UNIT-I

DIGITAL IMAGE FUNDAMENTALS & IMAGE TRANSFORMS

Digital image fundamentals & Image Transforms:- Digital Image fundamentals, Sampling and quantization, Relationship between pixels.

Image Transforms: 2-D FFT, Properties. Walsh transforms, Hadamard Transform, Discrete cosine Transform, Discrete Wavelet Transform.

DIGITAL IMAGE FUNDAMENTALS:

The field of digital image processing refers to processing digital images by means of digital computer. Digital image is composed of a finite number of elements, each of which has a particular location and value. These elements are called picture elements, image elements, pels and pixels. Pixel is the term used most widely to denote the elements of digital image.

An image is a two-dimensional function that represents a measure of some characteristic such as brightness or color of a viewed scene. An image is a projection of a 3-D scene into a 2D projection plane.

An image may be defined as a two-dimensional function ***f(x,y),*** where x and y are spatial (plane) coordinates, and the amplitude of ***f*** at any pair of coordinates **(x,y)** is called the intensity of the image at that point.

The term ***gray level*** is used often to refer to the intensity of monochrome images. Color images are formed by a combination of individual 2-D images.

For example: The RGB color system, a color image consists of three (red, green and blue) individual component images. For this reason many of the techniques developed for Monochrome images can be extended to color images by processing the three component images individually.

An image may be continuous with respect to the x- and y- coordinates and also in amplitude. Converting such an image to digital form requires that the coordinates, as well as the amplitude, be digitized.

APPLICATIONS OF DIGITAL IMAGE PROCESSING

Since digital image processing has very wide applications and almost all of the technical fields are impacted by DIP, we will just discuss some of the major applications of DIP.

Digital image processing has a broad spectrum of applications, such as

- Remote sensing via satellites and other spacecrafts
- Image transmission and storage for business applications
- Medical processing,
- RADAR (Radio Detection and Ranging)
- SONAR(Sound Navigation and Ranging)
- Acoustic image processing (The study of underwater sound is known as underwater acoustics or hydro acoustics.)
- Robotics and automated inspection of industrial parts.
- Images acquired by satellites are useful in tracking of
    - Earth resources;
    - Geographical mapping;
    - Prediction of agricultural crops,
    - Urban growth and weather monitoring
    - Flood and fire control and many other environmental applications.

Space image applications include:

- Recognition and analysis of objects contained in images obtained from deep space-probe missions.
- Image transmission and storage applications occur in broad cast television
- Teleconferencing
- Transmission of facsimile images(Printed documents and graphics) for office automation

Communication over computer networks

- Closed-circuit television based security monitoring systems and
- In military communications.

Medical applications:

- Processing of chest X-rays
- Cine angiograms
- Projection images of transaxial tomography and
- Medical images that occur in radiology nuclear magnetic resonance (NMR)
- Ultrasonic scanning

IMAGE PROCESSING TOOLBOX (IPT) is a collection of functions that extend the capability of the MATLAB numeric computing environment. These functions, and the expressiveness of the MATLAB language, make many image-processing operations easy to write in a compact, clear manner, thus providing a ideal software prototyping environment for the solution of image processing problem.

Components of Image processing System:

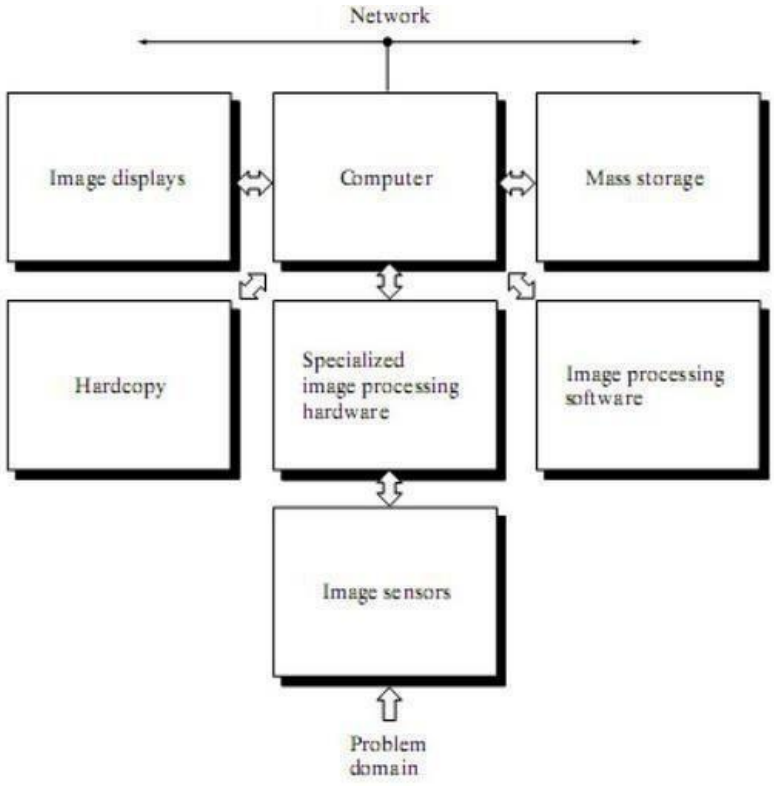


Figure: Components of Image processing System

Image Sensors: With reference to sensing, two elements are required to acquire digital image. The first is a physical device that is sensitive to the energy radiated by the object we wish to image and second is specialized image processing hardware.

Specialize image processing hardware: It consists of the digitizer just mentioned, plus hardware that performs other primitive operations such as an arithmetic logic unit, which performs arithmetic such addition and subtraction and logical operations in parallel on images.

Computer: It is a general purpose computer and can range from a PC to a supercomputer depending on the application. In dedicated applications, sometimes specially designed computer are used to achieve a required level of performance

Software: It consists of specialized modules that perform specific tasks a well designed package also includes capability for the user to write code, as a minimum, utilizes the specialized module. More sophisticated software packages allow the integration of these modules.

Mass storage: This capability is a must in image processing applications. An image of size 1024 x1024 pixels, in which the intensity of each pixel is an 8- bit quantity requires one Megabytes of storage space if the image is not compressed .Image processing applications falls into three principal categories of storage

i) Short term storage for use during processing
ii) On line storage for relatively fast retrieval
iii) Archival storage such as magnetic tapes and disks

Image display: Image displays in use today are mainly color TV monitors. These monitors are driven by the outputs of image and graphics displays cards that are an integral part of computer system.

Hardcopy devices: The devices for recording image includes laser printers, film cameras, heat sensitive devices inkjet units and digital units such as optical and CD ROM disk. Films provide the highest possible resolution, but paper is the obvious medium of choice for written applications.

Networking: It is almost a default function in any computer system in use today because of the large amount of data inherent in image processing applications. The key consideration in image is transmission bandwidth.

Fundamental Steps in Digital Image Processing:

There are two categories of the steps involved in the image processing:

1. Methods whose outputs are input are images.
2. Methods whose outputs are attributes extracted from those images.

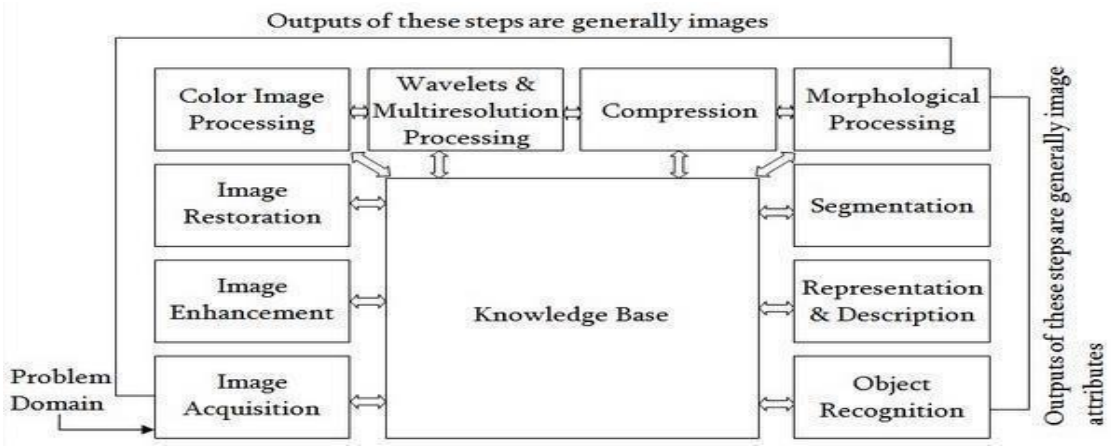


Fig: Fundamental Steps in Digital Image Processing

Image acquisition: It could be as simple as being given an image that is already in digital form. Generally the image acquisition stage involves processing such scaling.

Image Enhancement: It is among the simplest and most appealing areas of digital image processing. The idea behind this is to bring out details that are obscured or simply to highlight certain features of interest in image. Image enhancement is a very subjective area of image processing.

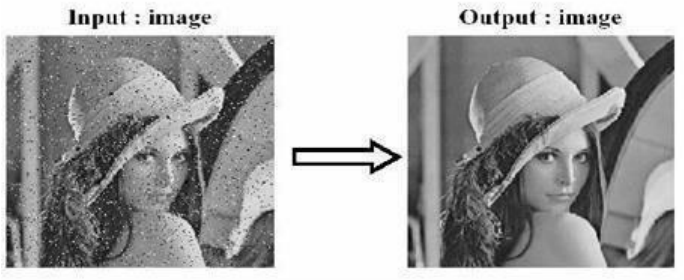
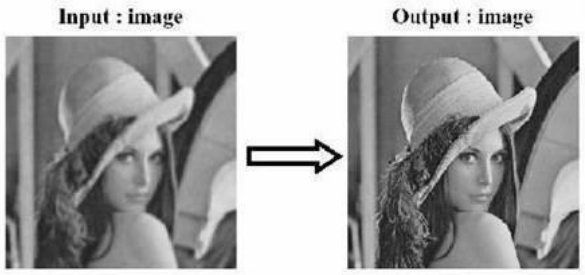


Image Restoration: It deals with improving the appearance of an image. It is an objective approach, in the sense that restoration techniques tend to be based on mathematical or probabilistic models of image processing. Enhancement, on the other hand is based on human subjective preferences regarding what constitutes a "good" enhancement result.



Color image processing: It is an area that is been gaining importance because of the use of digital images over the internet. Color image processing deals with basically color models and their implementation in image processing applications.

Wavelets and Multi resolution Processing: These are the foundation for representing image in various degrees of resolution.

Compression: It deals with techniques reducing the storage required to save an image, or the bandwidth required to transmit it over the network. It has to major approaches a) Lossless Compression b) Lossy Compression

Morphological processing: It deals with tools for extracting image components that are useful in the representation and description of shape and boundary of objects. It is majorly used in automated inspection applications.

Representation and Description: It always follows the output of segmentation step that is, raw pixel data, constituting either the boundary of an image or points in the region itself. In either case converting the data to a form suitable for computer processing is necessary.

Recognition: It is the process that assigns label to an object based on its descriptors. It is the last step of image processing which use artificial intelligence of software.

Knowledge base:

Knowledge about a problem domain is coded into an image processing system in the form of a knowledge base. This knowledge may be as simple as detailing regions of an image where the information of the interest in known to be located. Thus limiting search that has to be conducted is in seeking the information. The knowledge base also can be quite complex such interrelated list of all major possible defects in a materials inspection problems or an image database containing high resolution satellite images of a region in connection with change detection application.

A Simple Image Model:

An image is denoted by a two dimensional function of the form f{x, y}. The value or amplitude of f at spatial coordinates {x,y} is a positive scalar quantity whose physical meaning is determined

by the source of the image. When an image is generated by a physical process, its values are proportional to energy radiated by a physical source. As a consequence, f(x,y) must be nonzero and finite; that is o<f(x,y) <co The function f(x,y) may be characterized by two components- The amount of the source illumination incident on the scene being viewed.

(a) The amount of the source illumination reflected back by the objects in the scene These are called illumination and reflectance components and are denoted by i(x,y) an r (x,y) respectively.

The functions combine as a product to form f(x,y). We call the intensity of a monochrome image at any coordinates (x,y) the gray level (l) of the image at that point l= f (x, y.)

$$L_{min} \leq l \leq L_{max}$$

L_{min} is to be positive and Lmax must be finite

$$L_{min} = i_{min} r_{min}$$

$$L_{max} = i_{max} r_{max}$$

The interval [Lmin, Lmax] is called gray scale. Common practice is to shift this interval numerically to the interval [0, L-l] where l=0 is considered black and l= L-1 is considered white on the gray scale. All intermediate values are shades of gray of gray varying from black to white.

SAMPLING AND QUANTIZATION:

To create a digital image, we need to convert the continuous sensed data into digital from. This involves two processes – sampling and quantization. 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 the image along the line segment AB. To simple this function, we take equally spaced samples along line AB. The location of each samples 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.

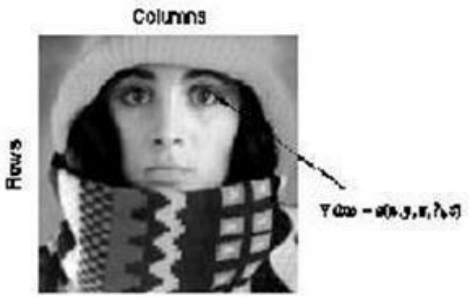
In order to form a digital, the gray level values must also be converted (quantized) into discrete quantities. So we divide the gray level scale into eight discrete levels ranging from eight level values. The continuous gray levels are quantized simply by assigning one of the eight discrete gray levels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

Starting at the top of the image and covering out this procedure line by line produces a two dimensional digital image.

Digital Image definition:

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 mathematics of that sampling process will be described in subsequent Chapters. For now we will look at some basic definitions associated with the digital image. The effect of digitization is shown in figure.

