniques discussed previously (although they do involve the coding of information). Indeed, watermarking and compression are diametrically opposed in several aspects. While compression aims to reduce the amount of data used to represent images, watermarking aims to enhance them by adding information and data (i.e., watermarks). Watermarks can be visible or invisible.

A visible watermark is a subimage or image that is opaque or semi-transparent and is superimposed on another image (i.e., the image being watermarked) to make it clear to the observer. Television networks frequently include visible watermarks (modelled after their logos) in the top or lower right-hand corner of the screen. As illustrated in the following example, visible watermarking is often accomplished in the spatial domain.

The image in Fig. 6.13(b) is the image's lower right quadrant with a scaled-down version of the watermark in Fig. 6.13 (a) superimposed on top. Using f_w the watermarked image as a starting point, we may represent it as a linear combination of the unmarked image f and the watermark w.

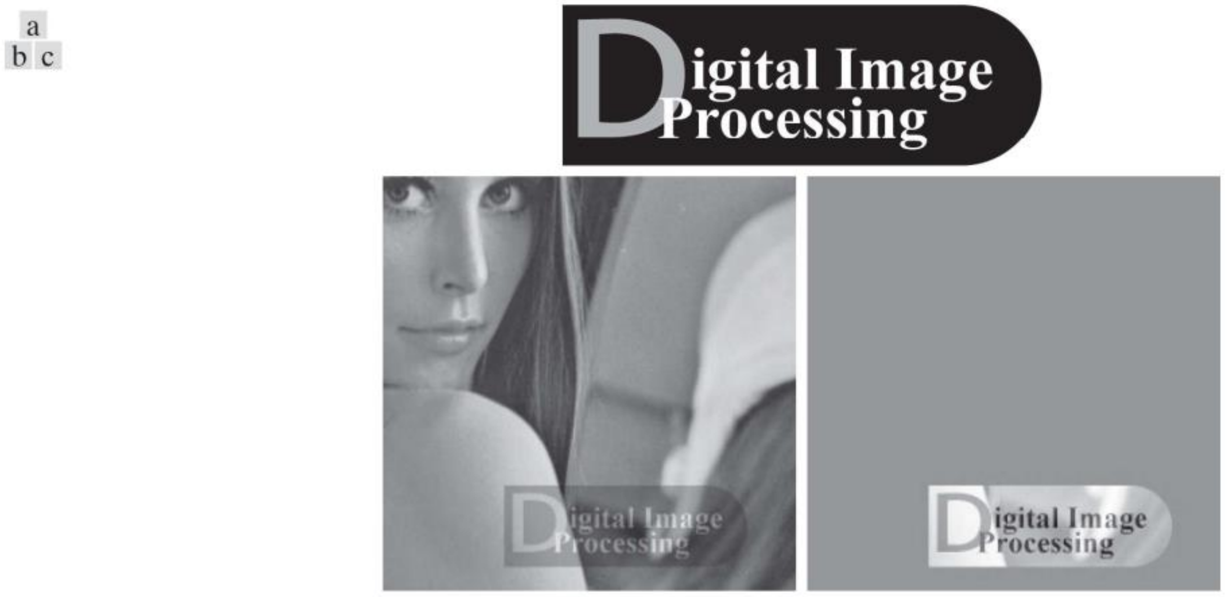


FIGURE 6.13

A simple visible watermark: (a) watermark; (b) the watermarked image; and (c) the difference between the watermarked image and the original (non-watermarked) image

Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

$$f_w = (1 - \alpha)f + \alpha w$$

Wherein constant determines the watermark's relative visibility to the underlying image. If α is 1, the watermark is transparent and fully obscures the underlying image. As the value approaches 0, more than just the underlying image is seen and less of the watermark. In general, $0<\alpha<1$, as shown in Fig. 6.13(b),$\alpha=0.3$. The computed difference (intensity-scaled) between the watermarked image in Fig. 6.13 (b) and the unmarked image is shown in Figure 6.13(c). Intensity 128, on the other hand, represents a difference of 0. It's worth noting that the underlying image is visible through the "semi transparent" watermark. Both Fig. 6.13(b) and the difference image in Fig. 6.13 (c) demonstrate this .

Invisible watermarks, in contrast to the visible watermark in the previous example, cannot be seen with the human eye. They are invisible yet recoverable with the use of a proper decoding algorithm. By introducing them as visually redundant information [information that the human eye ignores or cannot perceive], their invisibility is ensured. The example in Figure 6.14 (a) is simple. Due to the fact that the least significant bits of an 8-bit image have little effect on our perception of it, the watermark from Fig. 6.14 (a) was put or "hidden" in the image's two least significant bits. Using the notation presented previously, let us say

$$f_w = 4\left(\frac{f}{4}\right) + \frac{w}{64}$$

and conduct the computations using unsigned integer arithmetic. By dividing and multiplying by 4, the two least significant bits of f are set to 0, by dividing w by 64, the two most significant bits of w are shifted into the two least significant bit positions, and by adding the two results, the LSB watermarked image is generated. Note that in Fig. 6.14(a), the embedded watermark is not visible. However, by zeroing the image's most significant six bits and scaling the remaining values to the entire intensity range, the watermark can be retrieved as shown in Fig. 6.14. (b)

a b
c d

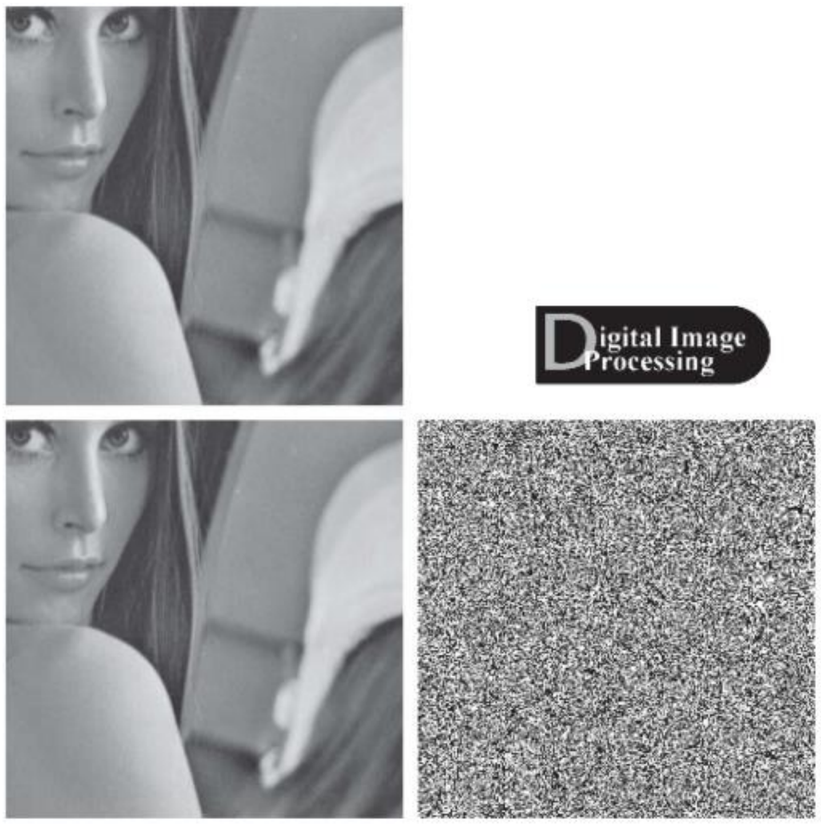


FIGURE 6.14 A simple invisible watermark: (a) watermarked image; (b) the extracted watermark; (c) the watermarked image after high quality JPEG compression and decompression; and (d) the extracted watermark from (c).

Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

A critical aspect of invisible watermarks is their resistance to removal attempts, both accidental and purposeful. Weak invisible watermarks are removed when the images in which they are embedded are modified. This is a useful quality in some applications, such as image authentication. As illustrated in Figures 6.14 (c) and (d), the LSB watermarked image in Figure 6.14 (a) has a delicate invisible watermark. The watermark is deleted if the image in (a) is compressed and decompressed using lossy JPEG. The outcome of compressing and decompressing Figure 6.14 (a) is shown in Figure 6.14 (c); the rms error is 2.1 bits. When we use the same procedure as in (b) to extract the watermark from this image, the outcome is incoherent [see Fig. 6.14 (d)]. While lossy compression and decompression saved the image's critical visual information, they obliterated the image's delicate watermark.
Robust invisible watermarks are meant to withstand image modification, whether accidental or purposeful. Lossy compression, linear and nonlinear filtering, cropping, rotation, and resampling are all examples of common accidental attacks. Intentional attacks can take the form of printing and rescanning, as well as the insertion of extra watermarks and/or noise. Naturally, it is superfluous to endure attacks that render the image unusable.

6.13 Unit End questions

1. What is data compression?
2. Write a short note on
   a) Coding Redundancy b) Interpixel Redundancy c) Psychovisual Redundancy
3. What is a Fidelity Criteria?
4. Explain the Huffman coding with an example.
5. Explain the Golomb Coding with an example.
6. Explain the Arithmetic Coding with an example.
7. Explain the LZW Coding with an example.
8. Explain the Run-length Coding with an example.
9. Explain the Symbol-based Coding with an example.
10. Write a short note on Block Transform Coding.
11. Write a short note on Predictive Coding.
12. Write a short note on Wavelet Coding.
13. Write a short note on Digital Image Watermarking.

6.14 Reference for further reading

1. Digital Image Processing Gonzalez and Woods Pearson/Prentice Hall Fourth 2018
2. Fundamentals of Digital Image Processing A K. Jain PHI
3. The Image Processing Handbook J. C. Russ CRC Fifth 2010

Chapter 7: Morphological Image Processing

7.0 Objective

Upon the completion of this chapter, the readers will know

- This section defines a number of important concepts related to the subject of mathematical morphology and how the method is extensively applied in digital image processing.
- A variety of tools have been used for binary image morphology, incorporating erosion, dilation, opening, closing, and the techniques can also be combined them to generate more complex tools.
- Capable of developing binary image morphology based algorithms for the performance of tasks including morphological smoothing, edge-detection and skeletonisation
- Learn how binary image morphology to gray scale images can be expanded.
- Will be able to develop grey scale image processing algorithms for tasks like textural segmentation, granulometry, gray-scale gradient computing and others

7.1 Preliminaries

Morphological image processing is a set of non-linear image processing relevant to the shape or morphology of the image features. As per Wikipedia, morphological operations only depend on the specific ordering of the values of the pixels, not the numerical values therein. Morphological operations may also be used in grayscale images such that the light transfer functions of the grayscale are uncertain and their absolute pixel values are either of no or minor concern.

Morphological techniques evaluate an image called a structural element with a small shape or template. The structuring element is placed anywhere in the image and compared with the corresponding pixel neighborhood. In some operations, the element is tested if it "fits" in the neighborhood, while others test if it "hits" or crosses it:

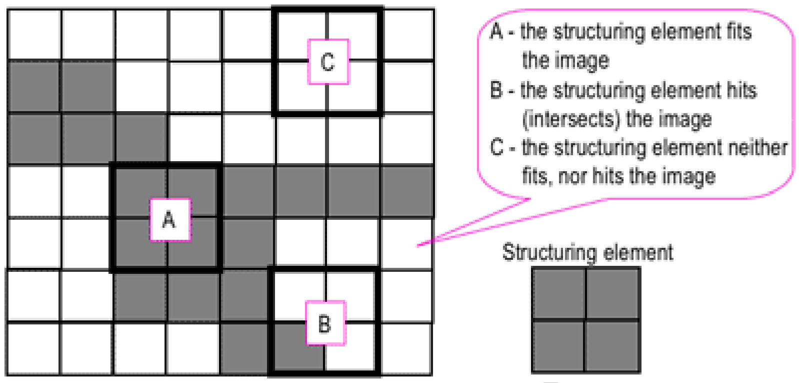

A - the structuring element fits the image
B - the structuring element hits (intersects) the image
C - the structuring element neither fits, nor hits the image
Structuring element
A
B
C

Figure 7.1 Probing of an image with a structuring element

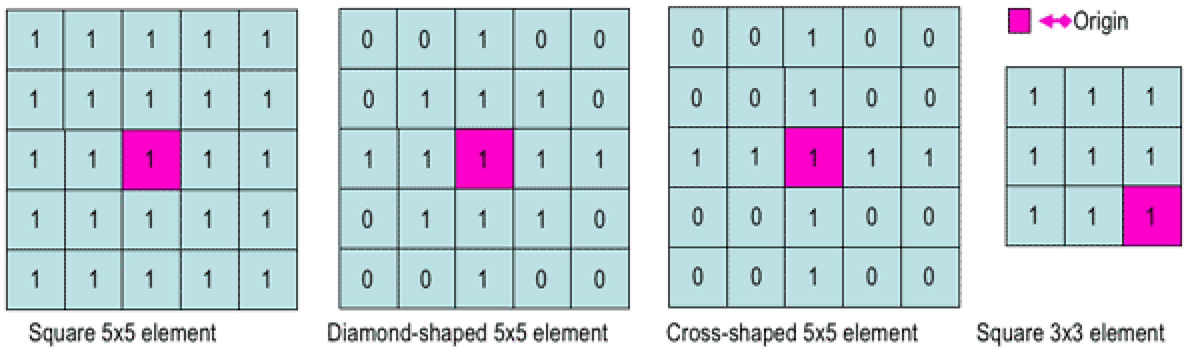
(**white and grey pixels have zero and non-zero values, respectively).**

(Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

A morphology operation on a binary image produces a new binary image with a pixel value not zero only, if the test at that location in the input image is successful.

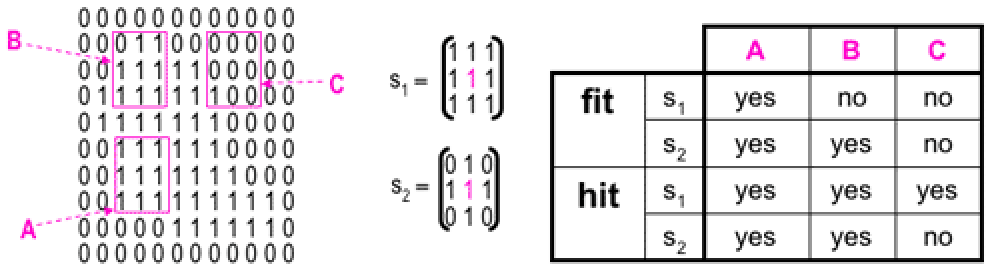
A small binary image is the structuring element, i.e. a small matrix of pixels, each with its own zero or one value:

- The sizes of the structuring element indicate the matrix dimensions.
- The pattern and zeroes specify the structure element shape.
- One of its pixels typically originates from a structuring element, but the origin can usually be outside the structuring element.



The odd dimensions of the structuring matrix and of the origin identified as the core of the matrix is a standard procedure. In morphological images, Structuring elements play the same role in linear image filtering as convolution kernels.

If a structuring element is positioned in a binary image, each pixel of the structuring element is associated with the corresponding pixel of the neighborhood. The structuring element should match the image if the corresponding image pixel is 1 for each of its pixels also set to 1. Likewise, if the corresponding image pixel is also 1. For at least one of their pixels set to 1, a structuring element is said to hit or intersect an image.



		A	B	C
fit	s_1	yes	no	no
	s_2	yes	yes	no
hit	s_1	yes	yes	yes
	s_2	yes	yes	no

Figure 7.2 Fitting and hitting of a binary image with structuring elements s_1 and s_2.

7.2 Erosion and dilation

Erosion

Erosion is the alternative to dilation. If dilation expands an image, erosion shrinks it. The structural element determines how the image is shrunk. Typically, the structural element is smaller than the image, measuring 3 x 3 size.