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..N-1 and n=0,1,2…N-1 is f(m,n). In fact, in most cases, is actually a function of many variables including depth, color and time (t).



There are three types of computerized processes in the processing of image

1) Low level process -these involve primitive operations such as image processing to reduce noise, contrast enhancement and image sharpening. These kind of processes are characterized by fact the both inputs and output are images.

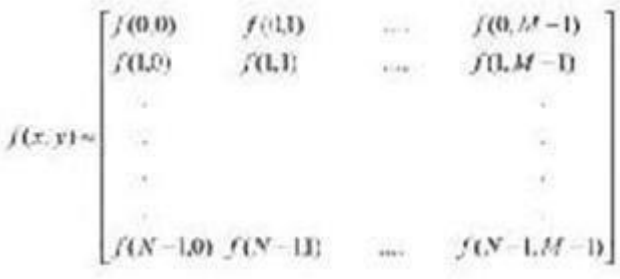
2) Mid level image processing - it involves tasks like segmentation, description of those objects to reduce them to a form suitable for computer processing, and classification of individual objects. The inputs to the process are generally images but outputs are attributes extracted from images.

3) High level processing – It involves "making sense" of an ensemble of recognized objects, as in image analysis, and performing the cognitive functions normally associated with vision.

Representing Digital Images:

The result of sampling and quantization is matrix of real numbers. Assume that an image f(x,y) is sampled so that the resulting digital image has M rows and N Columns. The values of the coordinates (x,y) now become discrete quantities thus the value of the coordinates at orgin become (X,y) =(0,0) The next Coordinates value along the first signify the image along the first

row. It does not mean that these are the actual values of physical coordinates when the image was sampled.



Thus the right side of the matrix represents a digital element, pixel or pel. The matrix can be represented in the following form as well. The sampling process may be viewed as partitioning the xy plane into a grid with the coordinates of the center of each grid being a pair of elements from the Cartesian products Z2 which is the set of all ordered pair of elements (Zi, Zj) with Zi and Zj being integers from Z. Hence f(x,y) is a digital image if gray level (that is, a real number from the set of real number R) to each distinct pair of coordinates (x,y). This functional assignment is the quantization process. If the gray levels are also integers, Z replaces R, the and a digital image become a 2D function whose coordinates and she amplitude value are integers. Due to processing storage and hardware consideration, the number gray levels typically is an integer power of2.

$$L=2^k$$

Then, the number, b, of bites required to store a digital image is b=M *N* k When M=N, the equation become $b=N^2*k$

When an image can have 2k gray levels, it is referred to as “k- bit”. An image with 256 possible gray levels is called an “8- bit image” ($256=2^8$).

Spatial and Gray level resolution:

Spatial resolution is the smallest discernible details are an image. Suppose a chart can be constructed with vertical lines of width w with the space between the also having width W, so a line pair consists of one such line and its adjacent space thus. The width of the line pair is 2w and there is 1/2w line pair per unit distance resolution is simply the smallest number of discernible line pair unit distance.

Gray levels resolution refers to smallest discernible change in gray levels. Measuring discernible change in gray levels is a highly subjective process reducing the number of bits R while repairing the spatial resolution constant creates the problem of false contouring.

It is caused by the use of an insufficient number of gray levels on the smooth areas of the digital image. It is called so because the rides resemble top graphics contours in a map. It is generally quite visible in image displayed using 16 or less uniformly spaced gray levels.

Image sensing and Acquisition:

The types of images in which we are interested are generated by the combination of an "illumination" source and the reflection or absorption of energy from that source by the elements of the "scene" being imaged. We enclose *illumination* and *scene* in quotes to emphasize the fact that they are considerably more general than the familiar situation in which a visible light source illuminates a common everyday 3-D (three-dimensional) scene. For example, the illumination may originate from a source of electromagnetic energy such as radar, infrared, or X-ray energy. But, as noted earlier, it could originate from less traditional sources, such as ultrasound or even a computer-generated illumination pattern. Similarly, the scene elements could be familiar objects, but they can just as easily be molecules, buried rock formations, or a human brain. We could even image a source, such as acquiring images of the sun. Depending on the nature of the source, illumination energy is reflected from, or transmitted through, objects. An example in the first category is light reflected from a planar surface. An example in the second category is when X-rays pass through a patient's body for the purpose of generating a diagnostic X-ray film. In some applications, the reflected or transmitted energy is focused onto a photo converter (e.g., a phosphor screen), which converts the energy into visible light. Electron microscopy and some applications of gamma imaging use this approach. The idea is simple: Incoming energy is transformed into a voltage by the combination of input electrical power and sensor material that is responsive to the particular type of energy being detected. The output voltage waveform is the response of the sensor(s), and a digital quantity is obtained from each sensor by digitizing its response. In this section, we look at the principal modalities for image sensing and generation.

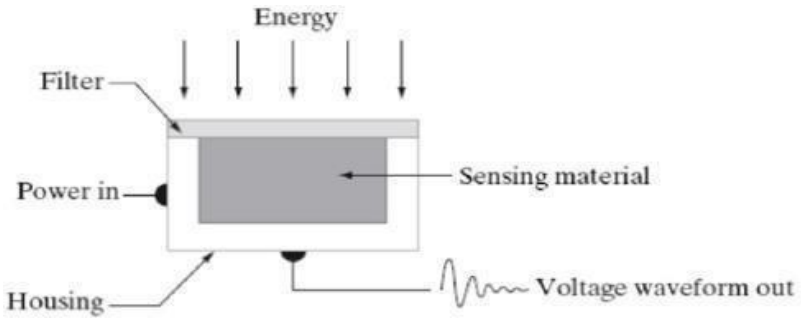


Fig: Single Image sensor

Fig: Line Sensor

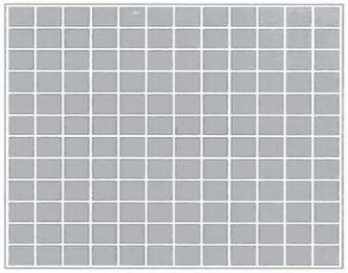
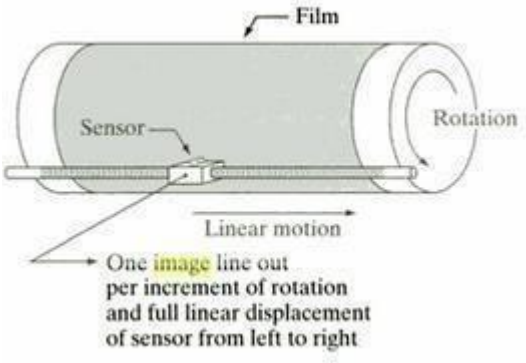


Fig: Array sensor

Image Acquisition using a Single sensor:

The components of a single sensor. Perhaps the most familiar sensor of this type is the photodiode, which is constructed of silicon materials and whose output voltage waveform is proportional to light. The use of a filter in front of a sensor improves selectivity. For example, a green (pass) filter in front of a light sensor favors light in the green band of the color spectrum. As a consequence, the sensor output will be stronger for green light than for other components in the visible spectrum.



In order to generate a 2-D image using a single sensor, there has to be relative displacements in both the x- and y-directions between the sensor and the area to be imaged. Figure shows 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. Other similar mechanical arrangements use a flat bed, with the sensor moving in two linear directions. These types of mechanical digitizers sometimes are referred to as micro

densitometers.

Image Acquisition using a Sensor strips:

A geometry that is used much more frequently than single sensors consists of an in-line arrangement of sensors in the form of a sensor strip, shows. The strip provides imaging elements in one direction. Motion perpendicular to the strip provides imaging in the other direction. This is the type of arrangement used in most flat bed scanners. Sensing devices with 4000 or more in- line sensors are possible. In-line sensors are used routinely in airborne imaging applications, in which the imaging system is mounted on an aircraft that flies at a constant altitude and speed over the geographical area to be imaged. One dimensional imaging sensor strips that respond to various bands of the electromagnetic spectrum are mounted perpendicular to the direction of flight. The imaging strip gives one line of an image at a time, and the motion of the strip completes the other dimension of a two-dimensional image. Lenses or other focusing schemes are used to project area to be scanned onto the sensors. Sensor strips mounted in a ring configuration are used in medical and industrial imaging to obtain cross-sectional ("slice") images of 3-Dobjects.

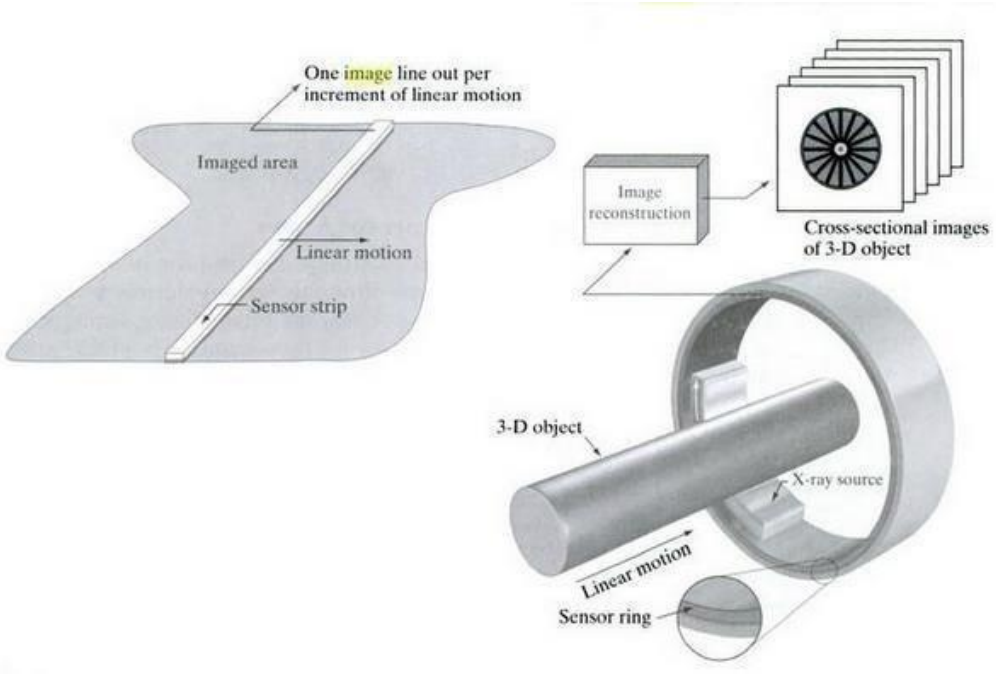


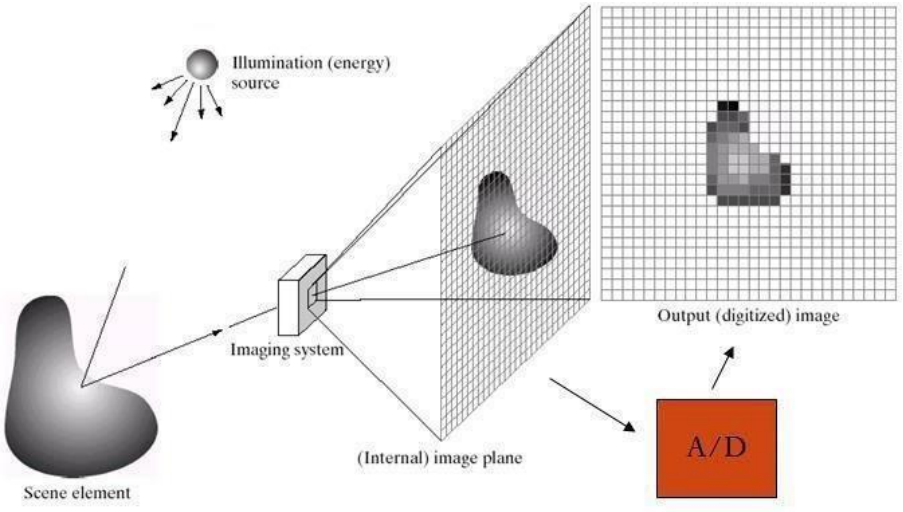
Fig: Image Acquisition using linear strip and circular strips.

Image Acquisition using a Sensor Arrays:

The individual sensors arranged in the form of a 2-D array. Numerous electromagnetic and some ultrasonic sensing devices frequently are arranged in an array format. This is also the predominant

arrangement found in digital cameras. A typical sensor for these cameras is a CCD array, which can be manufactured with a broad range of sensing properties and can be packaged in rugged arrays of elements or more. CCD sensors are used widely in digital cameras and other light sensing instruments. The response of each sensor is proportional to the integral of the light energy projected onto the surface of the sensor, a property that is used in astronomical and other applications requiring low noise images. Noise reduction is achieved by letting the sensor integrate the input light signal over minutes or even hours. The two dimensional, its key advantage is that a complete image can be obtained by focusing the energy pattern onto the surface of the array. Motion obviously is not necessary, as is the case with the sensor arrangements this figure shows the energy from an illumination source being reflected from a scene element, but, as mentioned at the beginning of this section, the energy also could be transmitted through the scene elements. The first function performed by the imaging system is to collect the incoming energy and focus it onto an image plane. If the illumination is light, the front end of the imaging system is a lens, which projects the viewed scene onto the lens focal plane. The sensor array, which is coincident with the focal plane, produces outputs proportional to the integral of the light received at each sensor. Digital and analog circuitry sweeps these outputs and converts them to a video signal, which is then digitized by another section of the imaging system.