This results in a faster computation time as compared to using larger structural elements. Almost identical to dilation, erosion will move the structural element from left to right and top to bottom.

At the center position, denoted by the structuring element's centers, the procedure checks to see if there is complete overlap with the structuring element or not.

If no complete overlapping occurs, the center pixel given by the structural element's center will be set to white or 0. Assume that X represents the reference binary image and B represents the structuring element. The equation for erosion is as follows:

$$X \ominus B = \{z \mid \hat{B}_z \in X\}$$

According to the equation, the outcome element z is only evaluated if the structuring element is a subset of or equal to the binary picture X. Fig. b illustrates this process. Again, the white square represents O, whereas the black square represents 1.

At position •, the erosion process begins. There is no total overlap in this case, and hence the pixel at location • remains white.

After shifting the structural element to the right, the identical condition is seen. Because there is no complete overlapping at position u, the black square designated with • • will be converted white.

The structural element is then relocated further to the point shown by •• •. Here, we observe that the overlapping is complete, since all of the structuring element's black squares overlap with the image's black squares.

As a result, the image's structuring element's center will be black. The result is seen in Figure 7.3 b after the structuring element reaches the final pixel. Erosion is a thinning operator that causes an image to become smaller. By eroding an image, it is possible to eliminate narrow sections while thinning out wider ones.

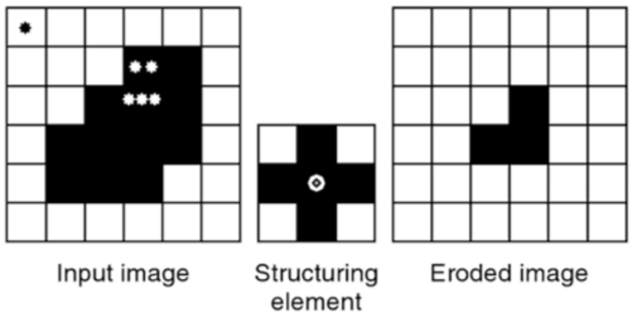


Figure 7.3 Erosion

(Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

Dilation

As the binary image is expanded from its original shape, dilation is defined as the technique of expanding a binary image from one shape to another. The structural element determines how the binary picture is expanded and how it is displayed.

This structuring element is smaller in size when compared to the image itself, and the size that is typically used for the structuring element is 3 × 3 pixels in size.

Dilation is similar to convolution in that the structuring element is mirrored and shifted left to right and top to bottom; at each shift, the process looks for any overlapping identical pixels between the structuring element and the binary image.

If there is an overlap, the pixels beneath the structuring element's center position will be set to 1 or black.

Assume that X represents the reference image and B represents the structuring element. Equation defines the dilation operation.

$$X \quad \oplus \quad B = \left\{ z \middle| \left[(\hat{B})_z \cap X \right] \in X \right\}$$

Where B is the rotation of the image B about the origin. The equation indicates that when the structuring element B dilates the image X, the outcome element z is that at least one element in B intersects with an element in X.

Whether this is the case, the position of the structural element within the image will be set to 'ON'. This procedure is depicted in Fig. 7.4 a. I is represented by the black square, and 0 is represented by the white square.

At first, the structuring element's center is aligned at position •. There is no overlap between the black squares of B and the black squares of X at this point; so, the square will remain white at position •.

After that, the structural element will be transferred to the right. At position **, one of B's black squares overlaps or intersects with X's black square.

As a result, the square at position • • will be turned to black. Consequently, the structural element B is shifted left to right and top to bottom on picture X to produce the dilated image shown in Fig. 7.4 a.

The dilation operator is a binary object enlargement operator. Dilation has a variety of applications, but the most common is bridging gaps in an image, as B expands the features of X.

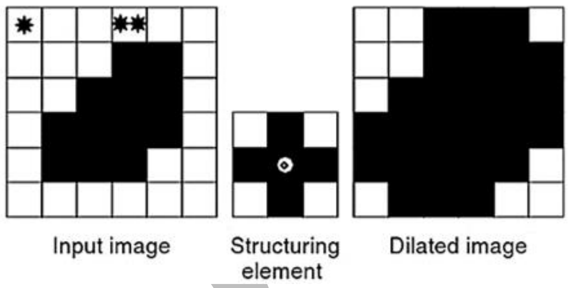


Figure 7.4 Dilation

(Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

7.3 Opening and Closing

Apart from the two major operations of erosion and dilation, there are two secondary operations that are critical in the processing of morphological images: opening and its opposite, closing. We are primarily concerned in opening, while the properties of closing are usually equivalent via complementation. While opening is defined in terms of erosion and dilation, it has a more geometric formulation in regards of structuring element fits, which serves as the basis for its implementation.

Opening

The opening of image A by image B is denoted by ∘ and is defined as the result of erosion and dilation by

$$A \circ B = (A \ominus B) \oplus B$$

The functional symbol for opening is $\gamma B(A)$. To illustrate the role of opening in processing, we use the following corresponding formulation:

$$A \circ B = \bigcup \{B_x : B_x \subset A\}$$

The opening is determined in this case by union all translations of the structural element that fit within the input image. Each fit is identified, and the opening is determined by adding the translations of the structural elements to each identified location. Indeed, this is exactly what eroding and eventually dilating refers to.

In Fig. 7.5, a rectangle is eroded and then dilated by a disks to demonstrate how opening is expressed as erosion followed by dilation. Additionally, the fitting effect can be discerned: opening the rectangle has resulted in it being rounded from the inside, this rounding being caused by the method in which the disks was "rolled around" inside the rectangle in order to establish a union of the fits. If the structural element had been a small square with a horizontal base, no rounding occurred and the opened image remained identical to the original.

In Fig. 7.5, we have two instances of opening. By opening with a disc, you create a filter that smooth from the inside out; in other words, it rounds corners that extend into the background. With a square structural element, the impression is significantly different.

Rather than viewing the opened image as the processing's final output, we might use an alternative approach. Consider subtracting the opening from the input image using set theory. This operator is referred to as the opening top-hat:

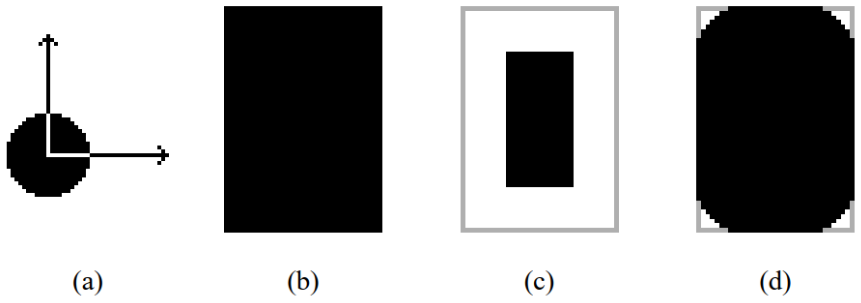


Figure 7.5 (a) Structuring element, (b) input image, (c) erosion, (d) opening.

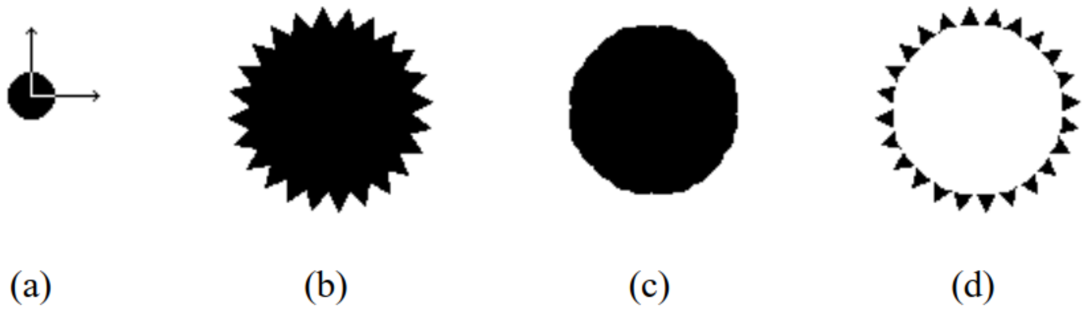


Figure 7.6. (a) Structuring element, (b) input image, (c) opening, (d) opening top-hat.

(Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

The opening top-hat in Fig. 7.5 comprises of protruding input-image corners into the background and can be used for recognition purposes. Figure 7.6 illustrates another application of the opening top-hat to detect gear teeth. Although the disk is frequently used because its shape effect is rotationally unchangeable, there are numerous circumstances when it is advantageous to use other types of structuring elements.

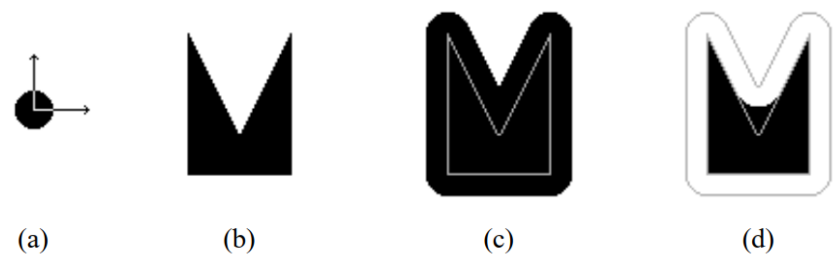


Figure 7.7 (a) Structuring element, (b) input image, (c) dilation, (d) closing

(Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

Closing

Closing is a complementary procedure to opening, which is described as a dilation followed by an erosion. The closing of A by B is indicated by $A \cdot$ B and defined by

$$A \cdot B = (A \oplus B) \ominus B$$

In functional notation, closing is also denoted by$\emptyset B(A)$. Closing is depicted in Figure 7.7. The effect is visible in the way the closing has been filtered from the outside, smoothing only the protruding corners into the image.

Because closing is the dual operator of the opening,

$$A \cdot B = (A^C \circ \breve{B})^C$$

Because closing is synonymous with opening, opening is synonymous with closing: yields that complement one another

$$A \circ B = (A^C \ \cdot \breve{B})^C$$

How the structuring element from the closing is reflected. If the object is a disc or any other symmetrical shape, reflection is irrelevant. We might use duality in conjunction with the union formulation of opening, thus fitting, or "rolling the ball," around the image's perimeter. With the use of an asymmetrical structuring element, Figure 7.8 depicts the duality between open and close.

Due to the fact that the closing contains the input image, subtracting the input image from the closing yields the closing top-hat operator:

$$A \,\hat{\cdot}\, B = (A \cdot B) - A$$

Figure 7.9 illustrates the closing and the closing top-hat in the application. A shape's convex hull is approximated by closing it with a large disk, and its convex hull inadequacies are approximated by closing it with a top-hat. Due to their strong discriminant property, these defects are frequently exploited in character recognition.

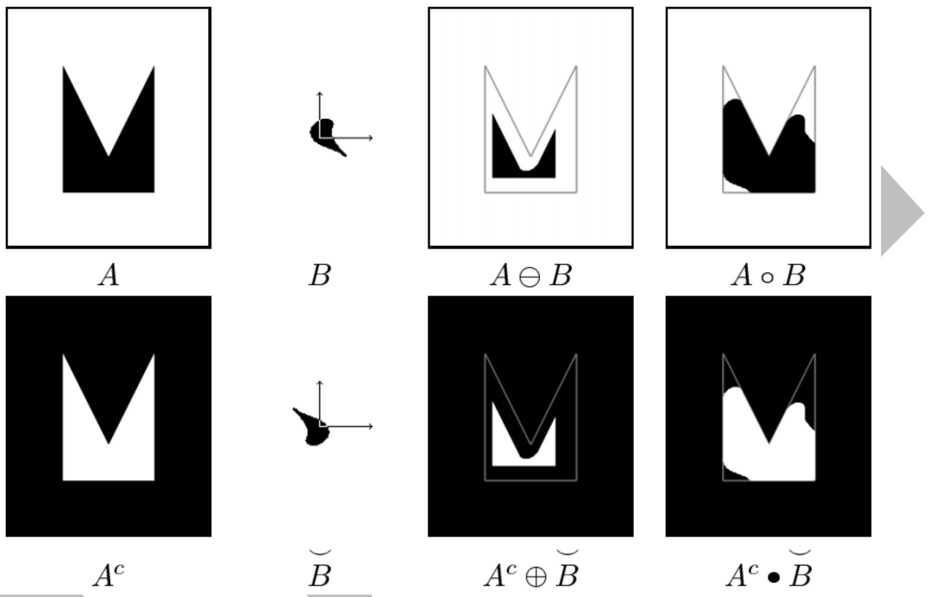


Figure 7.8 Duality between open and close with a nonsymmetrical structuring element.

(Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

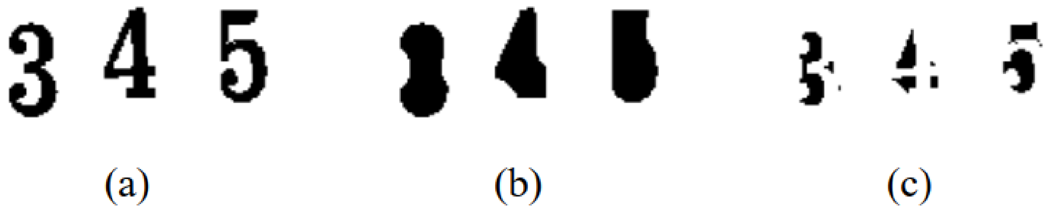


Figure 7.9 (a) Input image, (b) closing by a disk, (d) closing top-hat.

(Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

Consider the image and structural element, with the opening and closing represented. Opening, as a filter, has cleaned the boundary by removing minor extrusions; however, it has done so much more precisely than erosion, with the result that the opened image is a far more accurate duplicate of the original than the eroded image. Similar remarks apply to the closing, with the exception of minor invasions being filled. Notably, while the origin's position in relation to the structuring element influences both erosion and dilation, it has no effect on opening and shutting.

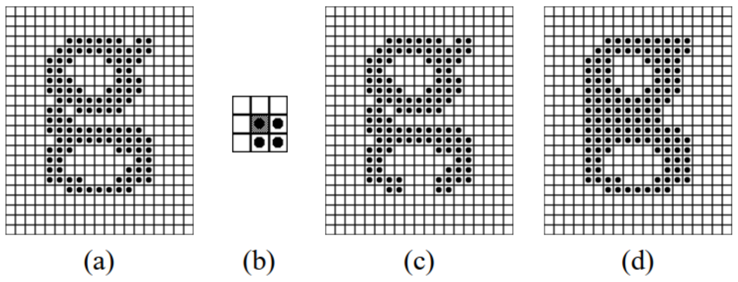


Figure 7.10 (a) Input image, (b) structuring element, (c) opening, (d) closing.

(Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

7.4 The Hit-or-Miss Transform

The hit-and-miss transform is a basic binary morphological operation that can be used to inspect an image for specific patterns of foreground and background pixels. It is, in fact, the fundamental operation of binary morphology, as it is the source of practically all other binary morphological operators. As is the case with other binary morphological operators, it accepts a binary image and a structuring element as input and outputs another binary image.

The hit-or-miss transformation of an image A by B is denoted by A $\circledast$B.

B is a structural element pair B=(B1,B2).. Instead of a single element,

B1: the collection of B elements associated with a certain object

B2 A collection of B elements that relate to the background

As follows is the definition of the hit-or-miss transform:

$$A \circledast B = (A \ominus B_1) \cap (A^C \ominus B_2)$$

This transform is important for locating all pixel configurations that correspond to the B_1 structure (i.e. a match) but not to the B_2 structure (i.e. a miss). As a result, the hit-or-miss transform is used to detect shapes.