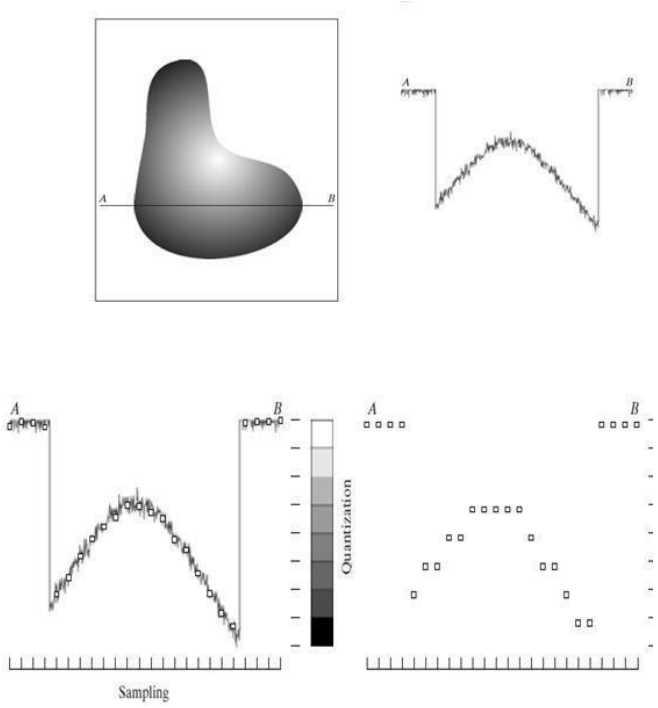
Image sampling and Quantization:



a b c d e

FIGURE An example of the digital image acquisition process. (a) Energy ("illumination") source. (b) An element of a scene. (c) Imaging system. (d) Projection of the scene onto the image plane. (e) Digitized image.

To create a digital image, we need to convert the continuous sensed data into digital form. This involves two processes: *sampling* and *quantization*. A continuous image, f(x, y), that we want to convert to digital form. An image may be continuous with respect to the x- and y-coordinates, and also in amplitude. To convert it to digital form, we have to sample the function in both coordinates and in amplitude. Digitizing the coordinate values is called *sampling*. Digitizing the amplitude values is called *quantization*.



Digital Image representation:

Digital image is a finite collection of discrete samples (*pixels*) of any observable object. The pixels represent a two- or higher dimensional "view" of the object, each pixel having its own discrete value in a finite range. The pixel values may represent the amount of visible light, infra red light, absorption of x-rays, electrons, or any other measurable value such as ultrasound wave impulses. The image does not need to have any visual sense; it is sufficient that the samples form a two-dimensional spatial structure that may be illustrated as an image. The images may be obtained by a digital camera, scanner, electron microscope, ultrasound stethoscope, or any other optical or non-optical sensor. Examples of digital image are:

- Digital photographs
- Satellite images
- radiological images (x-rays, mammograms)

- binary images, fax images, engineering drawings

Computer graphics, CAD drawings, and vector graphics in general are not considered in this course even though their reproduction is a possible source of an image. In fact, one goal of intermediate level image processing may be to reconstruct a model (e.g. vector representation) for a given digital image.

RELATIONSHIP BETWEEN PIXELS:

We consider several important relationships between pixels in a digital image.

NEIGHBORS OF A PIXEL

- A pixel p at coordinates (x,y) has four *horizontal* and *vertical* neighbors whose coordinates are given by: (x+1,y), (x-1, y), (x, y+1), (x,y-1)

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

This set of pixels, called the 4-*neighbors* or p, is denoted by $N_4(p)$. Each pixel is one unit distance from (x,y) and some of the neighbors of p lie outside the digital image if (x,y) is on the border of the image. The four *diagonal* neighbors of p have coordinates and are denoted by $N_D(p)$.

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

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

These points, together with the 4-neighbors, are called the 8-neighbors of p, denoted by $N_8(p)$.

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

As before, some of the points in $N_D(p)$ and $N_8(p)$ fall outside the image if (x,y) is on the

border of the image.

ADJACENCY AND CONNECTIVITY

Let v be the set of gray –level values used to define adjacency, in a binary image, v={1}. In a gray-scale image, the idea is the same, but *V* typically contains more elements, for example, *V* = {180, 181, 182… 200}.

If the possible intensity values 0 – 255, *V* set can be any subset of these 256 values.

if we are reference to adjacency of pixel with value.

Three types of adjacency

- 4- Adjacency – two pixel P and Q with value from V are 4 –adjacency if A is in the set $N_4(P)$
- 8- Adjacency – two pixel P and Q with value from V are 8 –adjacency if A is in the set $N_8(P)$
- M-adjacency –two pixel P and Q with value from V are m – adjacency if (i) Q is in $N_4(p)$ or (ii) Q is in $N_D(q)$ and the set $N_4(p) \cap N_4(q)$ has no pixel whose values are from V.
- Mixed adjacency is a modification of 8-adjacency. It is introduced to eliminate the ambiguities that often arise when 8-adjacency issued.
- For example:

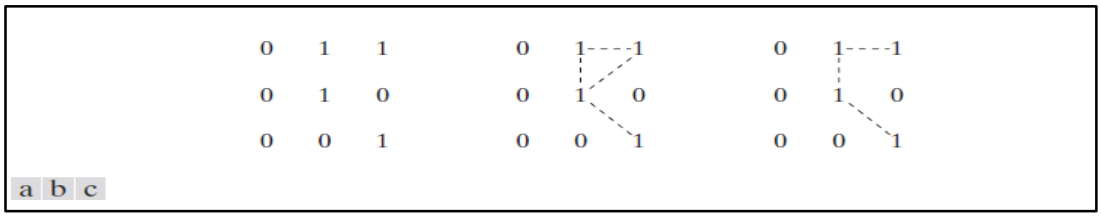


Fig:1.8(a) Arrangement of pixels; (b) pixels that are 8-adjacent (shown dashed) to the center pixel; (c) ***m***-adjacency.

Types of Adjacency:

- In this example, we can note that to connect between two pixels (finding a path between two pixels):
  - In 8-adjacency way, you can find multiple paths between two pixels
  - While, in m-adjacency, you can find only one path between two pixels
- So, m-adjacency has eliminated the multiple path connection that has been generated by the 8-adjacency.
- Two subsets S_1 and S_2 are adjacent, if some pixel in S_1 is adjacent to some pixel in S_2.

Adjacent means, either 4-, 8- or m-adjacency.

A Digital Path:

- A digital path (or curve) from pixel p with coordinate (x,y) to pixel q with coordinate (s,t) is a sequence of distinct pixels with coordinates (x_0,y_0), (x_1,y_1), …, (x_n, y_n) where $(x_0,y_0) = (x,y)$ and $(x_n, y_n) = (s,t)$ and pixels (x_i, y_i) and (x_{i-1}, y_{i-1}) are adjacent for $1 \leq i \leq n$
- n is the length of the path
- If $(x_0,y_0) = (x_n, y_n)$, the path is closed.

We can specify 4-, 8- or m-paths depending on the type of adjacency specified.

- Return to the previous example:

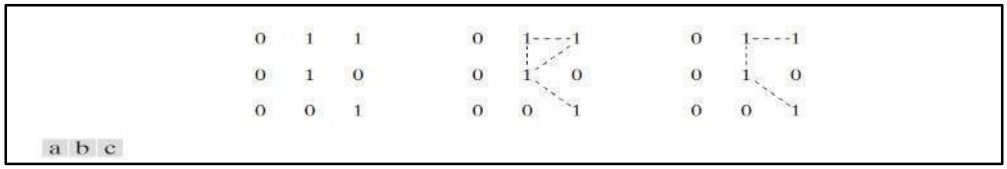


Fig:1.8 (a) Arrangement of pixels; (b) pixels that are 8-adjacent(shown dashed) to the center pixel; (c) m-adjacency.

In figure (b) the paths between the top right and bottom right pixels are 8-paths. And the path between the same 2 pixels in figure (c) is m-path

Connectivity:

- Let S represent a subset of pixels in an image, two pixels p and q are said to be connected in S if there exists a path between them consisting entirely of pixels in$S$.
- For any pixel p in S, the set of pixels that are connected to it in S is called a *connected component* of S. If it only has one connected component, then set S is called a *connected set.*

Region and Boundary:

- REGION: Let R be a subset of pixels in an image, we call R a region of the image if R is a connected set.
- BOUNDARY: The *boundary* (also called *border* or *contour*) of a region R is the set of pixels in the region that have one or more neighbors that are not in R.

If R happens to be an entire image, then its boundary is defined as the set of pixels in the first and last rows and columns in the image. This extra definition is required because an image has no neighbors beyond its borders. Normally, when we refer to a region, we are referring to subset of

an image, and any pixels in the boundary of the region that happen to coincide with the border of the image are included implicitly as part of the region boundary.

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}

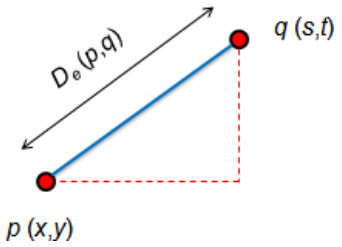
D [p.q] = D [p.q] and

D [p.q] ≥ O {D[p.q]+D(q,z)

- The ***Euclidean Distance*** between p and q is defined as:

$$D_e(p,q) = [(x-s)^2 + (y-t)^2]^{1/2}$$

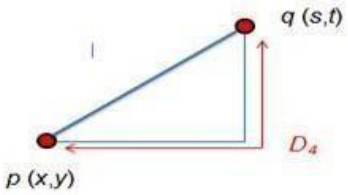
Pixels having a distance less than or equal to some value r from (x,y) are the points contained in a disk of radius ' r 'centered at (x,y)



- The D_4 *distance* (also called ***city-block distance***) between p and q is defined as:

$$D_4(p,q) = |x-s| + |y-t|$$

Pixels having a D_4 distance from (x,y), less than or equal to some value r form a Diamond centered at (x,y)



Example:

The pixels with distance D4 ≤ 2 from (x,y) form the following contours of constant distance.

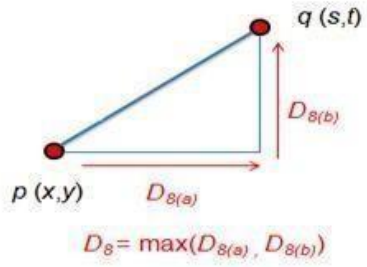
The pixels with $D_4 = 1$ are the 4-neighbors of (x,y)

		2		
	2	1	2	
2	1	0	1	2
	2	1	2	
		2		

- The D_8 *distance* (also called **c*hessboard distance***) between p and q is defined as:

$$D_8(p,q) = \max(|x-s|, |y-t|)$$

Pixels having a D_8 distance from (x,y), less than or equal to some value r form a square Centered at (x,y).



Example:

D_8 distance ≤ 2 from (x,y) form the following contours of constant distance.

2	2	2	2	2
2	1	1	1	2
2	1	0	1	2
2	1	1	1	2
2	2	2	2	2

- **$D_m$distance:**

  It is defined as the shortest m-path between the points.

  In this case, the distance between two pixels will depend on the values of the pixels along the path, as well as the values of their neighbors.

- Example:

  Consider the following arrangement of pixels and assume that p, p_2, and p_4 have value 1 and that p_1 and p_3 can have can have a value of 0 or 1 Suppose that we

	p_3	p_4
p_1	p_2	
p		

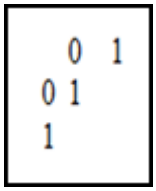
consider the adjacency of pixels values 1 (i.e. $V = \{1\}$)

Now, to compute the D_m between points p and p_4

Here we have 4 cases:

Case1: If $p_1 = 0$ and $p_3 = 0$

The length of the shortest m-path (the D_m *distance) is 2 (p,* p_2*,* p_4*)*

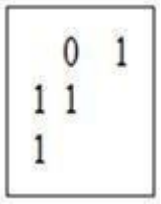


Case2: If $p_1 = 1$ and $p_3 = 0$

Now, p_1 *and p will no longer be adjacent (see m-adjacency definition)*

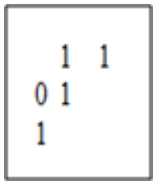
Then, the length of the shortest

path will be 3 (p, p_1, p_2, p_4)



Case3: If $p_1 = 0$ and $p_3 = 1$

The same applies here, and the shortest –m-path will be 3 (p, p_2, p_3, p_4)



Case4: If $p_1 = 1$ and $p_3 = 1$

The length of the shortest m-path will be 4 (p, p_1 ,p_2, p_3, p_4)

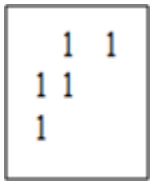


IMAGE TRANSFORMS:

2-D FFT:

2D Discrete Fourier Transform

The independent variable (t,x,y) is discrete

$$F_r = \sum_{k=0}^{N_0-1} f[k] e^{-jr\Omega_0 k}$$

$$f_{N_0}[k] = \frac{1}{N_0} \sum_{r=0}^{N_0-1} F_r e^{jr\Omega_0 k}$$

$$\Omega_0 = \frac{2\pi}{N_0}$$

$$F[u,v] = \sum_{i=0}^{N_0-1} \sum_{k=0}^{N_0-1} f[i,k] e^{-j\Omega_0 (ui+vk)}$$

$$f_{N_0}[i,k] = \frac{1}{N_0^{\,2}} \sum_{u=0}^{N_0-1} \sum_{v=0}^{N_0-1} F[u,v] e^{j\Omega_0 (ui+vk)}$$

$$\Omega_0 = \frac{2\pi}{N_0}$$

Properties

- Linearity $af(x,y) + bg(x,y) \Leftrightarrow aF(u,v) + bG(u,v)$
- Shifting $f(x-x_0, y-x_0) \Leftrightarrow e^{-j2\pi(ux_0+vy_0)} F(u,v)$
- Modulation $e^{j2\pi(u_0x+v_0y)} f(x,y) \Leftrightarrow F(u-u_0, v-v_0)$
- Convolution $f(x,y) * g(x,y) \Leftrightarrow F(u,v)G(u,v)$
- Multiplication $f(x,y)g(x,y) \Leftrightarrow F(u,v) * G(u,v)$
- Separability $f(x,y) = f(x)f(y) \Leftrightarrow F(u,v) = F(u)F(v)$