Using the two structural elements B1 and B2, use the hit-or-miss transform to determine the locations of the following shape pixel configuration in the image below.

$$\begin{matrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{matrix}$$

Shap

$$\begin{matrix} 0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0&0&0&0 \\ 0&0&1&0&0&1&1&1&1&0&0 \\ 0&1&1&1&0&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&1&1&0&0 \\ 0&0&0&0&1&0&0&1&1&1&0 \\ 0&0&0&1&1&1&0&0&1&0&0 \\ 0&0&0&0&1&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0 \end{matrix}$$

Image A

$$\begin{matrix} & 1 & \\ 1 & \boxed{1} & 1 \\ & 1 & \end{matrix}$$

B_1

$$\begin{matrix} 1 & & 1 \\ & \boxed{} & \\ 1 & & 1 \end{matrix}$$

B_2

Solution:

$$A \ominus B_1 = \begin{matrix} 0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&1&0&0 \\ 0&0&0&0&1&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0 \end{matrix} \qquad A^c = \begin{matrix} 1&1&1&1&1&1&1&1&1&1&1 \\ 1&1&0&1&1&1&1&1&1&1&1 \\ 1&1&0&1&1&0&0&0&0&1&1 \\ 1&0&0&0&1&1&1&1&1&1&1 \\ 1&1&0&1&1&1&1&0&0&1&1 \\ 1&1&1&1&0&1&1&0&0&0&1 \\ 1&1&1&0&0&0&1&1&0&1&1 \\ 1&1&1&1&0&1&1&1&1&1&1 \\ 1&1&1&1&1&1&1&1&1&1&1 \end{matrix}$$

$$A^c \ominus B_2 = \begin{matrix} 1&0&1&0&1&1&1&1&1&1&1 \\ 1&0&1&0&0&0&0&0&0&0&1 \\ 0&0&0&0&0&1&1&1&1&1&1 \\ 1&0&1&0&0&0&0&0&0&0&1 \\ 0&0&0&0&0&0&0&0&0&0&0 \\ 1&0&0&0&0&0&0&0&0&0&1 \\ 1&1&1&0&1&0&0&0&0&0&0 \\ 1&1&0&0&0&0&0&0&1&0&1 \\ 1&1&1&0&1&0&1&1&1&1&1 \end{matrix} \qquad A \circledast B = \begin{matrix} 0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&\mathbf{1}&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&\mathbf{1}&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0 \end{matrix}$$

The figure below illustrates how the hit-or-miss transform is applied to the image from the preceding example.

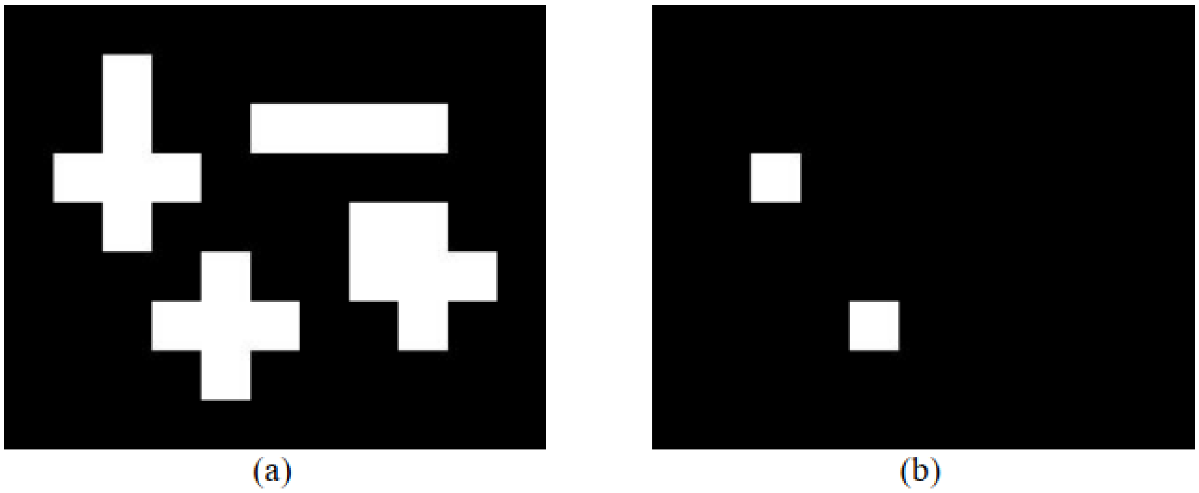


Figure 7.11 (a) Binary image. (b) Result of applying hit-or-miss transform.

(Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

7.5 Morphological Algorithms

Morphology's primary application is to extract image components that are useful for representing and describing shape. Boundary extraction, skeletonization (i.e. extracting the skeleton of an object), and thinning are all performed using morphological algorithms.

Boundary Extraction

The boundary of a set A, indicated by β(A), can be determined in the following manner:

$$\beta(A) - A - (A \ominus B)$$

where B denotes the structuring element.

The figure below illustrates how to extract an object's boundary from a binary image.

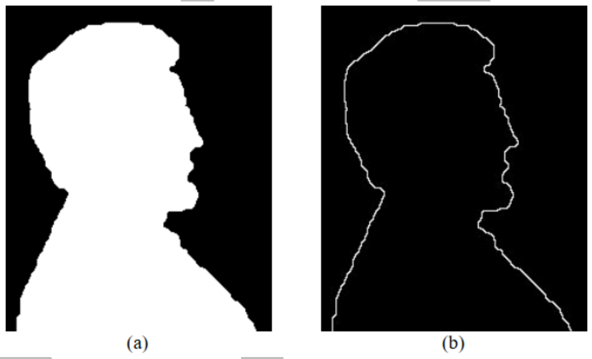


Figure 7.12 (a) Binary image. (b) Object boundary extracted using the previous equation and 3×3 square structuring element.

(Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

Due to the fact that the structuring element is 3x3 pixels in size, the resulting boundary is one pixel thick. Thus, utilizing the 5x5 structuring element results in a boundary that is between 2 and 3 pixels thick, as illustrated in the following figure.

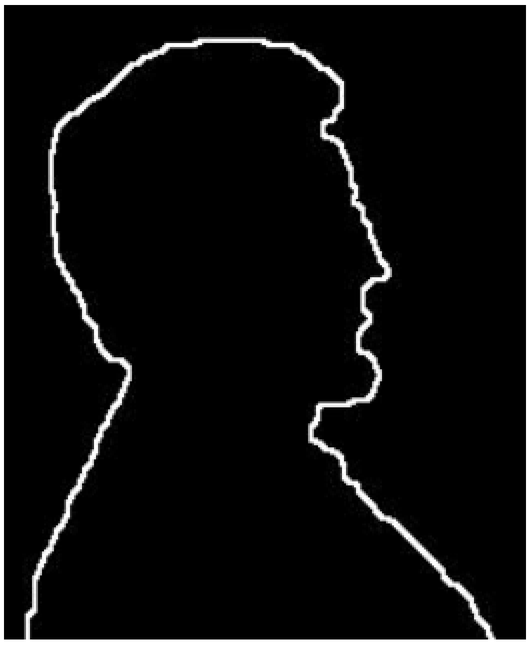


Figure 7.13 Object boundary extracted using 5×5 square structuring element

(Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

Thinning

Thinning is the process of reducing binary objects or shapes in an image to single-pixel width strokes. The definition of thinning a set A by a structuring element B is as follows:

$$A \otimes B = A - (A \circledast B) = A \cap (A \circledast B)^c$$

Because we are simply matching the pattern (shape) to the structuring elements, there is no need for a background operation in the hit-or-miss transform.

B is a sequence of structuring elements in this case:

$$\{B\} = \{B^1, B^2, B^3, \ldots, B^n\}$$

where Bi denotes Bi-1's rotation. Thus, the following can be expressed as the thinning equation:

$$A \otimes \{B\} = ((\ldots((A \otimes B^1) \otimes B^2)\ldots) \otimes B^n)$$

The entire procedure is repeated until no additional modifications are required. The following figure illustrates how to thinning the fingerprint ridges to a thickness of one pixel.

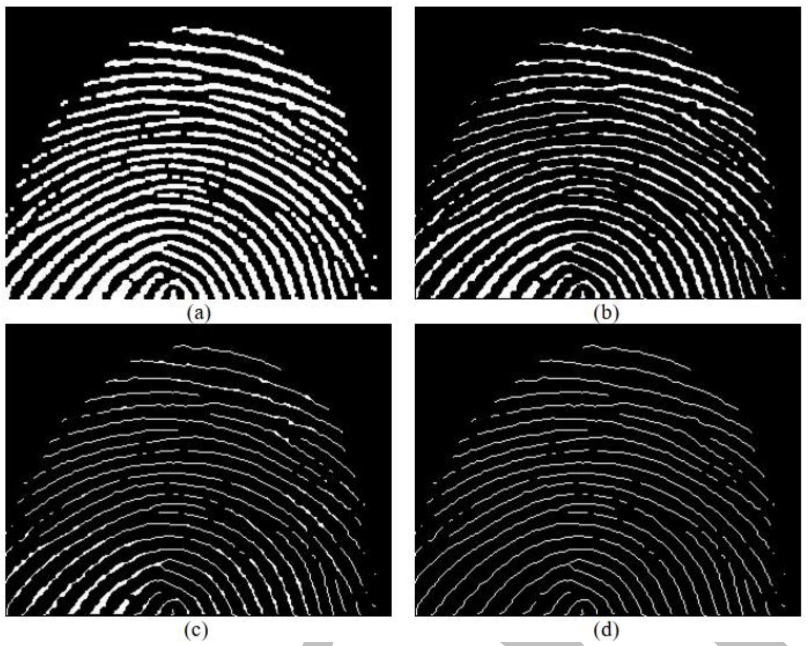


Figure 7.14 (a) Original fingerprint image. (b) Image thinned once. (c) Image thinned twice. (d) Image thinned until stability (no changes occur).

(Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

Figure 7.15 (a) illustrates a set of structural elements that are usually used for thinning (notice that B^i is equivalent B^{i-1} to rotated clockwise by 45^o and Fig. 7.15 (b) illustrates a set A that is to be thinned using the approach mentioned before. The result of thinning A with a single pass of B^1 to obtain A_1 depicted in Figure 7.15 (c). The result of thinning A_1 with B^2 is depicted in Figure 7.15 (c), and the outcomes of passes with the other structural components are depicted in Figures 7.15 (e) through (k) (there were no changes from $A_7$to A_8or from A_9 to A_{11}). Convergence occurred following the second pass of B^6 Figure 7.15 (l), which depicts the thinned result. Finally, Fig. 7.15 (m) illustrates the converted m-connectivity of the thinned set**.**

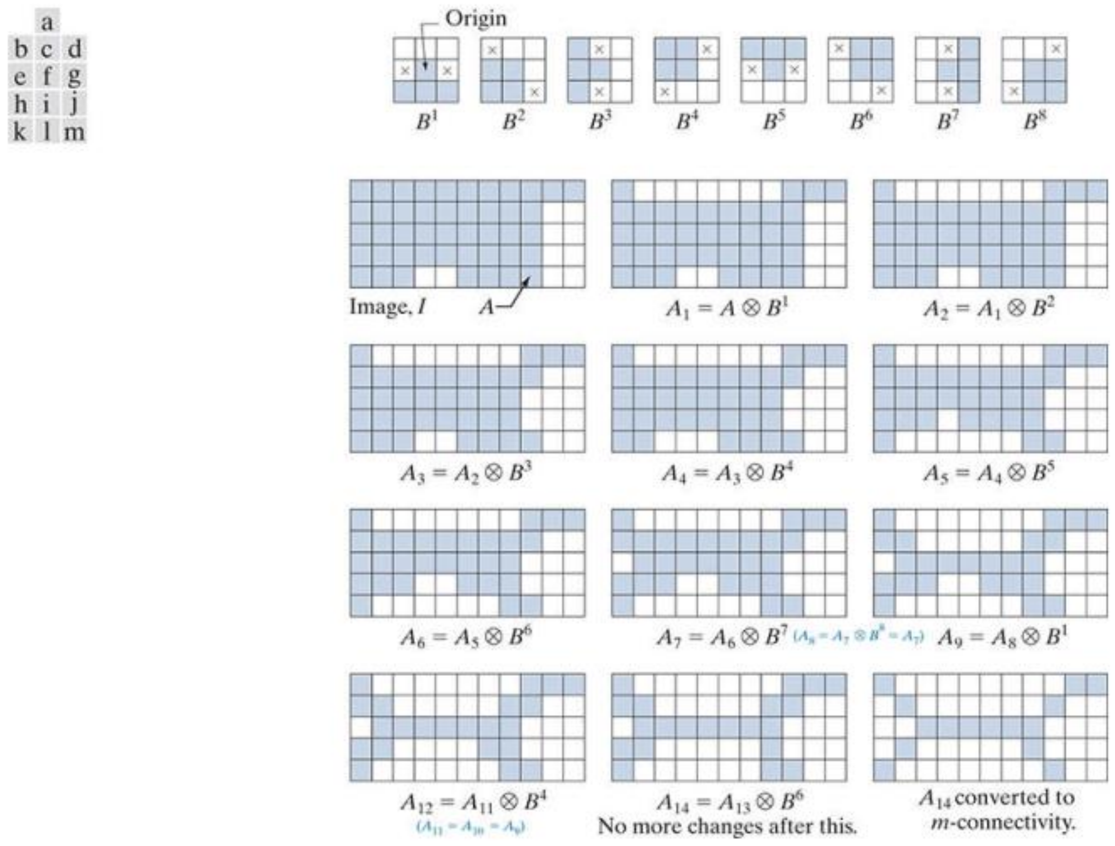

a
b c d
e f g
h i j
k l m
Origin
B^1 B^2 B^3 B^4 B^5 B^6 B^7 B^8
Image, I
A
$A_1 = A \otimes B^1$
$A_2 = A_1 \otimes B^2$
$A_3 = A_2 \otimes B^3$
$A_4 = A_3 \otimes B^4$
$A_5 = A_4 \otimes B^5$
$A_6 = A_5 \otimes B^6$
$A_7 = A_6 \otimes B^7$
$(A_8 = A_7 \otimes B^8 = A_7)$
$A_9 = A_8 \otimes B^1$
$A_{12} = A_{11} \otimes B^4$
$(A_{11} = A_{10} = A_9)$
$A_{14} = A_{13} \otimes B^6$
No more changes after this.
A_{14} converted to m-connectivity.

FIGURE 7.15 (a) Structuring elements. (b) Set A. (c) Result of thinning A B^1 with (shaded). (d) Result of thinning with A_1 and B_2(e)–(i) Results of thinning with the next six SEs. (There was no change between A_7andA_8 (j)–(k) Result of using the first four elements again. (l) Result after convergence. (m) Result converted to m-connectivity.

(Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

Thickening

Thickening is the morphological polar opposite of thinning, defined by the expression

$$A \odot B = A \cup (A \circledast B)$$

where B is a thickening structuring element. As with thinning, thickening is a sequential process:

$$A \odot \{B\} = ((..((A \odot B^1) \odot B^2)..) \odot B^n)$$

The structural components utilised for thickening are identical to those illustrated in Fig. **7.16** (a), except that both 1's and 0's have been changed. However, in practise, a separate algorithm for thickening is rarely utilised. Rather than that, the conventional approach is to thin the background of the set in question and then to complement the outcome. In other words, to thicken a set A, we first form A^c a thin A^c set and then complement it to achieve the thickened set A. This approach is illustrated in Figure 7.16. As previously stated, we display only set A and picture I, not the padded version of image I.