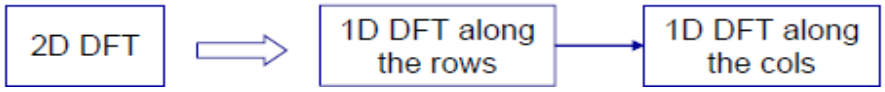
Separability

1. Separability of the 2D Fourier transform
   - 2D Fourier Transforms can be implemented as a sequence of 1D Fourier Transform operations performed *independently* along the two axis

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

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)e^{-j2\pi ux}e^{-j2\pi vy}dxdy=\int_{-\infty}^{\infty} e^{-j2\pi vy}dy\int_{-\infty}^{\infty} f(x,y)e^{-j2\pi ux}dx=$$

$$=\int_{-\infty}^{\infty} F(u,y)e^{-j2\pi vy}dy=F(u,v)$$



Separability

- Separable functions can be written as $f(x,y)=f(x)g(y)$

2. The FT of a separable function is the product of the FTs of the two functions

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

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h(x)g(y)e^{-j2\pi ux}e^{-j2\pi vy}dxdy=\int_{-\infty}^{\infty} g(y)e^{-j2\pi vy}dy\int_{-\infty}^{\infty} h(x)e^{-j2\pi ux}dx=$$

$$=H(u)G(v)$$

$$f(x,y)=h(x)g(y)\Rightarrow F(u,v)=H(u)G(v)$$

WALSH TRANSFORM:

We define now the 1-D Walsh transform as follows:

$$W(u)=\frac{1}{N}\sum_{x=0}^{N-1} f(x)\left[\prod_{i=0}^{n-1}(-1)^{b_i(x)b_{n-1-i}(u)}\right]$$

The above is equivalent to:

$$W(u)=\frac{1}{N}\sum_{x=0}^{N-1}f(x)(-1)^{\sum_{i=1}^{n-1}b_i(x)b_{n-1-i}(u)}$$

The transform kernel values are obtained from:

$$T(u,x)=T(x,u)=\frac{1}{N}\left[\prod_{i=0}^{n-1}(-1)^{b_i(x)b_{n-1-i}(u)}\right]=\frac{1}{N}(-1)^{\sum_{i=1}^{n-1}b_i(x)b_{n-1-i}(u)}$$

Therefore, the array formed by the Walsh matrix is a real symmetric matrix. It is easily shown that it has orthogonal columns and rows

1-D Inverse Walsh Transform

$$f(x)=\sum_{x=0}^{N-1}W(u)\left[\prod_{i=0}^{n-1}(-1)^{b_i(x)b_{n-1-i}(u)}\right]$$

The above is again equivalent to

$$f(x)=\sum_{x=0}^{N-1}W(u)(-1)^{\sum_{i=1}^{n-1}b_i(x)b_{n-1-i}(u)}$$

The array formed by the inverse Walsh matrix is identical to the one formed by the forward Walsh matrix apart from a multiplicative factor N.

2-D Walsh Transform

We define now the 2-D Walsh transform as a straightforward extension of the 1-D transform:

$$W(u,v)=\frac{1}{N}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1}f(x,y)\left[\prod_{i=0}^{n-1}(-1)^{b_i(x)b_{n-1-i}(u)+b_i(y)b_{n-1-i}(v)}\right]$$

- The above is equivalent to:

$$W(u,v)=\frac{1}{N}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1}f(x,y)(-1)^{\sum_{i=1}^{n-1}(b_i(x)b_{n-1-i}(u)+b_i(x)b_{n-1-i}(u))}$$

Inverse Walsh Transform

We define now the Inverse 2-D Walsh transform. It is identical to the forward 2-D Walsh transform

$$f(x,y)=\frac{1}{N}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1}W(u,v)\left[\prod_{i=0}^{n-1}(-1)^{b_i(x)b_{n-1-i}(u)+b_i(y)b_{n-1-i}(v)}\right]$$

The above is equivalent to:

$$f(x,y)=\frac{1}{N}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1}W(u,v)(-1)^{\sum_{i=1}^{n-1}(b_i(x)b_{n-1-i}(u)+b_i(x)b_{n-1-i}(u))}$$

HADAMARD TRANSFORM:

We define now the 2-D Hadamard transform. It is similar to the 2-D Walsh transform.

$$H(u,v)=\frac{1}{N}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1}f(x,y)\left[\prod_{i=0}^{n-1}(-1)^{b_i(x)b_i(u)+b_i(y)b_i(v)}\right]$$

The above is equivalent to:

$$H(u,v)=\frac{1}{N}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1}f(x,y)(-1)^{\sum_{i=1}^{n-1}(b_i(x)b_i(u)+b_i(x)b_i(u))}$$

We define now the Inverse 2-D Hadamard transform. It is identical to the forward 2-D Hadamard transform.

$$f(x,y)=\frac{1}{N}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1}H(u,v)\left[\prod_{i=0}^{n-1}(-1)^{b_i(x)b_i(u)+b_i(y)b_i(v)}\right]$$

The above is equivalent to:

$$f(x,y)=\frac{1}{N}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1}H(u,v)(-1)^{\sum_{i=1}^{n-1}(b_i(x)b_i(u)+b_i(x)b_i(u))}$$

DISCRETE COSINE TRANSFORM (DCT):

The discrete cosine transform (DCT) helps separate the image into parts (or spectral sub-bands) of differing importance (with respect to the image's visual quality). The DCT is similar to the discrete Fourier transform: it transforms a signal or image from the spatial domain to the frequency domain.

The general equation for a 1D (*N* data items) DCT is defined by the following equation:

$$F(u) = \left(\frac{2}{N}\right)^{\frac{1}{2}} \sum_{i=0}^{N-1} \Lambda(i).cos\left[\frac{\pi.u}{2.N}(2i+1)\right] f(i)$$

and the corresponding *inverse* 1D DCT transform is simple $F^{-1}(u)$, i.e.:

where

$$\Lambda(i) = \begin{cases} \frac{1}{\sqrt{2}} & \text{for} \xi = 0 \\ 1 & \text{otherwise} \end{cases}$$

The general equation for a 2D (*N* by *M* image) DCT is defined by the following equation:

$$F(u,v) = \left(\frac{2}{N}\right)^{\frac{1}{2}} \left(\frac{2}{M}\right)^{\frac{1}{2}} \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} \Lambda(i).\Lambda(j).cos\left[\frac{\pi.u}{2.N}(2i+1)\right] cos\left[\frac{\pi.v}{2.M}(2j+1)\right] .f(i,j)$$

and the corresponding *inverse* 2D DCT transform is simple $F^{-1}(u,v)$, i.e.:

where

$$\Lambda(\xi) = \begin{cases} \frac{1}{\sqrt{2}} & \text{for} \xi = 0 \\ 1 & \text{otherwise} \end{cases}$$

The basic operation of the DCT is as follows:

- The input image is N byM;
- f(i,j) is the intensity of the pixel in row i and columnj;
- F(u,v) is the DCT coefficient in row k1 and column k2 of the DCTmatrix.
- For most images, much of the signal energy lies at low frequencies; these appear in the upper left corner of theDCT.
- Compression is achieved since the lower right values represent higher frequencies,and are often small - small enough to be neglected with little visibledistortion.
- The DCT input is an 8 by 8 array of integers. This array contains each pixel's gray scale level;
- 8 bit pixels have levels from 0 to255.

DISCRETE WAVELET TRANSFORM (DWT):

There are many discrete wavelet transforms they are Coiflet, Daubechies, Haar, Symmlet etc.

Haar Wavelet Transform

The Haar wavelet is the first known wavelet. The Haar wavelet is also the simplest possible wavelet. The Haar Wavelet can also be described as a step function f(x) shown in Eq

$$f(x) = \begin{cases} 1 & 0 \le x < 1/2, \\ -1 & 1/2 \le x < 1, \\ 0 & otherwise. \end{cases}$$

Each step in the one dimensional Haar wavelet transform calculates a set of wavelet coefficients (Hi-D) and a set of averages (Lo-D). If a data set $s_0, s_1, \ldots, s_{N-1}$ contains N elements, there will be N/2 averages and N/2 coefficient values. The averages are stored in the lower half of the N element array and the coefficients are stored in the upperhalf.

The Haar equations to calculate an average (a_i) and a wavelet coefficient (c_i) from the data set are shown below Eq

$$a_i = \frac{s_i + s_{i+1}}{2} \qquad c_i = \frac{s_i - s_{i+1}}{2}$$

In wavelet terminology the Haar average is calculated by the scaling function. The coefficient is calculated by the wavelet function.

Two-Dimensional Wavelets

The two-dimensional wavelet transform is separable, which means we can apply a one-dimensional wavelet transform to an image. We apply one-dimensional DWT to all rows and then one-dimensional DWTs to all columns of the result. This is called the standard decomposition and it is illustrated in figure.

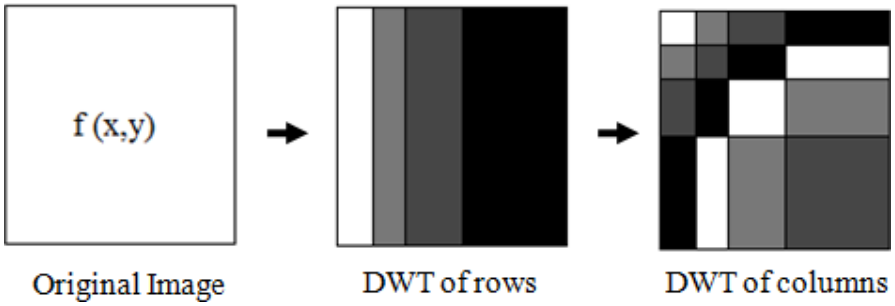


Fig: The standard decomposition of the two-dimensional DWT.

We can also apply a wavelet transform differently. Suppose we apply a wavelet transform to an image by rows, then by columns, but using our transform at one scale only. This technique will produce a result in four quarters: the top left will be a half-sized version of the image and the other quarter's high-pass filtered images. These quarters will contain horizontal, vertical, and

diagonal edges of the image. We then apply a one-scale DWT to the top-left quarter, creating

Smaller images, and so on. This is called the nonstandard decomposition, and is illustrated in figure.

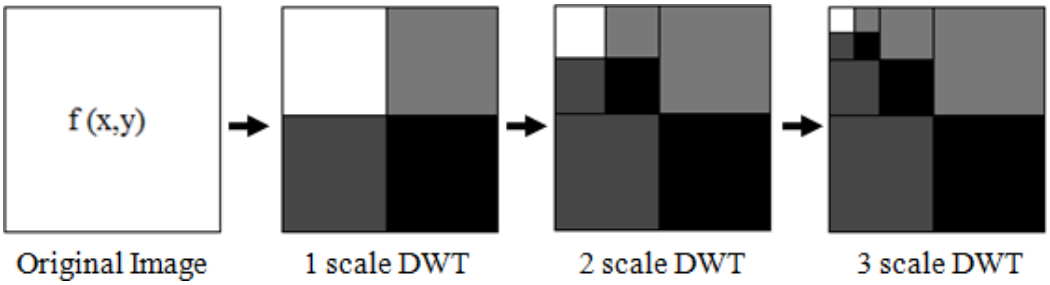


Fig: Non-standard decomposition of the two-dimensional DWT.

Steps for performing a one-scale wavelet transform are given below:

Step 1: Convolve the image rows with the low-pass filter.

Step 2 : Convolve the columns of the result of step 1 with the low-pass filter and rescale this to half its size by sub-sampling.

Step 3 : Convolve the result of step 1 with high-pass filter and again sub-sample to obtain an image of half the size.

Step 4 : Convolve the original image rows with the high-pass filter.

Step 5: Convolve the columns of the result of step 4 with the low-pass filter and recycle this to half its size by sub-sampling.

Step 6: Convolve the result of step 4 with the high-pass filter and again sub-sample to obtain an image of half the size.

At the end of these steps there are four images, each half the size of original. They are

1. The low-pass / low-pass image (LL), the result of step2,
2. The low-pass / high-pass image (LH), the result of step3,
3. The high-pass / low-pass image (HL), the result of step 5,and
4. The high-pass / high-pass image (HH), the result of step6

These images can be placed into a single image grid as shown in the figure.

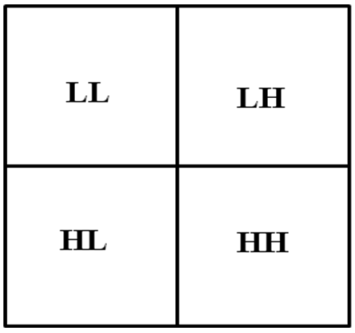


Fig: one-scale wavelet transforms in terms of filters.

Figure describes the basic dwt decomposition steps for an image in a block diagram form. The two-dimensional DWT leads to a decomposition of image into four components CA, CH, CV and CD, where CA are approximation and CH, CV, CD are details in three orientations (horizontal, vertical, and diagonal), these are same as LL, LH, HL, and HH. In these coefficients the watermark can be embedded.