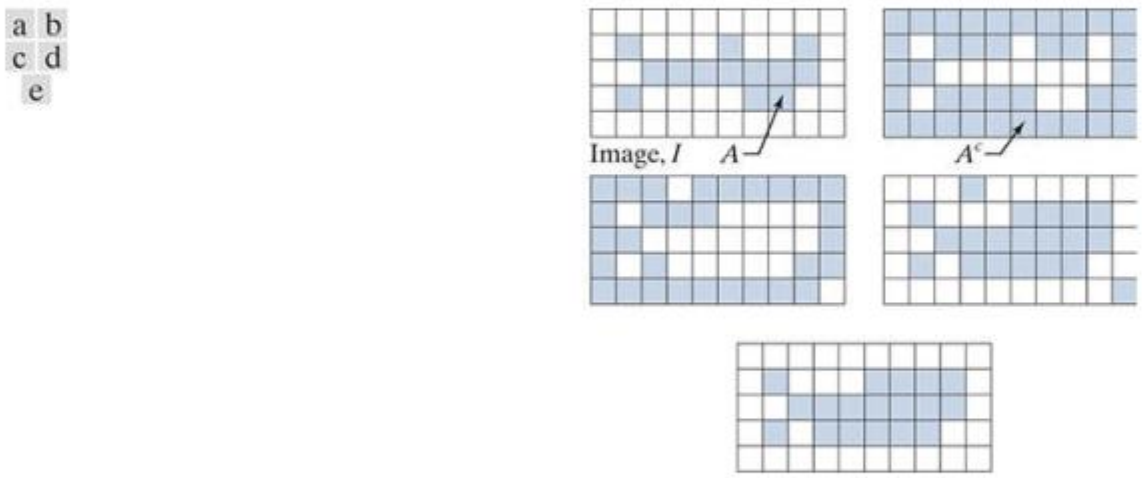


FIGURE 7.16 (a) Set A. (b) Complement of A. (c) Result of thinning the complement. (d Thickened set obtained by complementing (c). (e) Final result, with no disconnected points.

(Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

Convex hull

If the straight line segment connecting any two points in A lies entirely within A, the set is called to be convex. The smallest convex set containing an arbitrary set S is called the convex hull H. The difference between H and S is referred to as the convex deficiency of S. Convex hull and convex deficient are useful for describing objects.

Assume that B_i, i=1,2,3,4 are the four structural elements. The approach entails implementing the following equation:

$$X_k^i = (X_{k-i} \otimes B^i) \cup A \qquad i = 1,2,3,4 \;\; k = 1,2,3,4$$

with $X_0 = A$

When the procedure converges ($X_k^i = X_k^i - 1$), we let $D_i = X_k^i$. The convex hull of A is then

$$C(A) = \bigcup_{i-1}^{4} D_i$$

As an outcome, the procedure entails repeatedly applying the hit-or-miss transform to A with B1; once no further changes take place, the union with A is conducted and the outcome is denoted by D. Repeat the technique with B2 (applied to A) until no additional changes occur, and so forth... The convex hull of A is formed by the union of the four resultant Ds.

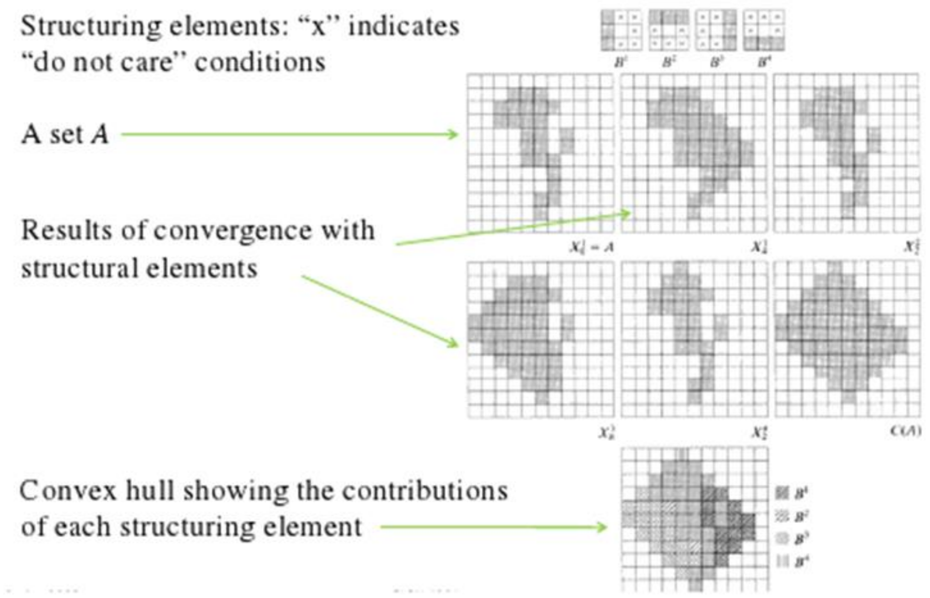

Structuring elements: "x" indicates "do not care" conditions
A set A
Results of convergence with structural elements
Convex hull showing the contributions of each structuring element
B^1
B^2
B^3
B^4
$X_0^1 = A$
$C(A)$

Figure 7.17 Convex hull

(Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

One disadvantage of the approach is that the convex hull can become larger than the minimum dimensions necessary to ensure convexity. One straightforward way to mitigate this effect is to constrain growth to the vertical and horizontal dimensions of the initial collection.

Hole filling

A hole is defined as a region of background pixels that is encircled by a connected border of foreground pixels.

Assume that A is a set containing eight connected boundaries, each boundary covering a background region (a hole). The purpose is to fill all holes with ones given a point in each hole (for binary images). We begin by creating an array X_0 of zeros (of the same size as the array containing A), except the locations in X_0 that correspond to the given point in each hole, which are set to one. The following process then replaces all of the holes with ones:

$$X_k = \left(X_{k-1} \oplus B \right) \cap A^c \qquad k = 1, 2, 3, \ldots$$

$$X_k = (X_{k-1} \oplus B) \cap \ A^c \ \text{ k=1,2,3}$$

where A is the symmetric structuring element

If $X_k = X_{k-1}$, the algorithm stops at iteration step k. Thus, the set X_k contains all filled holes, while the union of X_k and A includes all filled holes and their boundaries.

If left unchecked, the dilatation could fill the entire region. However, the intersection with the complement of A at each step restricts the result to the region of interest. This is an illustration of how a morphological process might be conditioned in order to achieve a desired property. In the current context, it is referred to as a conditional dilation.

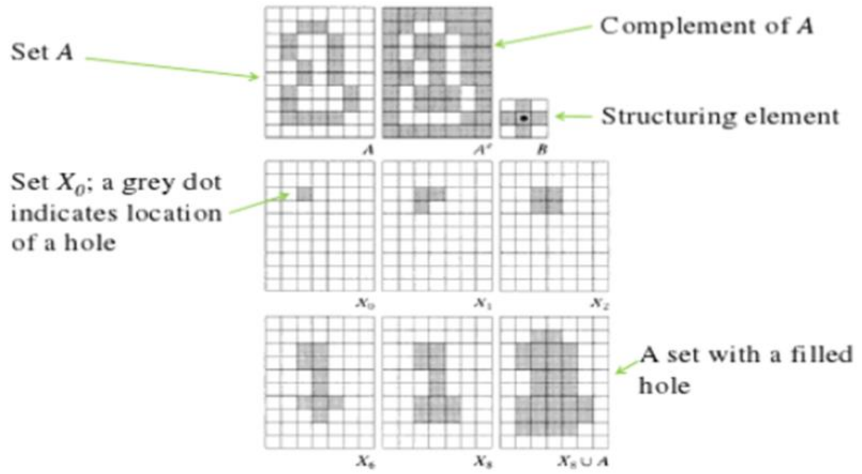
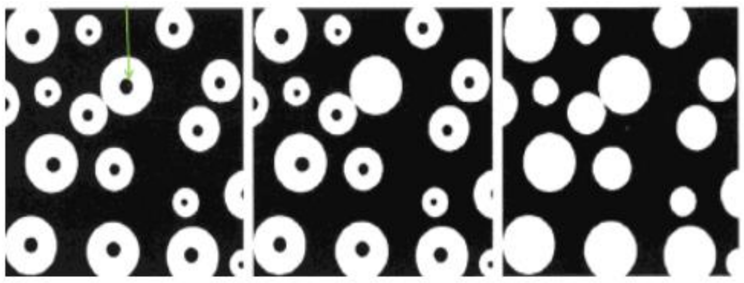


Figure 7.18 Hole filling

(Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

An image that could result from thresholding to 2 levels a scene containing polished spheres (ball bearings). Dark spots could be results of reflections. The objective is to eliminate reflections by hole filling…

A (white) point selected Result of filling that Result of filling all inside one sphere the spheres. A (white) point was selected as the result of filling all the spheres within one sphere.



Pruning

Pruning techniques are critical for thinning and skeletonizing algorithms, as these operations frequently leave parasitic elements that must be "cleaned up" in post-processing. We begin with a pruning problem and work our way up to a morphological solution. A popular approach to automated character recognition is to evaluate the shape of each character's skeleton. These skeletons are frequently defined by "spurs" (parasitic components). Spurs are formed during erosion as a result of inconsistencies in the strokes that comprise the characters. We propose a morphological technique for resolving this issue, presuming that the length of a parasitic component does not exceed a predefined number of pixels.

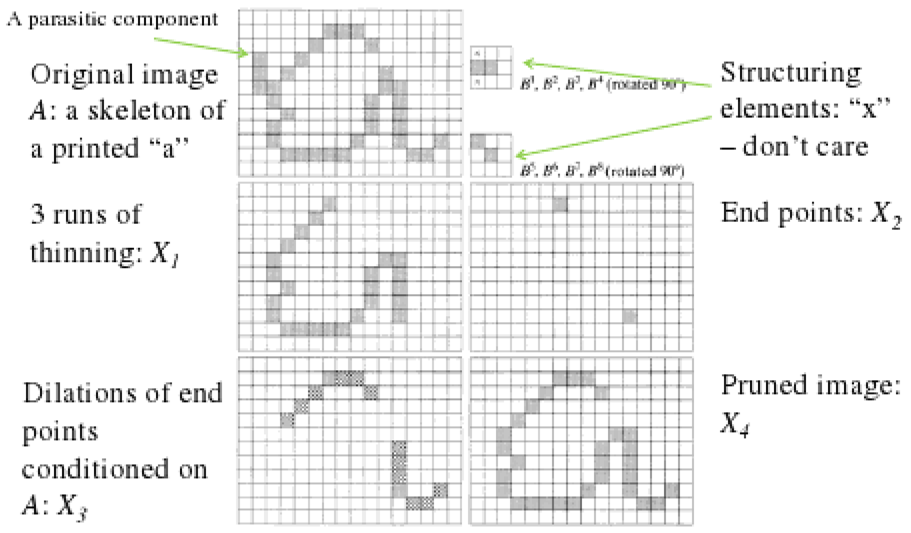

A parasitic component
Original image A: a skeleton of a printed "a"
B^1, B^2, B^3, B^4 (rotated 90°)
B^5, B^6, B^7, B^8 (rotated 90°)
Structuring elements: "x" – don't care
3 runs of thinning: X_1
End points: X_2
Dilations of end points conditioned on A: X_3
Pruned image: X_4

Figure 7.19 Pruning

(Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

The method is focused on gradually deleting a parasitic branch's termination point. Additionally, this reduces (or eliminates) the length of other branches in the character. In the absence of additional structural information, any branch with three or less pixels should be deleted. Thinning an input set A by the use of a sequence of structural components aimed at detecting.

Let

$$X_i = A \circ \{B\}$$

where {B} denotes the order of the structural elements. This sequence is made up of two distinct structures, each of which is rotated by 900 degrees to create a total of 8 pieces.

After applying the equation 3 times, the set X1 is obtained, and the character is "restored" to its original form but with the parasitic branches eliminated.

To begin, we create a set X_2 that contains all of the end points in X1:

$$X_2 = \bigcup_{k-1}^{8} (X_1 \otimes B^k)$$

where Bk denotes the identical structural parts. The following step is to dilate the end points three times using set A as a delimiter:

$$X_3 = (X_2 \oplus H) \cap A$$

where H is a three-dimensional three-dimensional structuring element composed of ones and the intersection with A as applied after each step. This type of conditional dilation prevents non-zero elements from appearing outside of the region of interest.

Finally, by combining X3 and X1, the desired result is obtained:

$$X_4 = X_1 \cup X_3$$

X3 occasionally includes the "tips" of some parasitic branches in more complex circumstances. This can occur when the branches' termination sites are close to the skeleton. Although these elements may be deleted in X1, they may reappear during the dilation because they are legitimate points in A. Because parasitic elements are typically brief in comparison to legitimate strokes, they are rarely picked up again. As they are disconnected regions, their detection and eradication are simple.

Skeletonization

Skeletonization is a technique for decreasing foreground regions in a binary image to a skeletal residue that retains the majority of the original region's extent and connectivity while discarding the majority of the original foreground pixels. To understand how this operates, consider that the foreground regions of the input binary image are composed of a uniformly slow-burning material. Simultaneously light flames along the region's boundary and observe the fire spread towards the interior. The fire will extinguish itself at spots when it crosses two distinct borders, and these points are referred to as the 'quench line'. This is the skeleton line. It is obvious from this definition that thinning results in the formation of a skeleton.

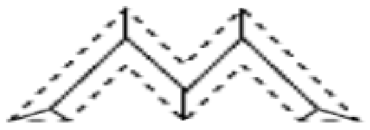


Figure 7.20: Skeletonization, Medial axis transform.

(Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

Another approach to see the skeleton is as the loci of the centres of bi-tangent circles that completely encompass the foreground region under consideration. This is illustrated in Figure **7.21** for a rectangular shape.

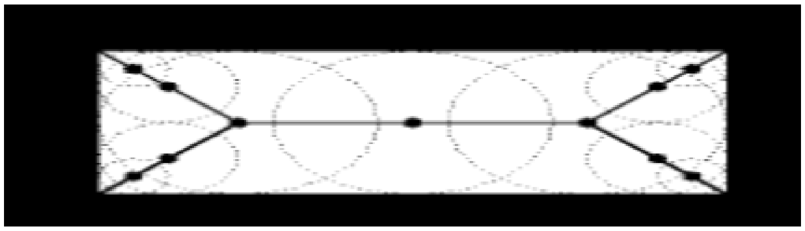


Figure 7.21: Skeleton of a rectangle defined in terms of bi-tangent circles.

(Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

Although the terms medial axis transform (MAT) and skeletonization are frequently used interchangeably, we shall make a subtle distinction between them. The skeleton is nothing more than a binary image of a simple skeleton. On the other hand, the MAT is a grayscale image in which each point on the skeleton has an intensity that corresponds to its distance from an object border.

A set A's skeleton S(A) can be thought of as:

a) If z is a point in S(A) and (D)z is the largest disk centred on z and included within A, no larger disks(not necessarily centred on z) containing (D)z and contained within A can be found. A maximal disks is denoted by the disk(D)z.

b) The disk (D)z makes two or more distinct contacts with the boundary of A.