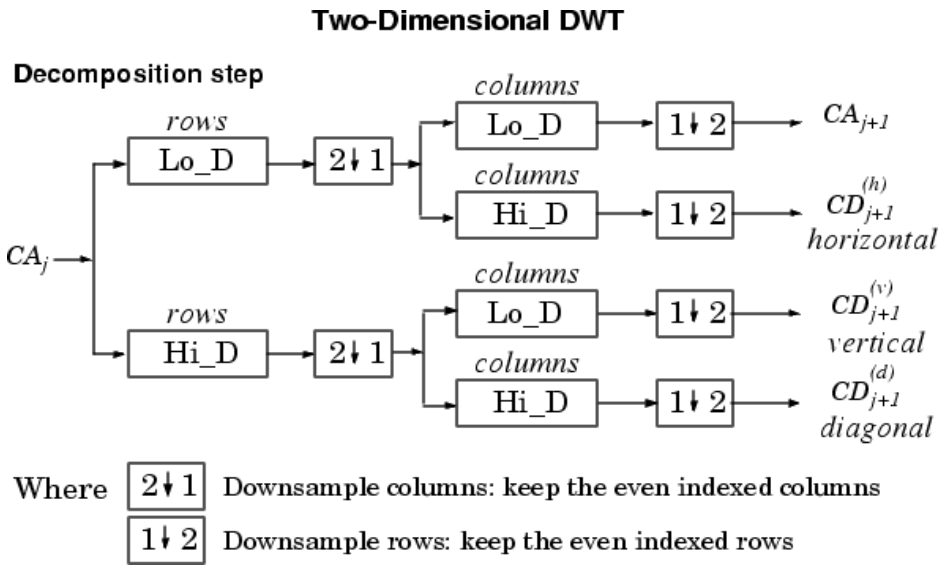


Fig: DWT decomposition steps for an image.

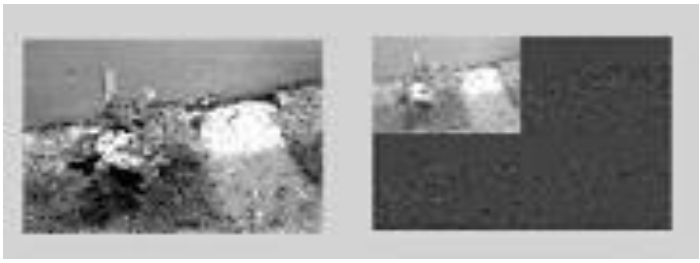


Fig: Original image and DWT decomposed image.

An example of a discrete wavelet transform on an image is shown in Figure above. On the left is the original image data, and on the right are the coefficients after a single pass of the wavelet transform. The low-pass data is the recognizable portion of the image in the upper left corner. The high-pass components are almost invisible because image data contains mostly low frequency information.

UNIT -II

IMAGE ENHANCEMENT

Image enhancement (spatial domain) : Introduction, Image Enhancement in Spatial Domain, Enhancement Through Point Operation, Types of Point Operation, Histogram Manipulation, gray level Transformation, local or neighborhood operation, median filter, spatial domain high- pass filtering.

Image enhancement (Frequency domain): Filtering in Frequency Domain, Obtaining Frequency Domain Filters from Spatial Filters, Generating Filters Directly in the Frequency Domain, Low Pass (smoothing) and High Pass (sharpening) filters in Frequency Domain

Image enhancement in Spatial Domain

Image enhancement approaches fall into two broad categories: spatial domain methods and frequency domain methods. The term spatial domain refers to the image plane itself, and approaches in this category are based on direct manipulation of pixels in an image.

Frequency domain processing techniques are based on modifying the Fourier transform of an image. Enhancing an image provides better contrast and a more detailed image as compare to non enhanced image. Image enhancement has very good applications. It is used to enhance medical images, images captured in remote sensing, images from satellite e.t.c. As indicated previously, the term spatial domain refers to the aggregate of pixels composing an image. Spatial domain methods are procedures that operate directly on these pixels. Spatial domain processes will be denoted by the expression.

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

where f(x, y) is the input image, g(x, y) is the processed image, and T is an operator on f, defined over some neighborhood of (x, y). The principal approach in defining a neighborhood about a point (x, y) is to use a square or rectangular subimage area centered at (x, y), as Fig. 2.1 shows. The center of the subimage is moved from pixel to pixel starting, say, at the top left corner. The operator T is applied at each location (x, y) to yield the output, g, at that location. The process utilizes only the pixels in the area of the image spanned by the neighborhood.

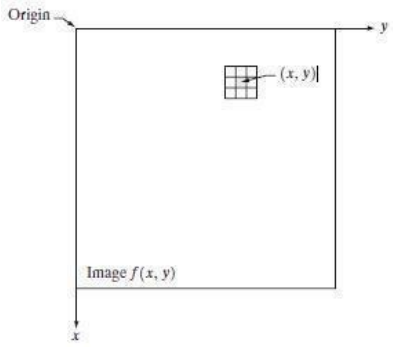


Fig.: 3x3 neighborhood about a point (x,y) in an image.

The simplest form of T is when the neighborhood is of size 1*1 (that is, a single pixel). In this case, g depends only on the value of f at (x, y), and T becomes a gray-level (also called an intensity or mapping) transformation function of the form

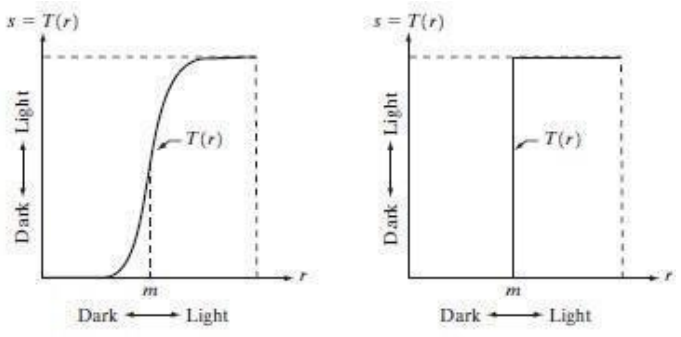
$$s = T (r)$$

where r is the pixels of the input image and s is the pixels of the output image. T is a transformation function that maps each value of 'r' to each value of 's'.

For example, if T(r) has the form shown in Fig. 2.2(a), the effect of this transformation would be to produce an image of higher contrast than the original by darkening the levels below m and brightening the levels above m in the original image. In this technique, known as contrast stretching, the values of r below m are compressed by the transformation function into a narrow range of s, toward black. The opposite effect takes place for values of r above m.

In the limiting case shown in Figure, T(r) produces a two-level (binary) image. A mapping of this form is called a thresholding function.

One of the principal approaches in this formulation is based on the use of so-called masks (also referred to as filters, kernels, templates, or windows). Basically, a mask is a small (say, 3*3) 2-D array, such as the one shown in Figure, in which the values of the mask coefficients determine the nature of the process, such as image sharpening. Enhancement techniques based on this type of



approach often are referred to as mask processing or filtering.

Fig: Gray level transformation functions for contrast enhancement.

Image enhancement can be done through gray level transformations which are discussed below.

BASIC GRAY LEVEL TRANSFORMATIONS:

- Image negative
- Log transformations
- Power law transformations
- Piecewise-Linear transformation functions

LINEAR TRANSFORMATION:

First we will look at the linear transformation. Linear transformation includes simple identity and negative transformation. Identity transformation has been discussed in our tutorial of image transformation, but a brief description of this transformation has been given here.

Identity transition is shown by a straight line. In this transition, each value of the input image is directly mapped to each other value of output image. That results in the same input image and output image. And hence is called identity transformation. It has been shown below:

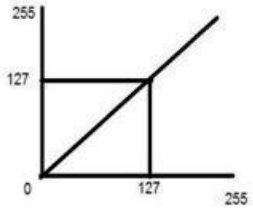


Fig. Linear transformation between input and output.

NEGATIVETRANSFORMATION:

The second linear transformation is negative transformation, which is invert of identity transformation. In negative transformation, each value of the input image is subtracted from the L-1 and mapped onto the output image

IMAGENEGATIVE: The image negative with gray level value in the range of [0,L-1] is obtained by negative transformation given by S =T(r) or

$$S = L - 1 - r$$

Where r= gray level value at pixel (x,y)

L is the largest gray level consists in the image

It results in getting photograph negative. It is useful when for enhancing white details embedded in dark regions of the image.

The overall graph of these transitions has been shown below.

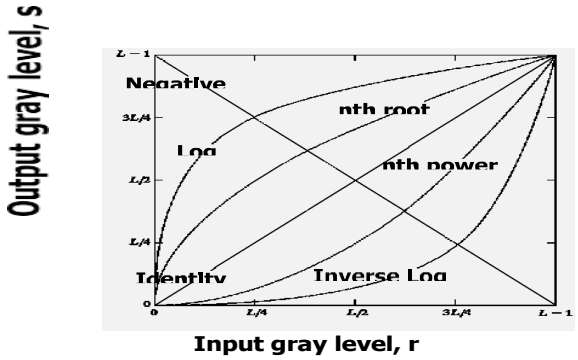


Fig. Some basic gray-level transformation functions used for image enhancement.

In this case the following transition has been done.

$$S = (L - 1) - r$$

since the input image of Einstein is an 8 bpp image, so the number of levels in this image are 256. Putting 256 in the equation, we get this

$$S = 255 - r$$

So each value is subtracted by 255 and the result image has been shown above. So what happens is that, the lighter pixels become dark and the darker picture becomes light. And it results in image negative.

It has been shown in the graph below.

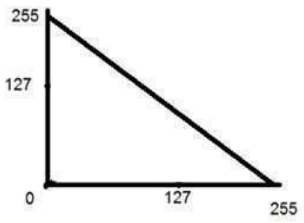


Fig. Negative transformations.

LOGARITHMIC TRANSFORMATIONS:

Logarithmic transformation further contains two type of transformation. Log transformation and inverse log transformation.

LOG TRANSFORMATIONS:

The log transformations can be defined by this formula

$$s = c \log(r + 1).$$

Where s and r are the pixel values of the output and the input image and c is a constant. The value 1 is added to each of the pixel value of the input image because if there is a pixel intensity of 0 in the image, then log (0) is equal to infinity. So 1 is added, to make the minimum value at least 1.

During log transformation, the dark pixels in an image are expanded as compare to the higher pixel values. The higher pixel values are kind of compressed in log transformation. This result in following image enhancement.

An another way of representing LOG TRANSFORMATIONS: Enhance details in the darker regions of an image at the expense of detail in brighter regions.

$$T(f) = C * \log (1+r)$$

- HereCisconstantandr≥0.
- The shape of the curve shows that this transformation maps the narrow range of low gray level values in the input image in to a wide range of output image.
- The opposite is true for high level values of input image.

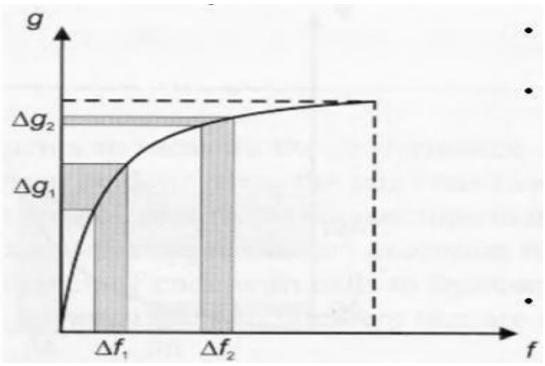


Fig. log transformation curve input vs output

POWER – LAW TRANSFORMATIONS:

There are further two transformation is power law transformations, that include nth power and nth root transformation. These transformations can be given by the expression:

$$s=cr^{\gamma}$$

This symbol γ is called gamma, due to which this transformation is also known as gamma transformation.

Variation in the value of γ varies the enhancement of the images. Different display devices / monitors have their own gamma correction, that's why they display their image at different intensity.

where c and g are positive constants. Sometimes Eq. (6) is written as $S = C (r + \varepsilon)^{\gamma}$ to account for an offset (that is, a measurable output when the input is zero). Plots of s versus r for various values of γ are shown in Figure. As in the case of the log transformation, 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 levels. Unlike the log function, however, we notice here a family of possible transformation curves obtained simply by varying γ.

In Fig that curves generated with values of γ>1 have exactly The opposite effect as those generated with values of γ<1. Finally, we Note that Eq. (6) reduces to the identity transformation when c=γ=1.

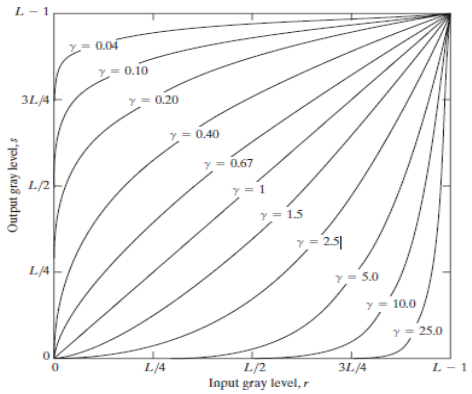


Fig: Plot of the equation $S = cr^{\gamma}$ for various values of γ (c =1 in all cases).

This type of transformation is used for enhancing images for different type of display devices. The gamma of different display devices is different. For example Gamma of CRT lies in between of 1.8 to 2.5, that means the image displayed on CRT is dark.

Varying gamma (γ) obtains family of possible transformation curves $S = C^* r^{\gamma}$

Here C and γ are positive constants. Plot of S versus r for various values of γ

is γ > 1 compresses dark values

Expands bright values

γ< 1(similar to Logtransformation)

Expands dark values

Compresses bright values

When C = γ = 1 , it reduces to identity transformation .

CORRECTING GAMMA:

$s=cr^{\gamma}$

$s=cr^{(1/2.5)}$

The same image but with different gamma values has been shown here.

Piecewise-Linear Transformation Functions:

A complementary approach to the methods discussed in the previous three sections is to use piecewise linear functions. The principal advantage of piecewise linear functions over the types of functions which we have discussed thus far is that the form of piecewise functions can be arbitrarily complex.

The principal disadvantage of piecewise functions is that their specification requires considerably more user input.

Contrast stretching: One of the simplest piecewise linear functions is a contrast-stretching transformation. Low-contrast images can result from poor illumination, lack of dynamic range in the imaging sensor, or even wrong setting of a lens aperture during image acquisition.

$$S= T(r)$$

Figure x(a) shows a typical transformation used for contrast stretching. The locations of points (r1, s1) and (r2, s2) control the shape of the transformation

Function. If r1=s1 and r2=s2, the transformation is a linear function that produces No changes in gray levels. If r1=r2, s1=0and s2= L-1, the transformation Becomes a thresholding function that creates a binary image, as illustrated In fig. 2.2(b).