A's skeleton can be described in terms of erosions and openings as follows:

$$S(A) = \bigcup_{k=0}^{k} s_k\,(A)$$

$$S_k(A) = (A \ominus B\,) - (A \ominus kB) \circ B$$

where B is a structural element that (A ⊖ kB) denotes k sequential erosions beginning with A; i. e., A is eroded first by B, followed by the result by B, and so on:

$$(A \ominus kB) = ((\ldots((A \ominus B\,) \ominus B\,) \ominus \ldots) \ominus B\,)$$

K is the last iterative step before A erodes to an empty set:

$$K = \max\{k \mid (A \ominus kB\,) \neq \emptyset\,\}$$

S(A) can be obtained as the union of the skeleton subsets $S_k(A)$, k = 0, 1, 2, …,K .A can be reconstructed from these subsets:

$$A = \bigcup_{k=0}^{k} (S_k\ (A) \oplus kB)$$

Where $(S_k\ (A) \oplus kB)$ denotes a series of k sequential dilations, beginning with S_k(A)

$$(S_k\ (A) \oplus kB) = ((\ldots((S_k\ (A) \oplus B) \oplus B) \oplus \ldots) \oplus B)$$

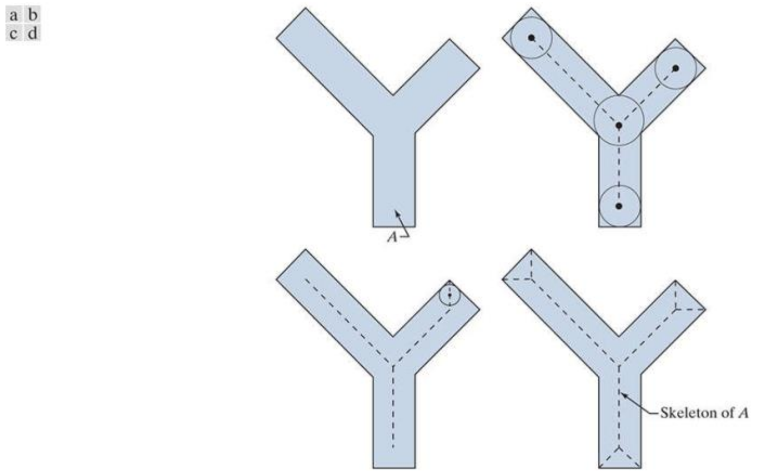


FIGURE 7.22 (a) Set A. (b) Various positions of maximum disks whose centers partially define the skeleton of A. (c) Another maximum disk, whose center defines a different segment of the skeleton of A. (d) Complete skeleton (dashed)

(Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

7.6 Morphological Reconstruction

Reconstruction is a morphological operation that entails the transformation of two images and a structural factor (instead of a single image and structuring element). The change begins with a single image, the marker. The second image, the mask, serves as a constraint on the transformation. Connectivity is defined by the structuring element used. We will use 8-connectivity (the default), which indicates that B is a 3 × 3 * matrix of 1s in the following discussion, with the center defined at coordinates (2, 2).

If G is the mask and F is the marker, the reconstruction of G from F, denoted $R_G(F)$, is defined by the following iterative procedure:

1. Initialize h_1 to be the marker image, F.

2. Create the structuring element: B = ones(3).

3. Repeat

$$h_{k+1} = (h_k \oplus B) \cap G$$

until $h_{k+1} = h_k$

4. $R_G(F) = h_{k+1}$

Marker F must be a subset of G:

$$F \subseteq G$$

The prior iterative approach is illustrated in Figure 10.21. While this iterative formulation is conceptually sound, there are far quicker computational approaches available.

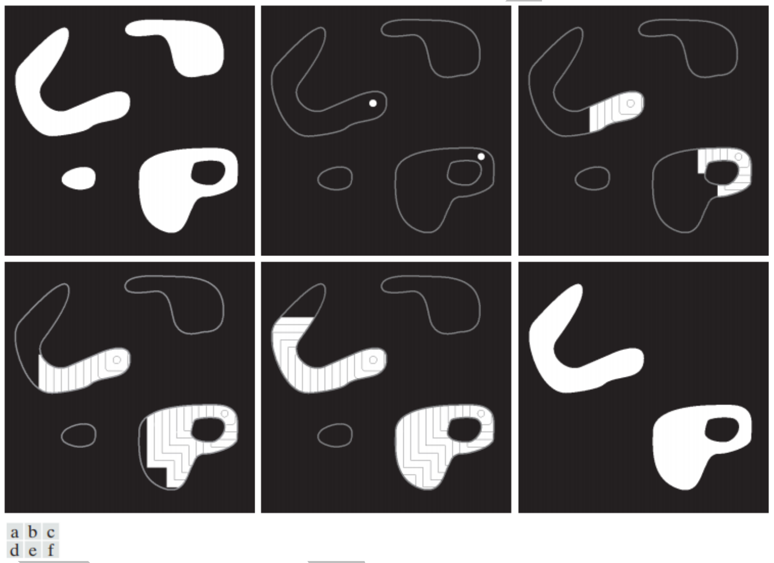


FIGURE 7.23 Morphological reconstruction. (a) Original image (the mask). (b) Marker image. (c)–(e) Intermediate result after 100, 200, and 300 iterations, respectively. (f) Final result. (The outlines of the objects in the mask image are superimposed on (b)–(e) as visual references.)

(Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

Geodesic Dilation and Erosion

Geodesic dilation and erosion are key procedures in the field of morphological reconstruction. Additionally, these processes add a new dimension to conventional dilation and erosion. They allow us to rebuild certain shapes within an image.

Nevertheless, we are still working with binary images in this case. If you're unfamiliar with binary images, they are images that contain either black or white pixels. In other words, we may divide these images into two groups: one foreground and one background.

The unique aspect of these two methods is that we employ two images instead of one for each. Likewise to the simple versions, we require an image to which we can apply dilation or erosion.

Along with this image, we'll need a masking image to control the resultant image's growth and shrinkage.

We must obtain the intersection of the masked picture and the dilated image using geodesic dilation. As previously stated, masking the image will constrain the growth of subsequent dilations.

Let denote the marker image and the mask image, F⊆G. The geodesic dilation of size 1 of the marker image with respect to the mask, denoted by $D_G^{(1)}(F)$

$$D_G^{(1)}(F) = (F \oplus B) \cap G$$

The geodesic dilation of size of the marker image with respect to, denoted by $D_G^{(n)}(F)$ is defined as

$$D_G^{(n)}(F) = D_G^{(1)}(F)(D_G^{(n-1)}(F))$$

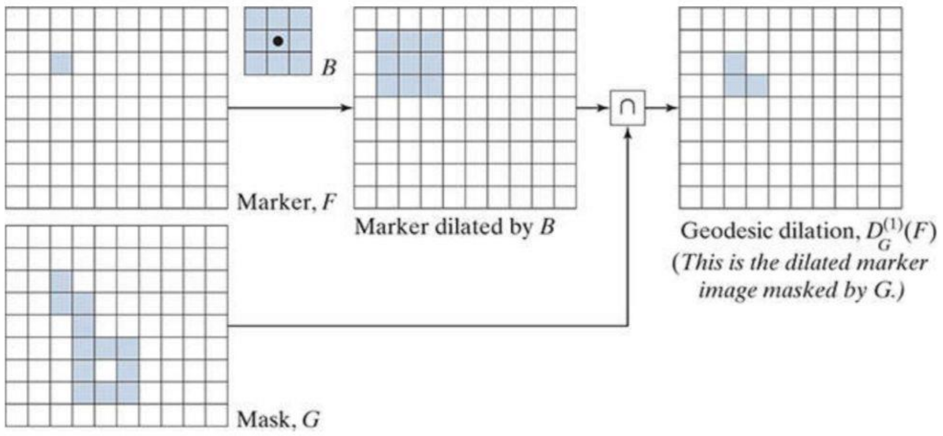

B
Marker, F
Marker dilated by B
∩
Geodesic dilation, $D_G^{(1)}(F)$
(This is the dilated marker image masked by G.)
Mask, G

FIGURE 7.24 Illustration of a geodesic dilation of size 1. Note that the marker image contains a point from the object in G. If continued, subsequent dilations and maskings would eventually result in the object contained in G.

(Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

The geodesic erosion of marker F's size 1 in relation to mask G is defined as

$$E_G^{(1)}(F) = (F \ominus B) \cup G$$

Where ∪ represents a predefined union (or logical OR operation). Geodesic erosion of F with regard to G of size n is defined as

$$E_G^{(n)}(F) = E_G^{(1)}(F)(E_G^{(n-1)}(F))$$

Where n≥1 is a positive number $and\ E_G^{(0)}(F) = F$ and at each step, the set union ensures that the geodesic erosion of an image keeps greater than or equal to its mask image. As the forms, Indicate, geodesic dilation and erosion are duals in terms of set complementation. Figure 7.25 illustrates a geodesic erosion of size one. The steps depicted are a direct application.

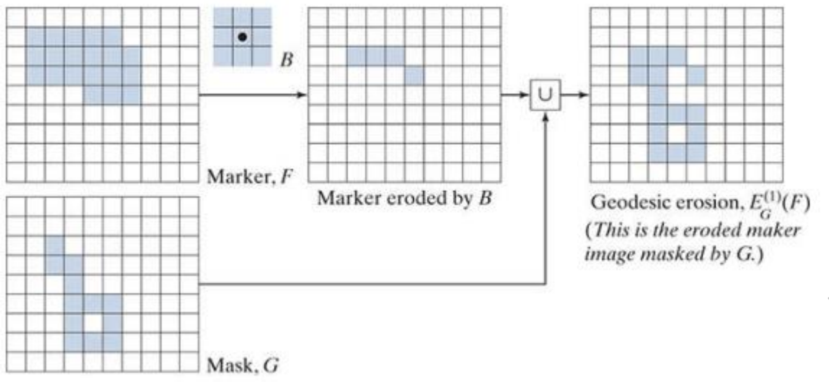

B
Marker, F
Marker eroded by B
∪
Geodesic erosion, $E_G^{(1)}(F)$
(This is the eroded maker image masked by G.)
Mask, G

FIGURE 7.25 Illustration of a geodesic erosion of size 1.

(Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

Geodesic dilation and erosion coincide over a finite number of repetitive steps as the mask restricts the marker image's propagation or shrinkage.

Morphological Reconstruction by Dilation and by Erosion

On the basis of previous notions, morphological reconstruction by dilation of a marker image F with respect to a mask image G, indicated $R_G^D(F)$, is defined as iterative geodesic dilation of F with respect to G until stability is obtained; that is,

$$R_G^D(F) = D_G^{(k)}(F)$$

with k such that

$$D_G^{(k)}(F) = D_G^{(k+1)}(F)$$

Reconstruction by dilation is seen in Figure 7.26. The process initiated in Figure 7.26 is continued in Figure 7.26 (a). After acquiring$D_G^{(1)}(F)$ this result, the next stage in reconstruction is to dilate it and then AND it with mask G $D_G^{(2)}(F)$, as shown in Fig. 7.26 (b). Following that, dilation of $D_G^{(2)}(F)$ and masking with G produces $D_G^{(3)}(F)$and so on. This approach is continued until the system reaches a state of stability. Adding one additional step to this example results $D_G^{(5)}(F) = D_G^{(6)}(F)$ in the image, morphologically reconstructed via dilation, being provided by $R_G^{(D)}(F) = D_G^{(5)}(F)$ as illustrated. As expected, the reconstructed image is identical to the mask.

a b c d
e f g h

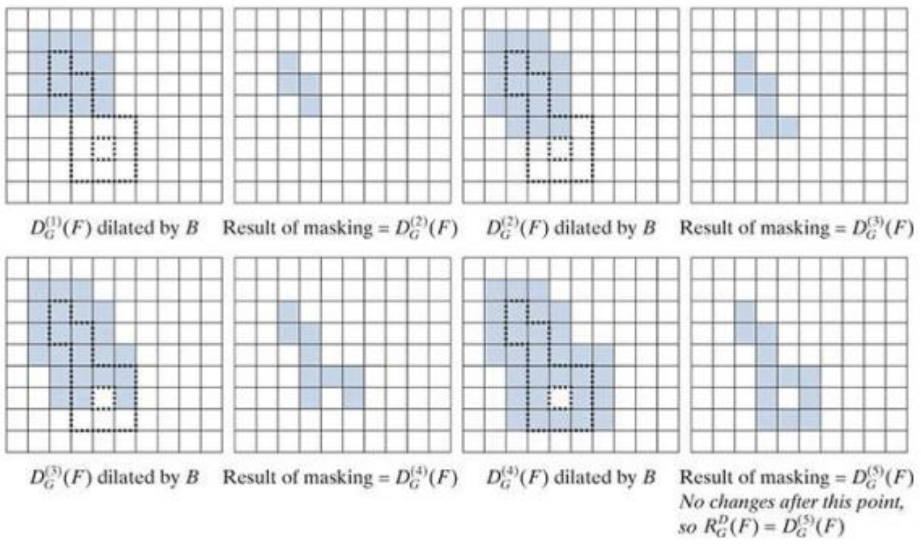

$D_G^{(1)}(F)$ dilated by B
Result of masking = $D_G^{(2)}(F)$
$D_G^{(2)}(F)$ dilated by B
Result of masking = $D_G^{(3)}(F)$
$D_G^{(3)}(F)$ dilated by B
Result of masking = $D_G^{(4)}(F)$
$D_G^{(4)}(F)$ dilated by B
Result of masking = $D_G^{(5)}(F)$
No changes after this point, so $R_G^D(F) = D_G^{(5)}(F)$

FIGURE 7.26 Illustration of morphological reconstruction by dilation. Sets $D_G^{(1)}(F)$,G, B and F are from Fig. 7.24 . The mask (G) is shown dotted for reference.

(Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

Similarly, the morphological reconstruction by erosion of a marker image F with regard to a mask image G, indicated $R_G^{(D)}(F)$, is specified as the iterative geodesic erosion of F with respect to G; that is,

$$R_G^E(F) = E_G^{(k)}(F)$$

with k such that

$$E_G^{(k)}(F) = E_G^{(k+1)}(F)$$

Create a figure similar to Fig 7.26a.m. for erosion-based morphological reconstruction. In terms of set complementation, reconstruction through dilation and erosion are duals.

Opening by Reconstruction

In morphological opening $(A \ominus B) \oplus B$, the erosion operation eliminates objects smaller than structuring element B, and the dilation operation recovers the size and shape of the remaining objects. However, restoration precision is greatly dependent on the type of structural element and the form of the restored objects during the dilation operation. The opening by reconstruction method allows for a more comprehensive restoration of items following erosion. It is defined as the reconstruction of n erosions of F by B in relation to using geodesic dilation.

$$O_R^{(n)}(F) = R_F^{(D)}(F \ominus nB)$$

Where $F \ominus nB$ is a marker image and F is a mask image used in morphological reconstruction via dilation.

$R_F^D[(F \ominus nB)] = D_F^{(k)}[(F \ominus nB)]$ D represents geodesic dilation with k iterations until stability is reached, i.e.

$D_F^{(k)}[(F \ominus nB)] = D_F^{(k-1)}[(F \ominus nB)]$. Hence $D_F^{(1)}[(F \ominus nB)] = ([(F \ominus nB)] \oplus B) \cap F$ Because the mask image constrains the marker image within the growth region, the dilation operation on the marker image will not extend outside the mask image. The marker image is thus a subset of the mask image. $[(F \ominus nB)] \subseteq F$

The images below demonstrate a straightforward opening-by-reconstruction procedure for extracting vertical strokes from an input text image. Due to the fact that the original image was transformed from grayscale to binary, it contains a few distortions in some characters, resulting in identical characters having differing vertical lengths. The structural element in this example is an 8-pixel vertical line that is used during the erosion operation to locate things of interest. Additionally, morphological reconstruction by dilatation $R_F^D[(F \ominus nB)] = D_F^{(k)}[(F \ominus nB)]$ iterates k=9 times until the resulting image converges.