Intermediate values of ar1, s1b and ar2, s2b produce various degrees Of spread in the gray levels of the output image, thus affecting its contrast. In general, $r1 \leq r2$ and $s1 \leq s2$ is assumed so that the function is single valued and monotonically increasing.

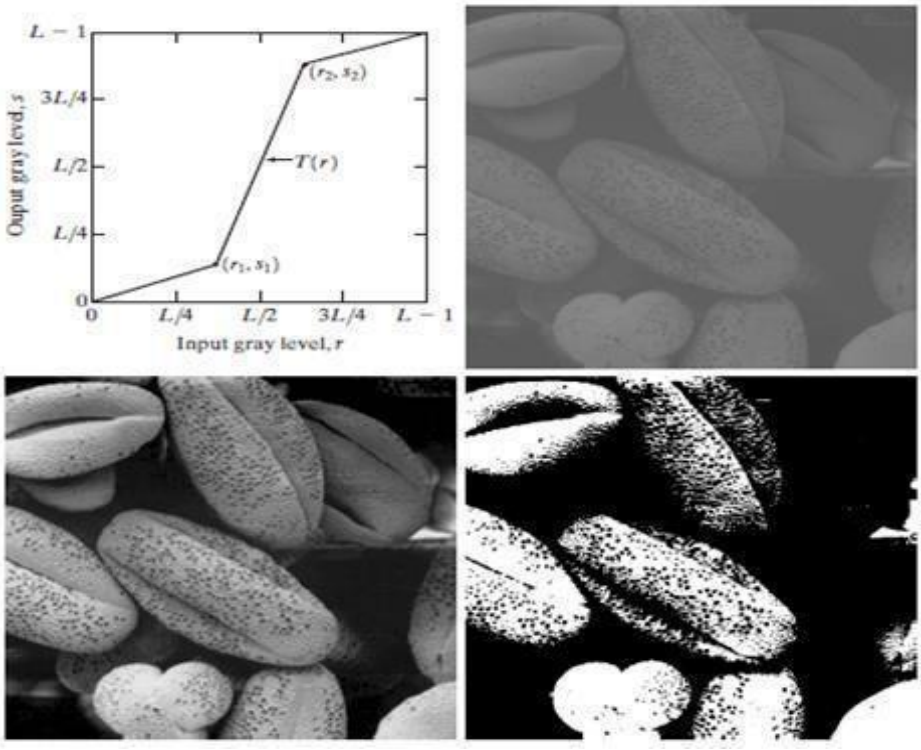


Fig: Contrast stretching. (a) Form of transformation function. (b) A low-contrast stretching. (c) Result of high contrast stretching. (d) Result of thresholding (original image courtesy of Dr.Roger Heady, Research School of Biological Sciences, Australian National University Canberra Australia.

Figure x(b) shows an 8-bit image with low contrast. Fig. x(c) shows the result of contrast stretching, obtained by setting (r1, s1)=(rmin, 0) and (r2, s2)=(r_{max},L-1) where r_{min} and r_{max} denote the minimum and maximum gray levels in the image, respectively.Thus, the transformation function stretched the levels linearly from their original range to the full range [0, L-1]. Finally, Fig. x(d) shows the result of using the thresholding function defined previously, with r1=r2=m, the mean gray level in the image. The original image on which these results are based is a scanning electron microscope image of pollen, magnified approximately 700 times.

Gray-level slicing:

Highlighting a specific range of gray levels in an image often is desired. Applications include enhancing features such as masses of water in satellite imagery and enhancing flaws in X-ray images.

There are several ways of doing level slicing, but most of them are variations of two basic themes. One approach is to display a high value for all gray levels in the range of interest and a

low value for all other gray levels.

This transformation, shown in Fig. y(a), produces a binary image. The second approach, based on the transformation shown in Fig.y (b), brightens the desired range of gray levels but preserves the background and gray-level tonalities in the image. Figure y (c) shows a gray-scale image, and Fig. y(d) shows the result of using the transformation in Fig. y(a).Variations of the two transformations shown in Fig. are easy to formulate.

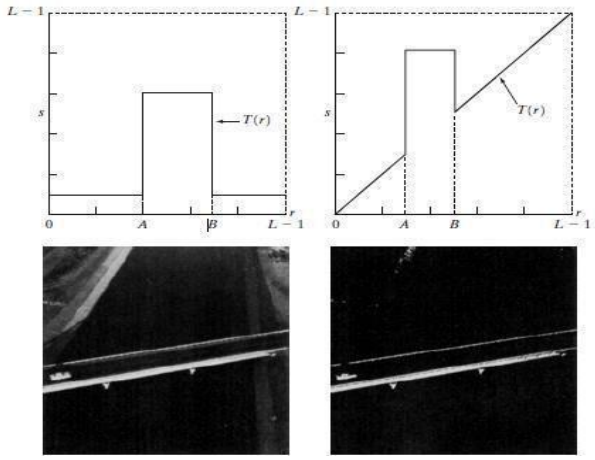


Fig. y (a)This transformation highlights range [A,B] of gray levels and reduces all others to a constant level (b) This transformation highlights range [A,B] but preserves all other levels. (c) An image . (d) Result of using the transformation in(a).

BIT-PLANE SLICING:

Instead of highlighting gray-level ranges, highlighting the contribution made to total image appearance by specific bits might be desired. Suppose that each pixel in an image is represented by 8 bits. Imagine that the 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 bytes comprising the pixels in the image and plane 7 contains all the high-orderbits.

Figure 3.12 illustrates these ideas, and Fig. 3.14 shows the various bit planes for the image shown in Fig. 3.13. Note that the higher-order bits (especially the top four) contain the majority of the visually significant data. The other bit planes contribute to more subtle details in the image. Separating a digital image into its bit planes is useful for analyzing the relative importance played by each bit of the image, a process that aids in determining the adequacy of the number of bits used to quantize each pixel.

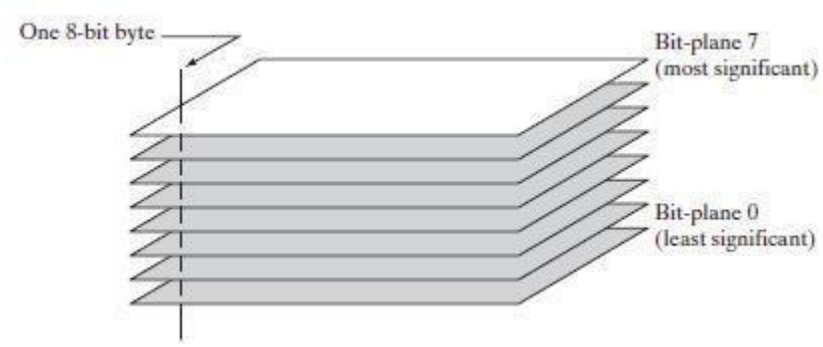


FIGURE Bit-plane representation of an 8-bit image.

In terms of bit-plane extraction for an 8-bit image, it is not difficult to show that the (binary) image for bit-plane 7 can be obtained by processing the input image with a thresholding gray-level transformation function that (1) maps all levels in the image between 0 and 127 to one level (for example, 0); and (2) maps all levels between 129 and 255 to another (for example, 255).The binary image for bit-plane 7 in Figure was obtained in just this manner.

Histogram Processing:

The histogram of a digital image with gray levels in the range [0, L-1] is a discrete function of the form

$$H(r_k)=n_k$$

where r_k is the k^{th} gray level and n_k is the number of pixels in the image having the level r_k..

A normalized histogram is given by the equation

$$p(r_k)=n_k/n \text{ for } k=0,1,2,\ldots..,L-1$$

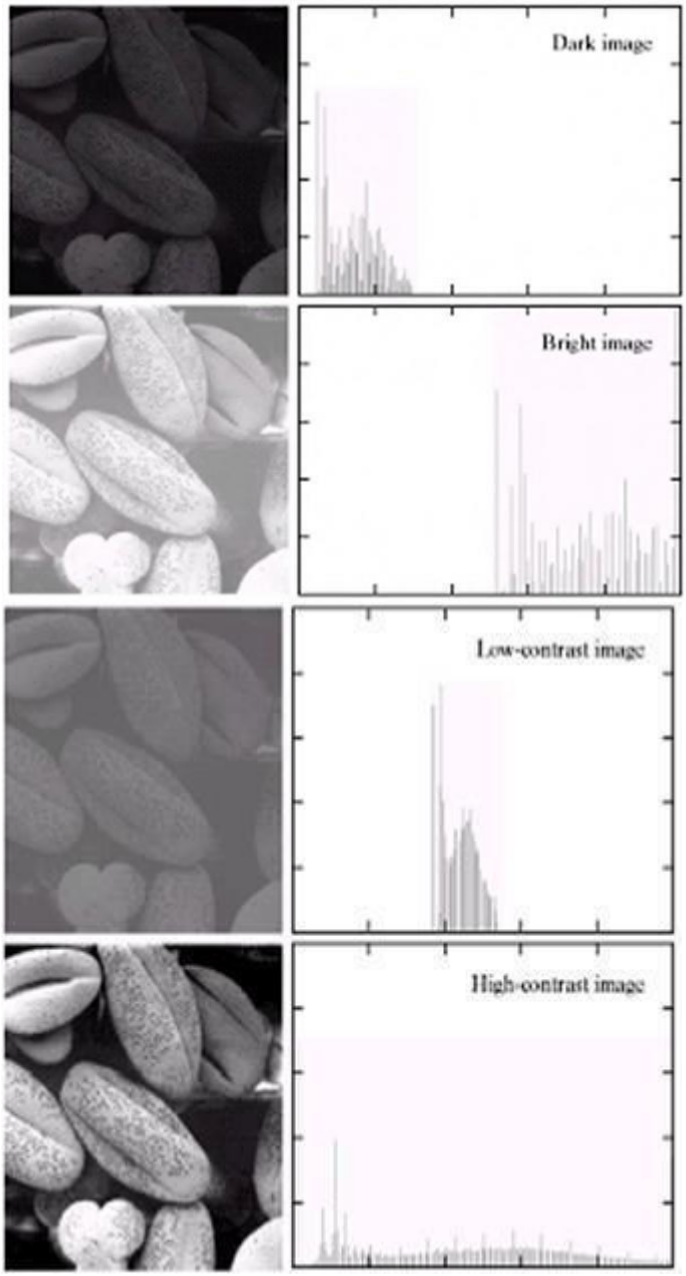
$P(r_k)$ gives the estimate of the probability of occurrence of gray level r_k.

The sum of all components of a normalized histogram is equal to 1.

The histogram plots are simple plots of $p(r_k)=n_k$ versus r_k.

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 baised 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 grayscale.

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.



Histogram Equalization:

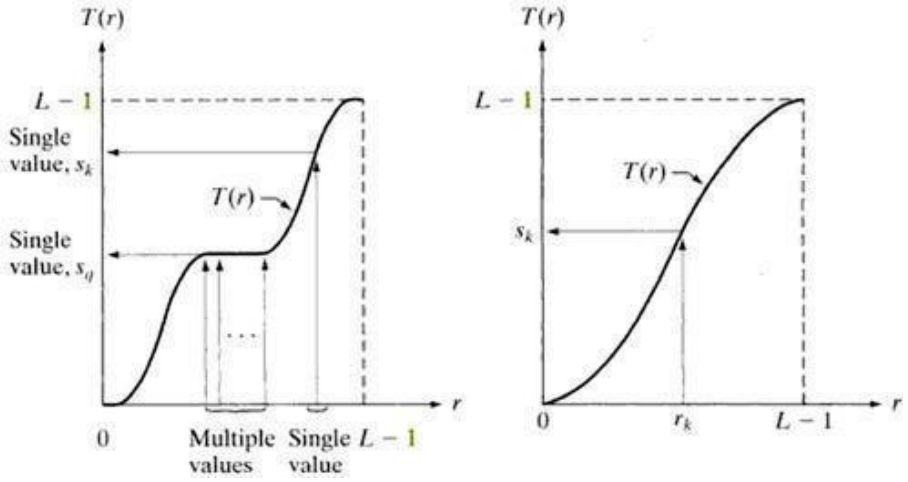
Histogram equalization is a common technique for enhancing the appearance of images. Suppose we have an image which is predominantly dark. Then its histogram would be skewed towards the lower end of the grey scale and all the image detail are compressed into the dark endof the histogram. If we could ‘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 representing black and r=1 representing white. The transformation function is of the form

S=T(r) where 0<r<1

It produces a level s for every pixel value r in the original image.



a b

FIGURE (a) Monotonically increasing function, showing how multiple values can map to a single value. (b) Strictly monotonically increasing function. This is a one-to-one mapping, both ways.

The transformation function is assumed to fulfill two condition T(r) is single valued and monotonically increasing in the internal 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 is given by the formula

$$p_s(s) = p_r(r)\left|\frac{dr}{ds}\right|$$

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.

A transformation function of a particular importance in image processing

$$s = T(r) = (L - 1)\int_0^r p_r(w)\,dw$$

This is the cumulative distribution function of r.

L is the total number of possible gray levels in the image.

IMAGE ENHANCEMENT IN FREQUENCY DOMAIN

BLURRING/NOISE REDUCTION: Noise characterized by sharp transitions in image intensity. Such transitions contribute significantly to high frequency components of Fourier transform. Intuitively, attenuating certain high frequency components result in blurring and reduction of image noise.