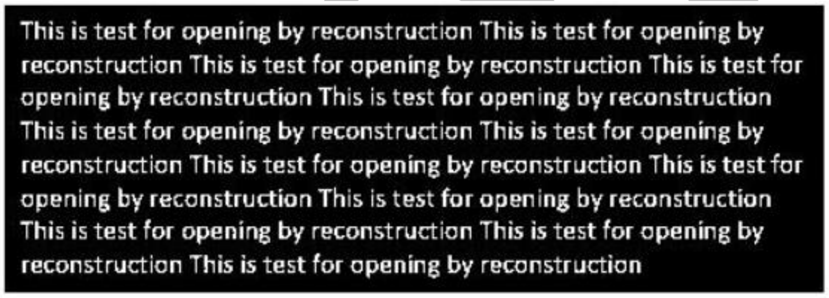

This is test for opening by reconstruction This is test for opening by reconstruction This is test for opening by reconstruction This is test for opening by reconstruction This is test for opening by reconstruction This is test for opening by reconstruction This is test for opening by reconstruction This is test for opening by reconstruction This is test for opening by reconstruction This is test for opening by reconstruction This is test for opening by reconstruction This is test for opening by reconstruction This is test for opening by reconstruction

FIGURE 7.27 Original image for opening by reconstruction

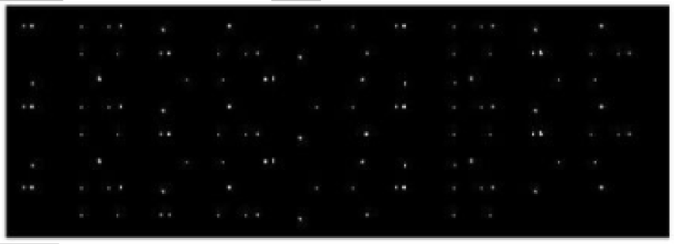


FIGURE 7.28 Marker image

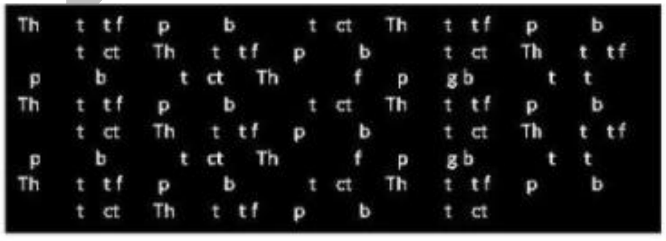


FIGURE 7.29 Result of opening by reconstruction

(Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

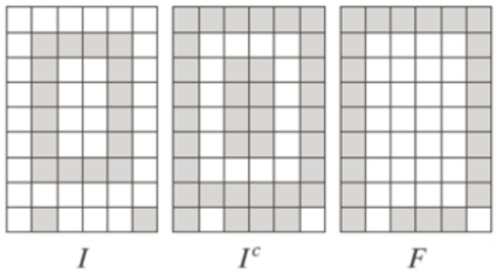
Automatic Algorithm for Filling Holes

We devised an algorithm for filling holes based on the knowledge of a hole's starting place. We present a fully automated approach for morphological reconstruction. Assume that I(x, y) is a binary image, and that we create a marker image F that is 0 everywhere except at the image boundary.

$$F(x,y) = \{ \begin{matrix} 1 - I(x,y) \\ 0 \quad otherwise \end{matrix} \; if\ (x,y) is\ on\ border\ of\ I$$

Only those dark pixels of I(x,y) that are adjacent to the boundary have a value of 1 in F(x,y).

The binary image that contains all regions (holes) is given by: $H = [R_{I^c}^{(D)}\ (F)]^c$



We seek to fill the hole by image I.

The complement surrounds the hole with a wall.

The marker image F is one at the border except for the original image's border pixels.

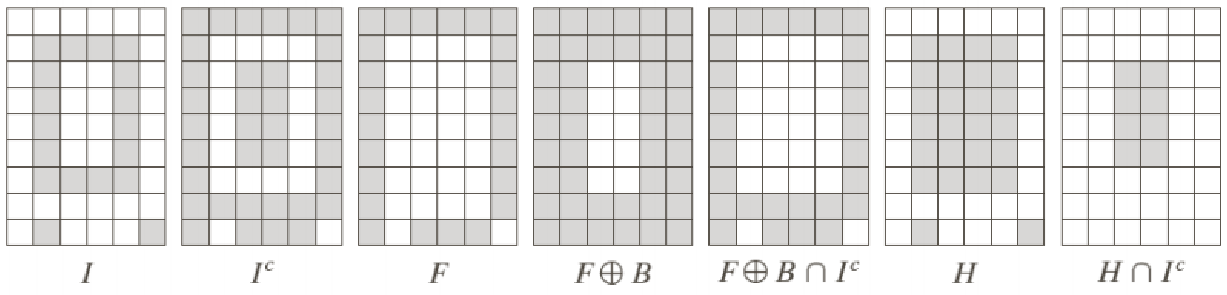


FIGURE 7.29.1Hole filling using morphological reconstruction.

(Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

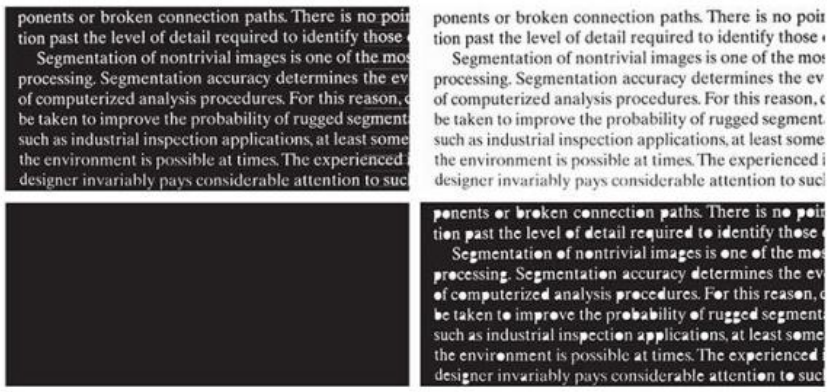
The marker F dilates inward starting at the boundary.

The complement acts as an AND mask, preventing any foreground pixels (including the wall) from altering throughout iterations.

Only the hole points are provided by the final procedure.

A more practical illustration is shown in Figure 7.30. The complement of the text image in Figure 7.30 (a) is shown in Figure 7.30 (b), and the marker image, F, produced is shown in Figure 7.30 (c). This image is entirely black with a white (1's) border, except for areas corresponding to 1's in the original image's border (the border values are difficult to identify visually at the magnification exhibited, and also since the page is practically white). Finally, Fig. 7.30 (d) illustrates the image after all holes have been filled.

a b
c d



ponents or broken connection paths. There is no poi
tion past the level of detail required to identify those
Segmentation of nontrivial images is one of the mos
processing. Segmentation accuracy determines the ev
of computerized analysis procedures. For this reason, c
be taken to improve the probability of rugged segment
such as industrial inspection applications, at least some
the environment is possible at times. The experienced i
designer invariably pays considerable attention to suc

ponents or broken connection paths. There is no poi
tion past the level of detail required to identify those
Segmentation of nontrivial images is one of the mos
processing. Segmentation accuracy determines the ev
of computerized analysis procedures. For this reason, c
be taken to improve the probability of rugged segment
such as industrial inspection applications, at least some
the environment is possible at times. The experienced i
designer invariably pays considerable attention to suc

ponents or broken connection paths. There is no poi
tion past the level of detail required to identify those
Segmentation of nontrivial images is one of the mos
processing. Segmentation accuracy determines the ev
of computerized analysis procedures. For this reason, c
be taken to improve the probability of rugged segment
such as industrial inspection applications, at least some
the environment is possible at times. The experienced i
designer invariably pays considerable attention to suc

FIGURE 7.30 (a) Text image of size pixels. (b) Complement of (a) for use as a mask image. (c) Marker image. (d) Result of hole-filling

(Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

Border Clearing

Object extraction from images is a critical problem in automated image analysis. A procedure for deleting items that touch (are connected) the image boundary is advantageous since it leaves only complete objects for subsequent processing. it indicates that only a portion of an object remains visible in the field of view.

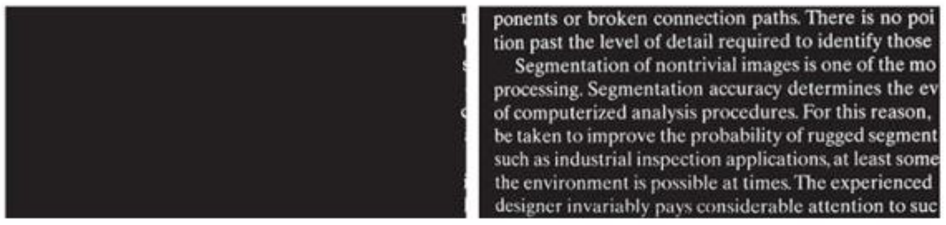
As a mask, the original image is used. The image of the marker is

$$F(x,y) = \{ \begin{matrix} 1 - I(x,y) \\ 0 \quad otherwise \end{matrix} \; if\ (x,y) is\ on\ border\ of\ I$$

The border clearing algorithm calculates a morphological reconstruction first $R_I^D(F)$, which essentially extracts the items that touch the boundary, and then generates a new image with no objects touching the borders $I - R_I^D(F)$.

Examine the original text image from Fig. 7.31 (a) once again. Figure 7.31 (a) illustrates the reconstruction $R_I^D(F)$generated by employing a 1's 3x3 structural element. The objects that contact the original image's border can be seen on the right side of Fig. 7.31 (a). The image X in Figure 7.31 (b) was produced. If the task at hand is automated character recognition, obtaining an image in which no characters contact the border is extremely beneficial because it avoids the difficulty of recognizing partial characters (at best a challenging work).

a b



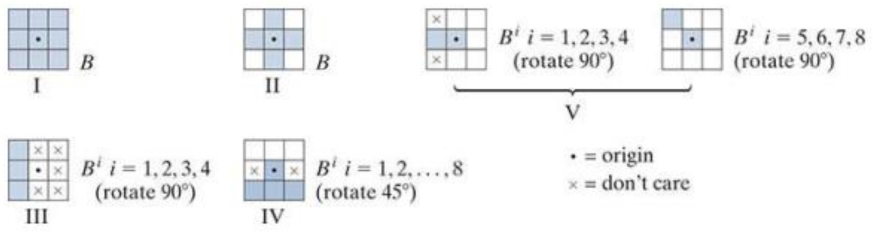
ponents or broken connection paths. There is no poi
tion past the level of detail required to identify those
Segmentation of nontrivial images is one of the mo
processing. Segmentation accuracy determines the ev
of computerized analysis procedures. For this reason,
be taken to improve the probability of rugged segment
such as industrial inspection applications, at least some
the environment is possible at times. The experienced
designer invariably pays considerable attention to suc

FIGURE 7.31 (a) Reconstruction by dilation of marker image. (b) Image with no objects touching the border.

(Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

7.6 Summary of Morphological Operations on Binary Images

The structure elements employed in the various binary morphological approaches covered thus far are summarized in Figure 7.32. The shaded elements represent foreground values (which are normally denoted by 1's in numerical arrays), the white elements represent background values (which are typically denoted by 0's), and the x's represent "don't care" elements.



B
I
B
II
B^i $i = 1, 2, 3, 4$ (rotate 90°)
B^i $i = 5, 6, 7, 8$ (rotate 90°)
V
B^i $i = 1, 2, 3, 4$ (rotate 90°)
III
B^i $i = 1, 2, \ldots, 8$ (rotate 45°)
IV
• = origin
× = don't care

FIGURE 7.32 Five basic types of structuring elements used for binary morphology

(Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

Grayscale Morphology

Grayscale morphology's structuring components provide the same primary role as their binary counterparts: they act as "probes" to evaluate a given image for specific properties. In grayscale morphology, structural features fall into one of two categories: nonflat and flat. Each is illustrated in Figure 7.33. The image in Figure 7.33 (a) depicts a hemispherical grayscale SE, while Figure 7.33 (c) depicts a horizontal intensity profile through its center. Figure 7.33 (b) depicts a flat structuring element in the shape of a disk, while Figure 7.33 (d) depicts the element's equivalent intensity profile. To facilitate understanding, the elements in Fig. 7.33 are depicted as continuous quantities; their computer implementation is based on digital approximations. Grayscale nonflat

SEs are not widely utilized in practice due to a variety of issues. The erosion of an image f caused by a SE b at any point (x,y) is defined as the image's minimum value in the region coinciding with b when b's origin is at (x,y):

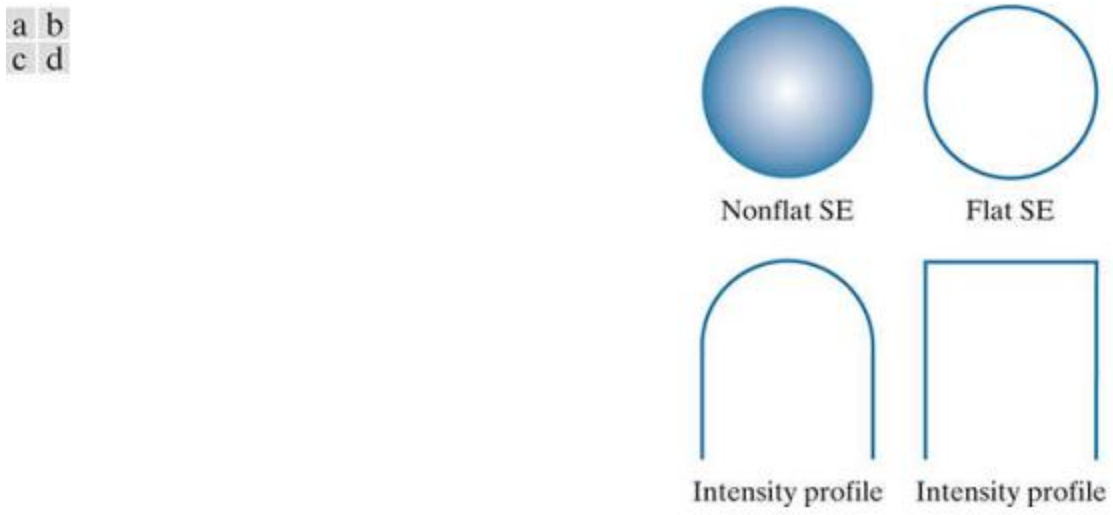


FIGURE 7.33 Nonflat and flat structuring elements, and corresponding horizontal intensity profiles through their centers. All examples in this section are based on flat SEs.

(Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

Grayscale Erosion and Dilation

The grayscale erosion of *f* caused by a flat structuring element b at locations (x, y) is defined as the image's least value in the region coincident with b(x, y) when b's origin is at (x, y) . The erosion of an image f by a structuring element b at (x, y) is expressed in equation form as

$$[f \ominus b](x,y) = \min_{(s,t)\in b}\{f(x+s, y+t)\}$$

A similar concept can be applied to spatial correlation, in which the coordinates x and y are incremented through all possible values to arrive at the origin of b, which then visits every pixel in f. That is, we locate the structural element at every pixel location in the image, and use that as the starting point to find the erosion of *f* by b. Erosion happens only at the locations that have a minimum value of *f* with respect to b. For example, if b is a square structuring element of size 3X3 and a point has the minimum value of *f* in the region that it spans, then finding the erosion at that point requires locating the minimum of the nine values of *f* that are located within the defined region that is spanned by b when its origin is at that point.