IDEAL LOW-PASS FILTER:

Cuts off all high-frequency components at a distance greater than a certain distance from origin (cutoff frequency).

$$H(u,v) = 1, \text{ if } D(u,v) \le D_0$$

$$0, \text{ if } D(u,v) > D_0$$

Where D0 is a positive constant and D(u,v) is the distance between a point (u,v) in the frequency domain and the center of the frequency rectangle; that is

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

Where as P and Q are the padded sizes from the basic equations

Wraparound error in their circular convolution can be avoided by padding these functions with zeros,

VISUALIZATION: IDEAL LOW PASS FILTER:

As shown in fig.below

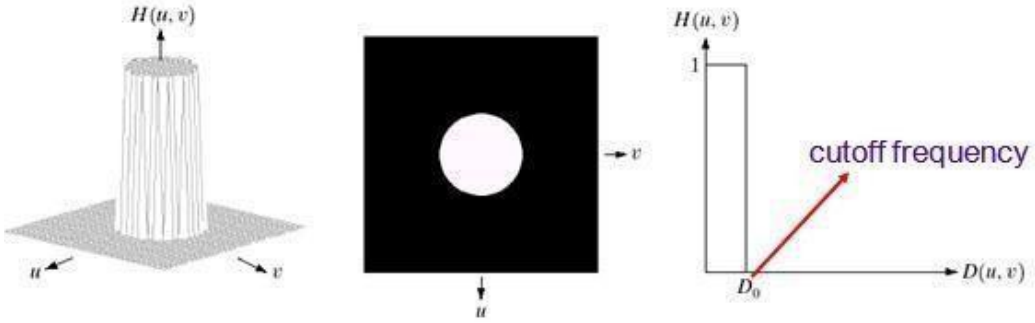


Fig: ideal low pass filter 3-D view and 2-D view and line graph.

EFFECT OF DIFFERENT CUT OFF FREQUENCIES:

Fig.below(a) Test pattern of size 688x688 pixels, and (b) its Fourier spectrum. The spectrum is double the image size due to padding but is shown in half size so that it fits in the page. The superimposed circles have radii equal to 10, 30, 60, 160 and 460 with respect to the full-size spectrum image. These radii enclose 87.0, 93.1, 95.7, 97.8 and 99.2% of the padded image power respectively.

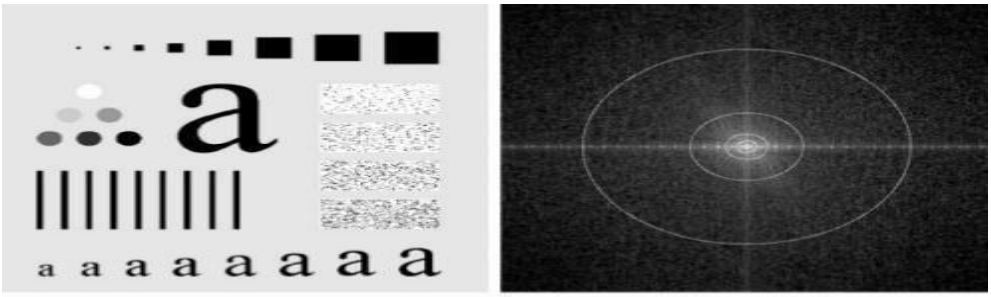


Fig: (a) Test pattern of size 688x688 pixels (b) its Fourier spectrum

Fig: (a) original image, (b)-(f) Results of filtering using ILPFs with cutoff frequencies set

at radii values 10, 30, 60, 160 and 460, as shown in figure. The power removed by these filters was 13, 6.9, 4.3, 2.2 and 0.8% of the total, respectively.

As the cutoff frequency decreases,

- image becomes more blurred
- Noise becomes increases
- Analogous to larger spatial filter sizes

The severe blurring in this image is a clear indication that most of the sharp detail information in the picture is contained in the 13% power removed by the filter. As the filter radius is increases less and less power is removed, resulting in less blurring. Fig. (c) through (e) are characterized by "ringing" , which becomes finer in texture as the amount of high frequency content removed decreases.

WHY IS THERE RINGING?

Ideal low-pass filter function is a rectangular function

The inverse Fourier transform of a rectangular function is a sinc function.

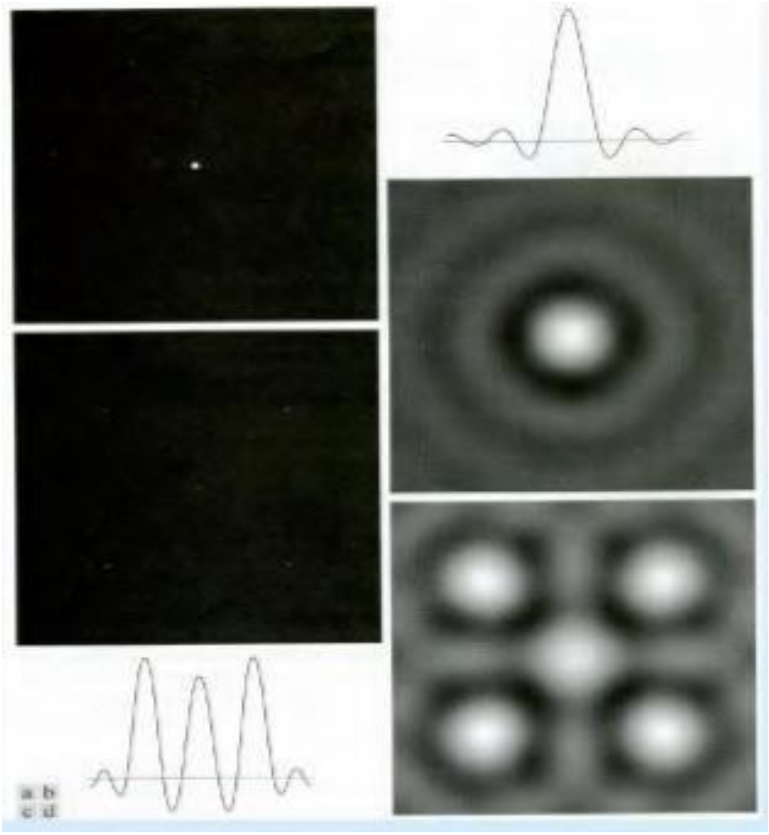


Fig. Spatial representation of ILPFs of order 1 and 20 and corresponding intensity profiles through the center of the filters(the size of all cases is 1000x1000 and the cutoff frequency is 5),

observe how ringing increases as a function of filter order.

BUTTERWORTH LOW-PASS FILTER:

Transfer function of a Butterworth low pass filter (BLPF) of order n, and with cutoff frequency at a distance D_0 from the origin, is defined as

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

Transfer function does not have sharp discontinuity establishing cutoff between passed and filtered frequencies.

Cut off frequency D_0 defines point at which H(u,v) = 0.5

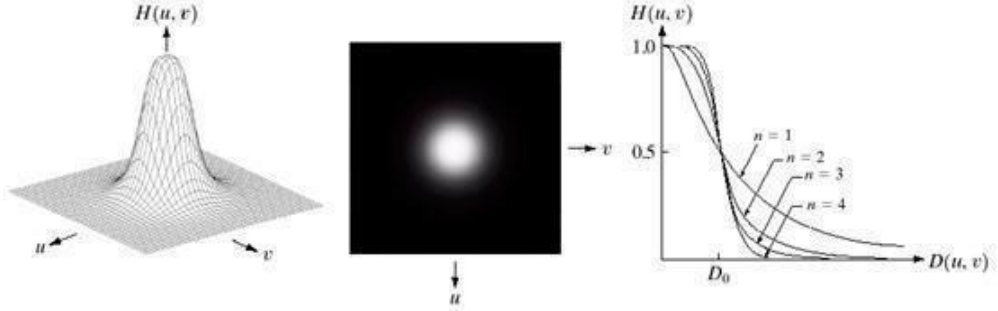


Fig. (a) Perspective plot of a Butterworth low pass-filter transfer function. (b) Filter displayed as an image. (c)Filter radial cross sections of order 1 through 4.

Unlike the ILPF, the BLPF transfer function does not have a sharp discontinuity that gives a clear cutoff between passed and filtered frequencies.

BUTTERWORTH LOW-PASS FILTERS OF DIFFEREN T FREQUENCIES:

Fig. (a) Original image.(b)-(f) Results of filtering using BLPFs of order 2, with cutoff frequencies at the radii

Fig. shows the results of applying the BLPF of eq. to fig.(a), with n=2 and D0 equal to the five radii in fig.(b) for the ILPF, we note here a smooth transition in blurring as a function of increasing cutoff frequency. Moreover, no ringing is visible in any of the images processed with this particular BLPF, a fact attributed to the filter's smooth transition between low and high frequencies.

A BLPF of order 1 has no ringing in the spatial domain. Ringing generally is imperceptible in filters of order 2, but can become significant in filters of higher order.

Fig.shows a comparison between the spatial representation of BLPFs of various orders (using a cutoff frequency of 5 in all cases). Shown also is the intensity profile along a horizontal scan line through the center of each filter. The filter of order 2 does show mild ringing and small negative values, but they certainly are less pronounced than in the ILPF. A butter worth filter of order 20 exhibits characteristics similar to those of the ILPF (in the limit, both filters are identical).

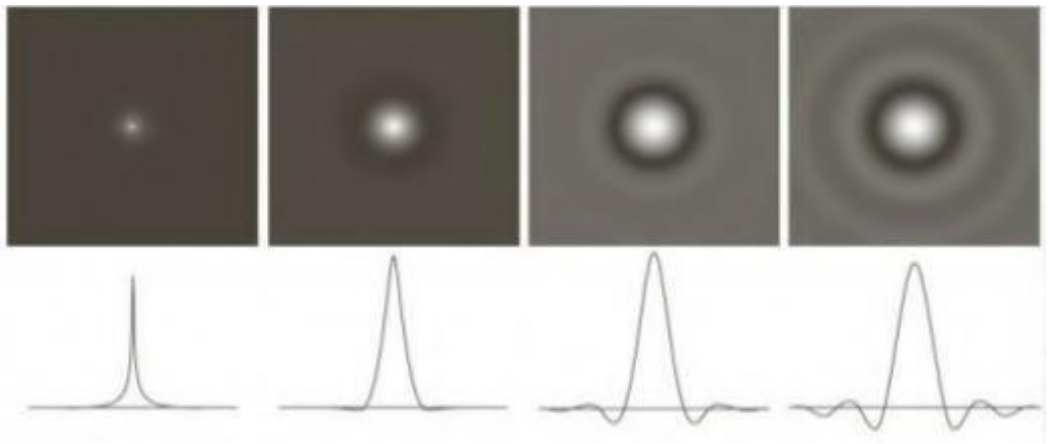


Fig.2.2.7 (a)-(d) Spatial representation of BLPFs of order 1, 2, 5 and 20 and corresponding intensity profiles through the center of the filters (the size in all cases is 1000 x 1000 and the cutoff frequency is 5) Observe how ringing increases as a function of filter order.

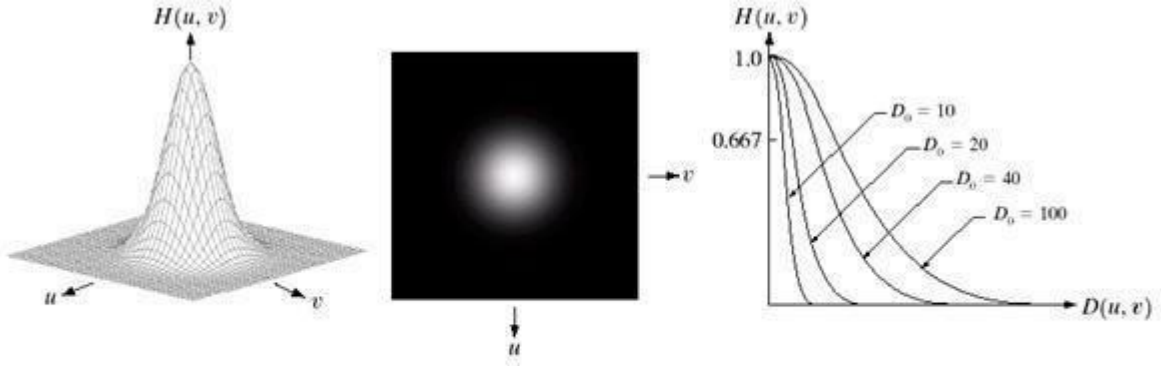
GAUSSIAN LOWPASS FILTERS:

The form of these filters in two dimensions is given by

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

- This transfer function is smooth, like Butterworth filter.
- Gaussian in frequency domain remains a Gaussian in spatial domain
- Advantage: No ringing artifacts.

Where D_0 is the cutoff frequency. When $D(u,v) = D_0$, the GLPF is down to 0.607 of its maximum value. This means that a spatial Gaussian filter, obtained by computing the IDFT of above equation., will have no ringing. Fig..Shows a perspective plot, image display and radial cross sections of a GLPF function.



a b c

FIGURE (a) Perspective plot of a GLPF transfer function. (b) Filter displayed as an image. (c) Filter radial cross sections for various values of D_0.

Fig. (a) Perspective plot of a GLPF transfer function. (b) Filter displayed as an image. (c). Filter radial cross sections for various values of D_0

Fig.(a) Original image. (b)-(f) Results of filtering using GLPFs with cutoff frequencies at the radii shown in figure.

Fig. (a) Original image (784x 732 pixels). (b) Result of filtering using a GLPF with D0 = 100. (c) Result of filtering using a GLPF with D0 = 80. Note the reduction in fine skin lines in the

magnified sections in (b) and(c).

Fig. shows an application of lowpass filtering for producing a smoother, softer-looking result from a sharp original. For human faces, the typical objective is to reduce the sharpness of fine skin lines and small blemished.

IMAGE SHARPENING USING FREQUENCY DOMAIN FILTERS:

An image can be smoothed by attenuating the high-frequency components of its Fourier transform. Because edges and other abrupt changes in intensities are associated with high-frequency components, image sharpening can be achieved in the frequency domain by high pass filtering, which attenuates the low-frequency components without disturbing high-frequency information in the Fourier transform.