Correspondingly, the grayscale dilation of *f* via a flat structuring element b at any point on the coordinate plane (x, y) is determined by the maximum value of the image in the window spanned by $\hat{b}$ when the origin of $\hat{b}$ is at (x, y). That is,

$$[f \oplus b](x,y) = \max_{(s,t)\in\hat{b}}\{f(x-s,y-t)\}$$

Whereas in grayscale erosion with a flat SE, each neighbourhood of (x, y) containing a coincident of b computes the minimum intensity value of *f* in the neighbourhood, we expect that the eroded image will be darker than the original. It also follows that bright features will be reduced in size, while dark features will be enlarged. In Fig. 7.34 (b), erosion is demonstrated by using a disk SE with a radius of 2 pixels and a height of 2 pixels. What we're seeing here is obvious: the effects that have been described are apparent in the eroded image. Consider, for example, the extent to which the intensities of the small bright dots in Fig. 7.34 (b) were reduced, which caused them to barely be visible. At the same time, the dark features in the figure grew in thickness. The erosion is slightly darker in general compared to the original image's background.

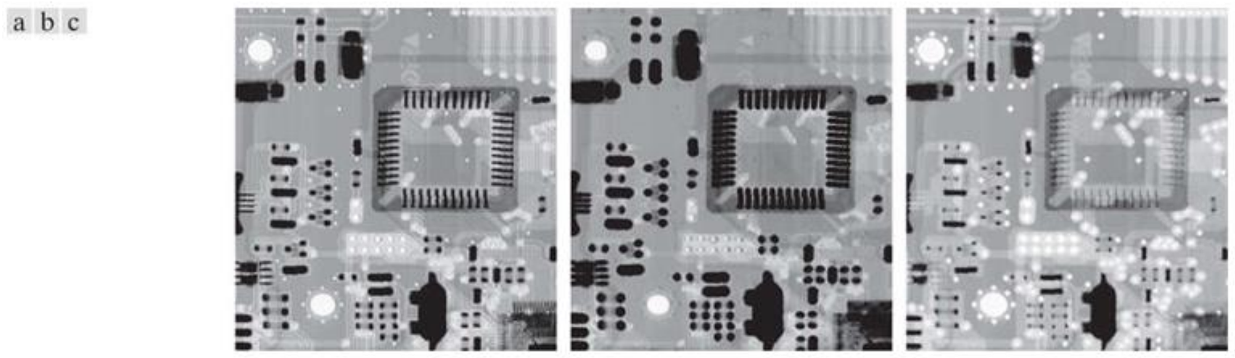


FIGURE 7.34 (a) Gray-scale X-ray image of size pixels. (b) Erosion using a flat disk SE with a radius of 2 pixels. (c) Dilation using the same SE

(Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

As shown in Fig. 7.34 (c), this was the outcome of dilation of the same SE . Erosion does the opposite of creating these effects. To make the features more apparent, the light ones were thickened and the darker ones were made less noticeable. In Fig. 7.34 (a) lacks the visible of thin black connecting wires located in the left, middle, and right bottom. However, unlike the eroded small white dots in Fig. 7.34 (b), the dark dots in the dilated image are still obvious. As it turns out, the reason the black dots are larger than the white dots is because the black dots were initially larger than the white dots in terms of their size relative to the SE. Finally, compare the dilated and non-dilated images in Fig. 7.34 (a)

The erosion of image f by a nonflat SE b_N is defined as:

$$[f \ominus b](x,y) = \min_{(s,t)\in b_N}\{f(x+s,y+t) - b_N(s,t)\}$$

The dilation of image f by a nonflat SE b_N is defined as:

$$[f \oplus b_N](x,y) = \max_{(s,t)\in\widehat{b_N}}\{f(x-s,y-t) + \widehat{b_N}(s,t)\}$$

When the SE is flat, the equations become identical to previous ones, as long as they do not contain a constant.

When dual operations like erosion and dilation are taken into consideration, one's functions and abilities can be complemented and reflected.

$$[f \ominus b]^c(x,y) = \left[f^c \oplus \hat{b}\right](x,y)$$

Similarly,

$$[f \oplus b]^c(x,y) = \left[f^c \oplus \hat{b}\right](x,y)$$

Grayscale Opening and Closing

The opening of image f by SE b is:

$$f \circ b = (f \ominus b) \oplus b$$

Opening is merely f being eroded of f by b, followed by being dilated the result by b. Likewise, the grayscale closing of f by b is signified by $f \cdot b$

$$f \cdot b = (f \oplus b) \ominus b$$

Grayscale images' opening and closing are duals in terms of complementation and SE reflection:

7.35 And

$$(f \circ b)^c = f^c \cdot \hat{b}$$

The concept is illustrated in one dimension in Figure 7.35. Assume the curve in Fig. 7.35 (a) represents an image's intensity profile along a single row. In Figure 7.35 (b), a flat structuring element is shown in so many positions, pushed up against the curve's bottom. The thick curve in Fig. 7.35 (c) represents the entire opening. So because structuring element is too large to fit completely inside the upward peaks of the curve, the opening clips the tops of the peaks, with the amount clipped proportional to the structuring element's reach into the peak. In general, openings are used to eliminate small, bright details while maintaining the overall intensity levels and larger bright features.

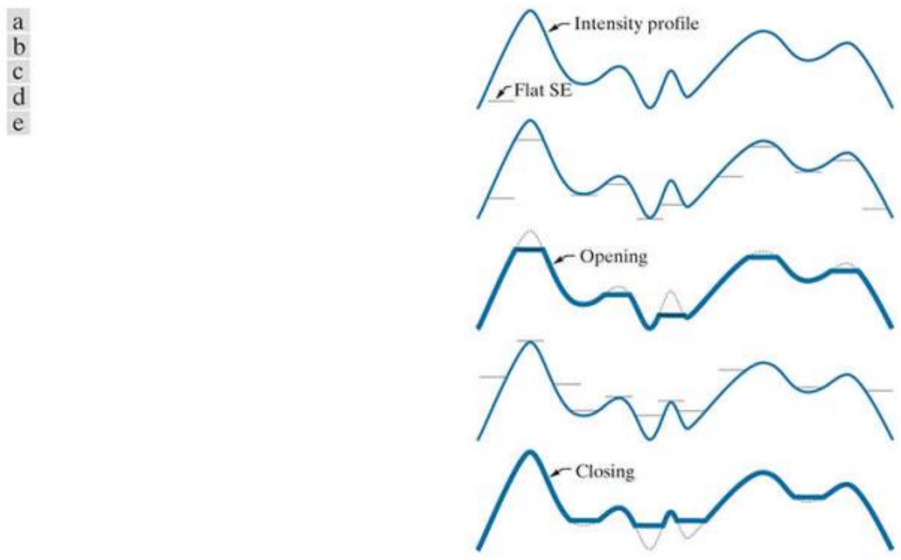


FIGURE 7.35 Grayscale opening and closing in one dimension. (a) Original 1-D signal. (b) Flat structuring element pushed up underneath the signal (c) Opening. (d) Flat structuring element pushed down along the top of the signal. (e) Closing

(Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

Closing is depicted graphically in Figure 7.35 (d). Take note of how the structuring element is pushed down onto the curve as it is translated to all regions. The closing, as illustrated in Fig. 7.35 (e), is formed by determining the lowest points achieved by every part of the structuring element since it slides against the curve's upper side. The grayscale opening compliance with the following requirements:

a. $f^{\circ}b \dashv f$
b. $if\ f_1 \dashv f_2\ \ then\ f_1{}^{\circ}b \dashv f_2{}^{\circ}b$
c. $(f^{\circ}b) \circ b = f^{\circ}b$

The notation $q \dashv r$ is being used to denote that the domain of q is a subset of the domain of r, as well as that any (x, y) in the domain of q is also a subset of the domain of r.

Likewise, the closing operation meets the following requirements:

a. $f \dashv f \cdot b$
b. $if\ f_1 \dashv f_2\ \ then\ f_1 \cdot b \dashv f_2 \cdot b$
c. $(f \cdot b) \cdot b = f \cdot b$

Example 1 Grayscale opening and closing.

Figure 7.36 broadens the one-dimensional concepts illustrated in Figure 7.36 to two dimensions. 7.35 The image in Figure 7.36 (a) is identical to the one and Fig. 7.36 (b) is the opening created by a disk structuring element with a height of one pixel and a radius of three pixels. As expected, the intensity of all bright features decreased in proportion to their sizes relative to the SE. When compared to Fig. 7.36 (b), we see that, in contrast to erosion, opening had a negligible effect on the image's dark features and a negligible effect on the background. Likewise, Fig. 7.36 (c) illustrates the image being closed with a disc of radius 5 (because the small round black dots are larger than the small white dots, a larger disc was required to achieve comparable results to the opening). The bright details and background in this image were largely unaffected, but the dark features were attenuated, with the degree of attenuation dependent on the features' relative sizes to the SE.

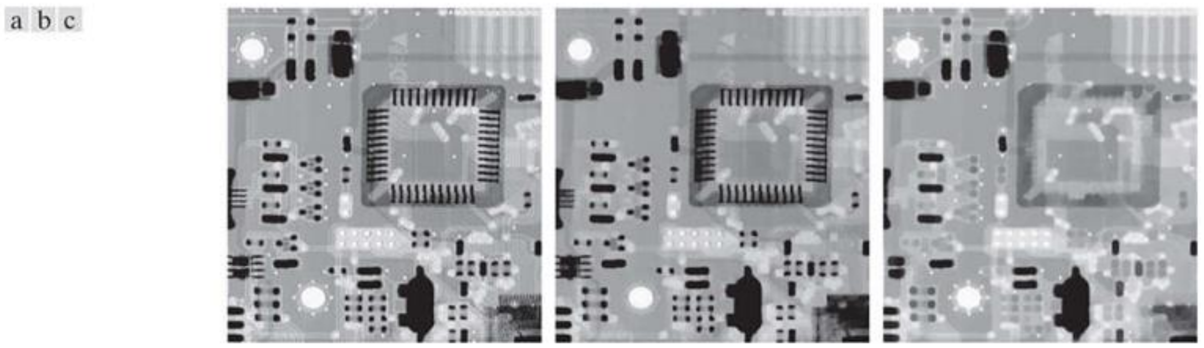


FIGURE 7.36 (a) A grayscale X-ray image of size pixels. (b) Opening using a disk SE with a radius of pixels. (c) Closing using an SE of radius 5.

(Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

7.8 Some Basic Grayscale Morphological Algorithms

Numerous grayscale morphological techniques have been developed on the basis of the grayscale morphological concepts introduced thus far. The following discussion illustrates several of these algorithms.

Due to the fact that opening inhibits bright details smaller than the predefined SE while removing dark details fairly unaffected and closing has the opposite effect, these two operations are frequently combined as morphological filters for image smoothing and noise removal. Consider Fig. 7.37 (a), which depicts an X-ray image of the Cygnus Loop supernova. Suppose again for purpose of this study that the central light region is the objective and that the smaller components are noise. Our goal is to eliminate noise. The result of opening the original image with a flat disk of radius 1 and then closing it with a SE of the same size is shown in Figure 7.37 (b). The figures 7.37 (c) and (d) illustrate the same operation with SEs of 3 and 5 radii, respectively. As expected, this sequence demonstrates progressive elimination of small components as SE size increases. The final result demonstrates that noise has been largely eliminated. The noise components on the lower right side of the image could not be completely removed due to their size being larger than the other successfully removed image elements.

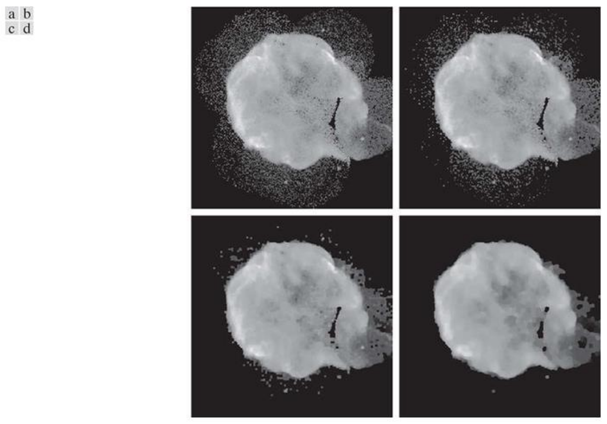


FIGURE 7.37 (a) image of the Cygnus Loop supernova, taken in the X-ray band by NASA's Hubble Telescope. (b)–(d) Results of performing opening and closing sequences on the original image with disk structuring elements of radii, 1, 3, and 5, respectively

(Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

Morphological Gradient

Dilation and erosion, in conjunction with image subtraction, can be used to obtain the morphological gradient, g, of a grayscale image f in the following manner:

$$g = (f \oplus b) - (f \ominus b)$$

Where b denotes an appropriate structuring element. The cumulative consequence of this equation is that dilation thickens and erosion shrinks regions in an image. Their distinction highlights the divisions between regions. Because homogenous regions are unaffected (as long as the SE is not too large in relation to the image's resolution), the subtraction operation seeks to eliminate them. As a result, an image is created in which the edges are enhanced and the homogeneous areas are suppressed, creating a "derivative like" (gradient) effect.

An illustration of this is shown in Figure 7.38. Figure 7.38 (a) depicts a CT scan of the head, while the following two figures depict the opening and closing with a flat SE of 1's. Take note of the aforementioned thickening and shrinking. The morphological gradient obtained is depicted in Figure 7.38 (d). As expected of a 2-D derivative image, the boundaries between regions are clearly defined.

a b
c d

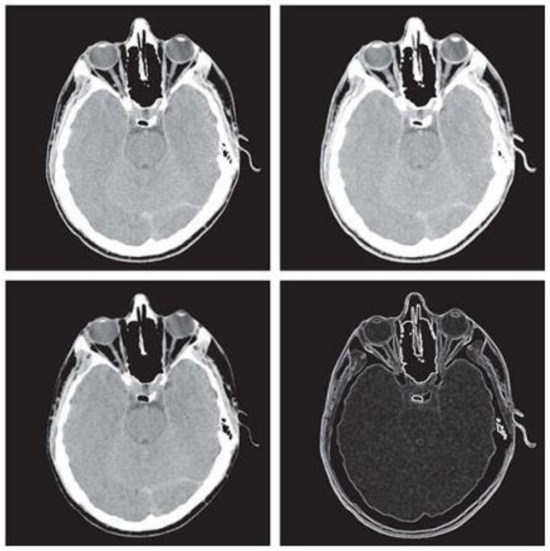


FIGURE 7.38 (a) image of a head CT scan. (b) Dilation. (c) Erosion. (d) Morphological gradient, computed as the difference between (b) and (c)

(Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

Top-Hat and Bottom-Hat Transformations

Combining image subtraction with open and closed operations yields a top-hat transform and a bottom-hat transform. A grayscale image f's top-hat transformation is defined as f minus its opening.

$$T_{hat}(f) = f - (f \cdot b)$$

Similarly, the bottom-hat transformation of f is defined as the closing of f minus f:

$$B_{hat}(f) = (f \cdot b) - f$$

These transformations have a variety of applications, one of which is the removal of objects from images using an element that opens or closes instead of resizing the object that was removed. After that, the difference operation produces an image that contains only the component that was deleted. In the case of bright objects on a dark background, the top-hat transform is used, whereas the bottom-hat transform is used in the case of the opposite. The white top hat transform and the black bottom-hat transform are two common names for these transformations.

Take, for example, Fig. 7.39 (a), which depicts a grain of rice in various states of development. In this image, nonuniform lighting was used to create the image, as evidenced by the darker area in the bottom rightmost portion of the image. As shown in Figure 7.39 (b), thresholding was performed with the Otsu method, which is an optimal thresholding method . When nonuniform illumination is used in conjunction with a nonuniform image, the result is segmentation errors in the dark area (where several grains of rice were not obtained from the background) and in the top left portion, where parts of the background were interpreted as rice. Figure 7.39 (c) depicts the image being opened by a disk with a radius of 40 degrees. This SE was large enough that it could not be accommodated in any of the objects. A result of this was that the objects were removed, having left only a close estimate of the background to be seen. In this image, the shading pattern is clearly visible. It is expected that the background will become more uniform as a result of subtracting this image from the original (i.e., applying a top-hat transformation). As illustrated in Fig. 7.39 (d), this is indeed the case. Although the background is not perfectly uniform, the differences between light and dark extremes are less extreme, and this was sufficient to produce a correct thresholding result, in which all of the rice grains were properly extracted using Otsu's method, as shown in Fig. 7.39 (e).

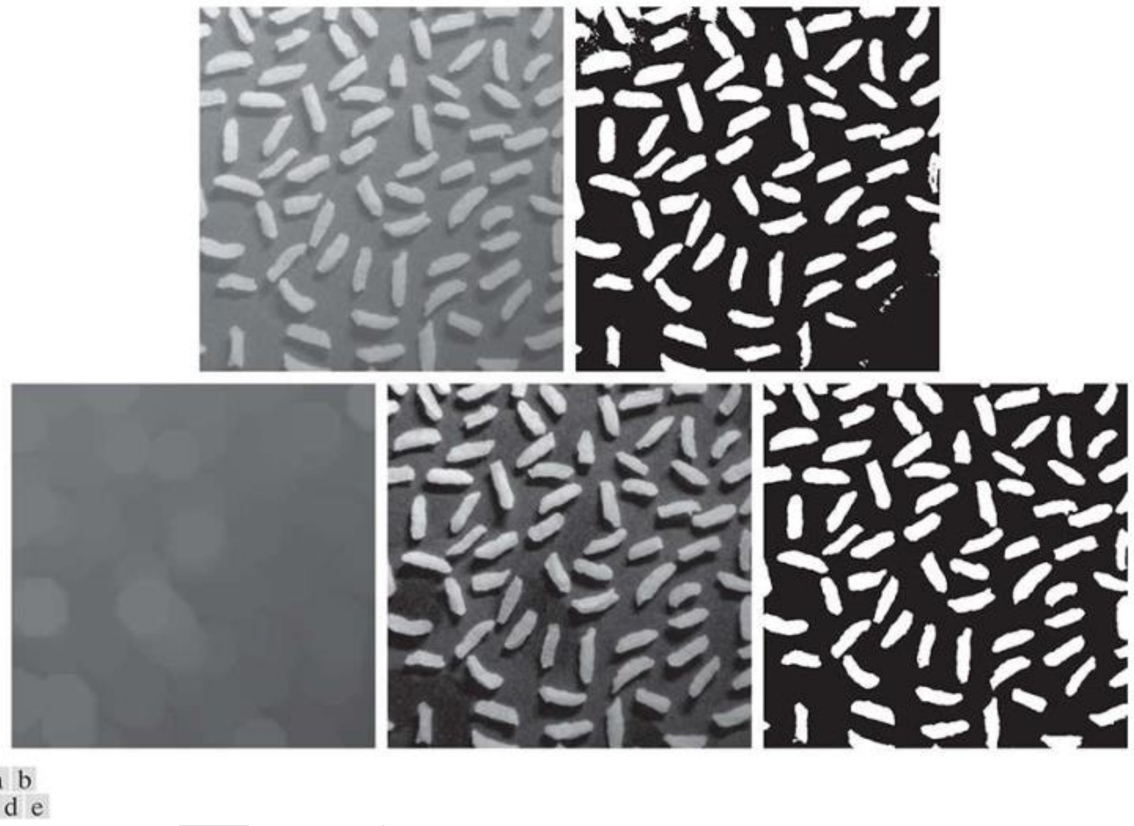


FIGURE 7.39 Using the top-hat transformation for shading correction. (a) Original image of size pixels. (b) Thresholded image. (c) Image opened using a disk SE of radius 40. (d) Top-hat transformation (the image minus its opening). (e) Thresholded top-hat image.

(Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

Granulometry

Granulometry is a branch of science that is concerned with determining the size distribution of particles in an image by analysing the image. It is difficult to count particles when they are not neatly separated, which makes counting based on individual particle identification a difficult task.

It is possible to estimate particle size distribution indirectly using morphology, without having to identify and measure individual particles, by using morphology.

The technique is simple and clear. When dealing with particles that have regular shapes and are lighter in color than the background, the technique comprises in implementing openings with SEs that grow in size as the particle size increases. The underlying concept is that opening operations of a specific size should have the greatest impact on regions of the input image that contain particles of similar size to the opening operation. We compute the sum of the pixel values for each image that results from an opening in the window. Because, as we discussed earlier, openings reduce the intensity of light features in an image, this sum, which is referred to as the surface area, decreases as the SE size increases. In this case, the procedure produces a 1-D array, with each element corresponding to a single pixel in the opening for the size SE corresponding to that particular location in the array. For the purpose of highlighting the differences between successive openings, we compute the difference between adjacent elements of the one-dimensional array. If the differences are plotted, the peaks in the plot are indicative of the size distributions of the particles in the image that are most prevalent in the image.

Consider the image of wood dowel plugs in Fig. 7.40 (a), which depicts two dominant sizes of wood dowel plugs. Due to the possibility of variations in the openings caused by the wood grain in the dowels, smoothing the openings is a sensible preprocessing step. Figure 7.40 (b) depicts an image that has been smoothed using the morphological smoothing filter discussed previously, with a disk radius of 5. Figures 7.40 (c) through (f) depict image openings created by disks with radii of 10, 20, 25, and 30 degrees, respectively. The small dowels' contribution to the intensity contribution in Fig. 7.40 (d) has been nearly eliminated, as can be seen in the figure. The contribution of the large dowels has been significantly reduced in Fig. 7.40 (e), and it has been reduced even further in Fig. 7.40 (f). Because it is smaller in size than the other larger dowels, the large dowel near the top right corner of the image is much darker than the others in Fig. 7.40 (e). If we had been attempting to detect faulty dowels, this would have been useful information to have.

a b c
d e f

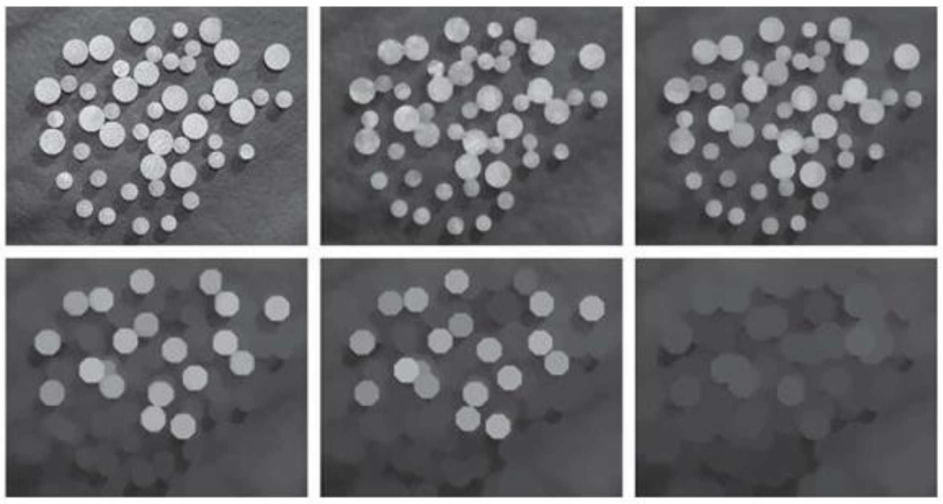


FIGURE 7.40 (a) image of wood dowels. (b) Smoothed image. (c)–(f) Openings of (b) with disks of radii equal to 10, 20, 25, and 30 pixels, respectively.

(Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

A plot of the difference array is depicted in Figure 7.41. As stated earlier, we assume statistically significant differences (peaks in the plot) across radii at which the SE is significant enough to accommodate a collection of particles with approximately the same diameter as the particle . The result shown in Fig. 7.41 has 2 different peaks, which indicates that there are two dominant object sizes in the image.

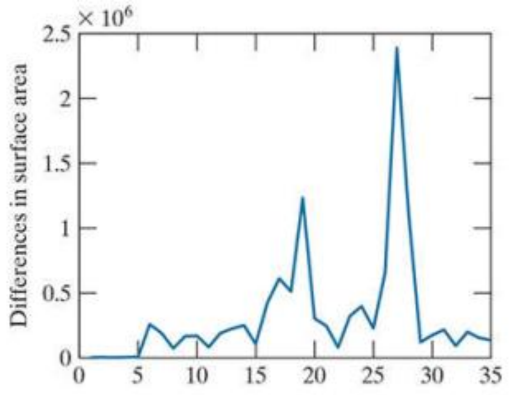


FIGURE 7.41 Differences in surface area as a function of SE disk radius, r. The two peaks indicate that there are two dominant particle sizes in the image.

(Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

Textural Segmentation

Figure 7.42 (a) depicts a image of dark blobs overlaid on a light background, which is a noisy image. The image is divided into two textural regions: a region on the right that is composed of large blobs, and a region on the left that is composed of smaller blobs. Finding a boundary between two regions depending on their textural content, although in this scenario is dictated by the sizes and spatial distribution of the blobs. The process of dividing an image into regions is referred to as segmentation.

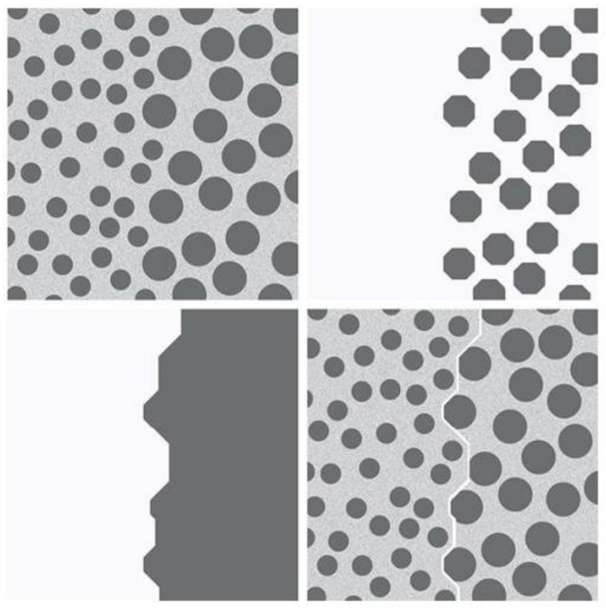


FIGURE 7.42 Textural segmentation. (a) A image consisting of two types of blobs. (b) Image with small blobs removed by closing (a). (c) Image with light patches between large blobs removed by opening (b). (d) Original image with boundary between the two regions in (c) superimposed. The boundary was obtained using a morphological gradient.

(Reference from "Digital Image Processing fourth edition by Rafael C. Gonzalez and Richard E. Woods)

As we can see, the objects of interest are darker than the background, and thus we know that if we close the image with a structural element that is larger than the small, distracting blobs, the blobs will be erased. Fig. 7.42 (b), which shows the outcome of closing the input image with a disk with a radius of 30 pixels, demonstrates that this is in fact the case. It is around 25 pixels in radius for the smallest blobs. As a result, we have an image of large, dark blobs on a light background at this stage. We can achieve a similar result by opening this image with a structuring element that is large in comparison to the separation between these blobs. The net outcome must be an image wherein the light patches between both the blobs are erased, leaving the dark blobs and the newly dark patches between these blobs. A disk with a radius of 60 is used to obtain the goals shown in Figure 7.42 (c).

Grayscale Morphological Reconstruction

Grayscale morphological reconstruction is based on the same principles as binary images. The marker and mask pictures, respectively, are denoted by the *f* and *g*. We will consider that the both images are grayscale images of the same size, which means that the intensity of f at any point in the image will be smaller than the intensity of g at that point in the image. With respect to g, the geodesic dilation of *f* of size 1 is seen and is defined as.

$$D_g^{(1)}(f) = (f \oplus b) \wedge g$$

where ∧ Represents the point-wise minimum operator and b is a valid structuring element. The geodesic dilation of size 1 is obtained by computing the dilation of *f* by *b* first, then choosing the minimum between the result and *g* at each location (x, y)

The geodesic dilation of size n of *f* with respect to g is defined as

$$D_g^{(n)}(f) = D_g^{(1)}(D_g^{(n-1)}f)$$

With $D_g^{(0)}(f) = f$

Similarly, the geodesic erosion of size 1 of *f* with respect to g is defined as

$$E_g^{(1)}(f) = (f \ominus b) \vee g$$

where denotes the point-wise maximum operator. The geodesic erosion of size n is defined as

$$E_g^{(n)}(f) = E_g^{(1)}(E_g^{(n-1)}f)$$

With $E_g^{(0)}(f) = f$

The morphological reconstruction by dilation of a grayscale mask image, g, by a grayscale marker image, *f*, indicated by is described as the geodesic dilation of *f* with respect to g, iterated until stability is achieved;

$$R_g^{(D)}(f) = D_g^{(k)}(f)$$

With k such that$D_g^{(k)}(f) = D_g^{(k+1)}(f)$

The morphological reconstruction caused by erosion of g by f, defined by $R_g^{(E)}(f)$ is similarly defined as

$$R_g^{(E)}(f) = E_g^{(k)}(f)$$

With k such that $E_g^{(k)}(f) = E_g^{(k+1)}(f)$

For the same reasons as in the binary case, opening by reconstruction of grayscale images begins with eroding the input image and using it as a marker, and then utilizes that image as the mask. The opening through reconstruction of size n of an image *f* is represented by dilation of the erosion of size n of *f* with respect to *f*; that is,

$$O_R^{(n)}(f) = R_f^{(D)}(f \ominus nb)$$

Where $f \ominus nb$ denotes n successive erosions by b, starting with f ,

Furthermore, the closing of an image *f* by reconstruction of size *n* is defined as the reconstruction by erosion of the dilation of size *n* of *f* with respect to *f*; that is,

$$C_R^{(n)}(f) = R_f^{(E)}(f \oplus nb)$$

In this case, represent n the number of successive dilations by b, starting from the letter f. Because of duality, it is possible to achieve the closing by reconstruction of an image by complementing the image, acquiring the opening by reconstruction, and then complementing the outcome. As a final point, as demonstrated in the following example, an important method known as top-hat by reconstruction involves subtracting with an image the opening created by reconstruction.

7.8 Unit End questions

1. What is a Morphological image processing?
2. Write a short note on Erosion and Dilation.
3. Write a short note on opening and closing.
4. Explain the Hit-or-Miss Transform.
5. Write a short note on Boundary Extraction.
6. Write a short note on Thinning and Thickening.
7. Write a short note on Convex hull and Hole filling.
8. Write a short note on Pruning and Skeletonization.
9. Explain the Geodesic Dilation and Erosion.
10. Explain the Morphological Reconstruction by Dilation and by Erosion.
11. Write a short note on Opening by Reconstruction.
12. State the algorithm for filling holes.
13. Explain the Grayscale Morphology.
14. Write a short note on Grayscale Erosion and Dilation.
15. Explain the Grayscale Opening and Closing.
16. Explain the Morphological Smoothing.
17. Explain the Morphological Gradient.
18. Write a short note on Top-Hat and Bottom-Hat Transformations.
19. What is a Granulometry?
20. What is a Textural Segmentation?

7.9 Reference for further reading

1. Digital Image Processing Gonzalez and Woods Pearson/Prentice Hall Fourth 2018
2. Fundamentals of Digital Image Processing A K. Jain PHI
3. The Image Processing Handbook J. C. Russ CRC Fifth 2010