The filter function H(u,v) are understood to be discrete functions of size PxQ; that is the discrete frequency variables are in the range u = 0,1,2,…….P-1 and v = 0,1,2,…….Q-1.

The meaning of sharpening is

- Edges and fine detail characterized by sharp transitions in image intensity
- Such transitions contribute significantly to high frequency components of Fourier transform

- Intuitively, attenuating certain low frequency components and preserving high frequency components result in sharpening.

Intended goal is to do the reverse operation of low-pass filters

- When low-pass filter attenuated frequencies, high-pass filter passes them
- When high-pass filter attenuates frequencies, low-pass filter passes them.

A high pass filter is obtained from a given low pass filter using the equation.

$$H_{hp}(u,v) = 1 - H_{tp}(u,v)$$

Where $H_{lp}(u,v)$ is the transfer function of the low-pass filter. That is when the low-pass filter attenuates frequencies; the high-pass filter passed them, and vice-versa.

We consider ideal, Butter-worth, and Gaussian high-pass filters. As in the previous section, we illustrate the characteristics of these filters in both the frequency and spatial domains. Fig..Shows typical 3-D plots, image representations and cross sections for these filters. As before, we see that the Butter-worth filter represents a transition between the sharpness of the ideal filter and the broad smoothness of the Gaussian filter. Fig. Discussed in the sections the follow, illustrates what these filters look like in the spatial domain. The spatial filters were obtained and displayed by using the procedure used.

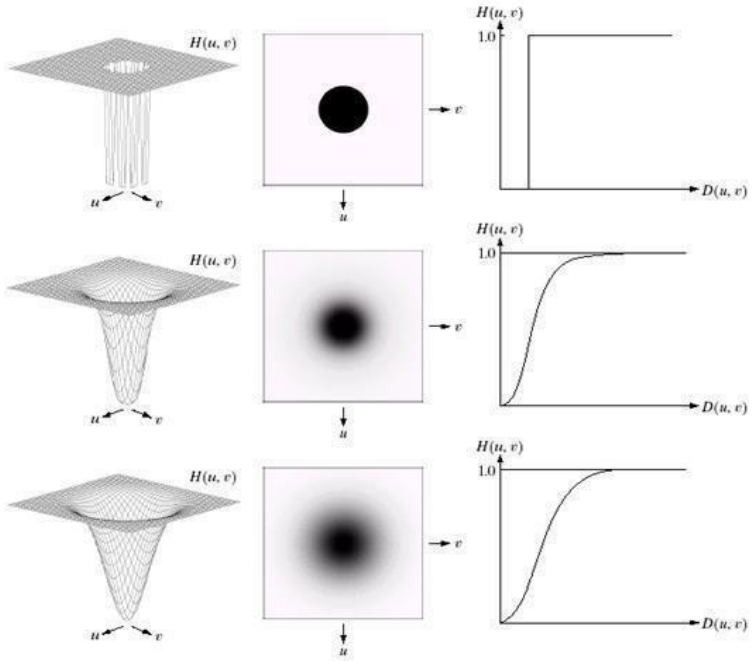


Fig: Top row: Perspective plot, image representation, and cross section of a typical ideal high-pass filter. Middle and bottom rows: The same sequence for typical butter-worth and Gaussian high-pass filters.

IDEAL HIGH-PASS FILTER:

A 2-D ideal high-pass filter (IHPF) is defined as

$$H(u,v) = 0, \text{ if } D(u,v) \leq D_0$$

$$1, \text{ if } D(u,v) > D_0$$

Where D_0 is the cutoff frequency and D(u,v) is given by eq. As intended, the IHPF is the opposite of the ILPF in the sense that it sets to zero all frequencies inside a circle of radius D_0while passing, without attenuation, all frequencies outside the circle. As in case of the ILPF, the

IHPF is not physically realizable.

SPATIAL REPRESENTATION OF HIGHPASS FILTERS:

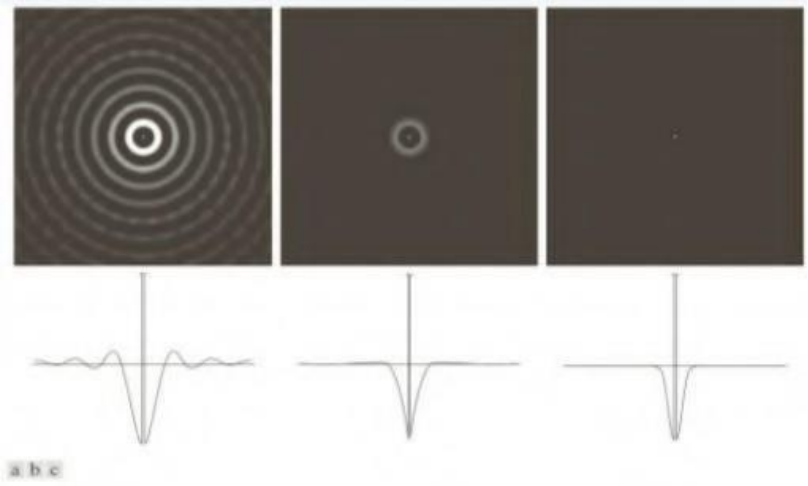


Fig..Spatial representation of typical (a) ideal (b) Butter-worth and (c) Gaussian frequency domain high-pass filters, and corresponding intensity profiles through their centers.

We can expect IHPFs to have the same ringing properties as ILPFs. This is demonstrated clearly in Fig..which consists of various IHPF results using the original image in Fig.(a) with D_0 set to 30, 60,and 160 pixels, respectively. The ringing in Fig. (a) is so severe that it produced distorted, thickened object boundaries (e.g.,look at the large letter “a”). Edges of the top three circles do not show well because they are not as strong as the other edges in the image (the intensity of these three objects is much closer to the background intensity, giving discontinuities of smaller magnitude).

Fig.. Results of high-pass filtering the image in Fig.(a) using an IHPF with $D_0 = 30$, 60, and 160.

The situation improved somewhat with D0 = 60. Edge distortion is quite evident still, but now we begin to see filtering on the smaller objects. Due to the now familiar inverse relationship between the frequency and spatial domains, we know that the spot size of this filter is smaller than the spot of the filter with D0 = 30. The result for D0 = 160 is closer to what a high-pass filtered image should look like. Here, the edges are much cleaner and less distorted, and the smaller objects have been filtered properly.

Of course, the constant background in all images is zero in these high-pass filtered images because high pass filtering is analogous to differentiation in the spatial domain.

BUTTER-WORTH HIGH-PASS FILTERS:

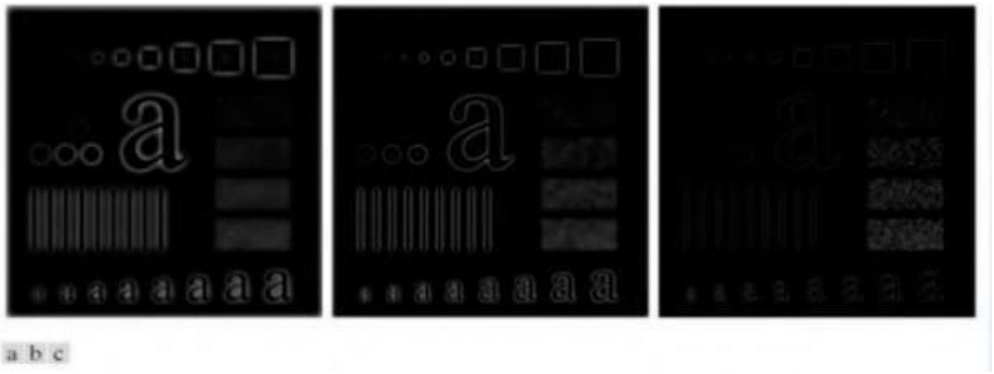
A 2-D Butter-worth high-pass filter (BHPF) of order n and cutoff frequency D_0 is defined as

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

Where D(u,v) is given by Eq.(3). This expression follows directly from Eqs.(3) and (6). The middle row of Fig.2.2.11.shows an image and cross section of the BHPF function.

Butter-worth high-pass filters to behave smoother than IHPFs. Fig.2.2.14.shows the performance of a BHPF of order 2 and with D0 set to the same values as in Fig.2.2.13. The boundaries are much less distorted than in Fig.2.2.13. Even for the smallest value of cutoff frequency.

FILTERED RESULTS: BHPF:



a b c

Fig. Results of high-pass filtering the image in Fig.2.2.2(a) using a BHPF of order 2 with D_0 = 30, 60, and 160 corresponding to the circles in Fig.2.2.2(b). These results are much smoother than those obtained with an IHPF.

GAUSSIAN HIGH-PASS FILTERS:

The transfer function of the Gaussian high-pass filter (GHPF) with cutoff frequency locus at a distance D_0 from the center of the frequency rectangle is given by

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

Where D(u,v) is given by Eq.(4). This expression follows directly from Eqs.(2) and (6). The third row in Fig.2.2.11.shows a perspective plot, image and cross section of the GHPF function. Following the same format as for the BHPF, we show in Fig.2.2.15. Comparable results using GHPFs. As expected, the results obtained are more gradual than with the previous two filters.

FILTERED RESULTS: GHPF:

a b c

Fig. Results of high-pass filtering the image in fig.(a) using a GHPF with D0 = 30, 60 and 160, corresponding to the circles in Fig.(b).

UNIT-III

IMAGE RESTORATION

Image Restoration: Degradation Model, Algebraic Approach to Restoration, Inverse Filtering, Least Mean Square Filters, Constrained Least Squares Restoration.

IMAGE RESTORATION:

Restoration improves image in some predefined sense. It is an objective process. Restoration attempts to reconstruct an image that has been degraded by using a priori knowledge of the degradation phenomenon. These techniques are oriented toward modeling the degradation and then applying the inverse process in order to recover the original image. Restoration techniques are based on mathematical or probabilistic models of image processing. Enhancement, on the other hand is based on human subjective preferences regarding what constitutes a "good" enhancement result. 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:

- During display mode
- Acquisition mode, or
- Processing mode
    - Sensor noise
    - Blur due to camera misfocus
    - Relative object-camera motion
    - Random atmospheric turbulence
- Others

Degradation Model:

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.

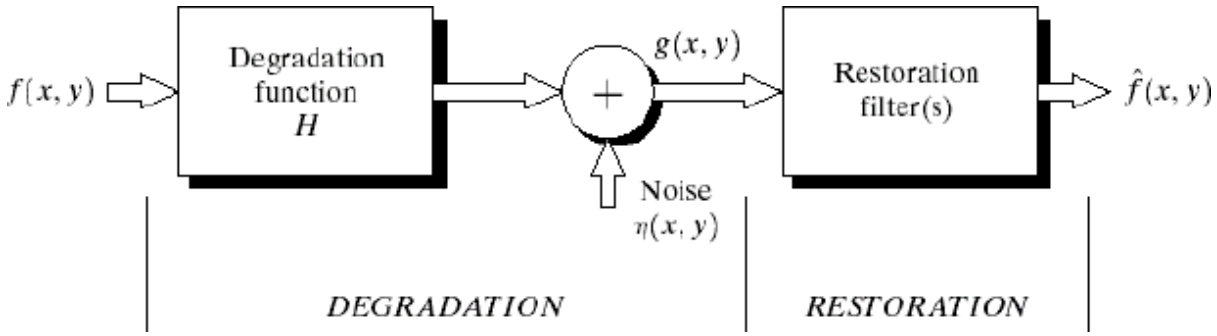


Fig: A model of the image Degradation / Restoration process

Noise Models:

The principal source of noise in digital images arises during image acquisition and /or transmission. The performance of imaging sensors is affected by a variety of factors, such as environmental conditions during image acquisition and by the quality of the sensing elements themselves. Images are corrupted during transmission principally due to interference in the channels used for transmission. Since main sources of noise presented in digital images are resulted from atmospheric disturbance and image sensor circuitry, following assumptions can be made i.e. the noise model is spatial invariant (independent of spatial location). The noise model is uncorrelated with the object function.

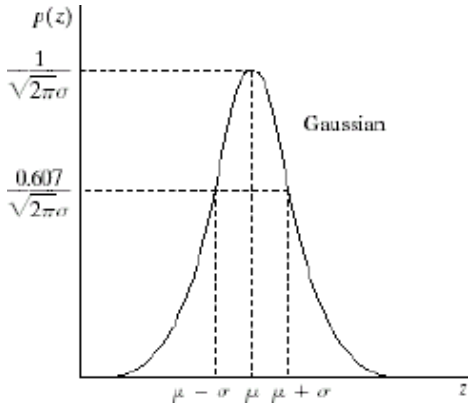
Gaussian Noise:

These noise models are used frequently in practices because of its tractability in both spatial and

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

frequency domain. The PDF of Gaussian random variable is

Where z represents the gray level, μ= mean of average value of z, σ= standard deviation.



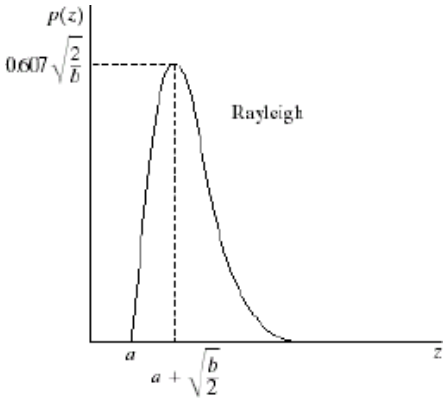
Rayleigh Noise:

Unlike Gaussian distribution, the Rayleigh distribution is no symmetric. It is given by the formula.

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

The mean and variance of this density is

$$m = a + \sqrt{\pi b/4}, \sigma^2 = \frac{b(4-\pi)}{4}$$



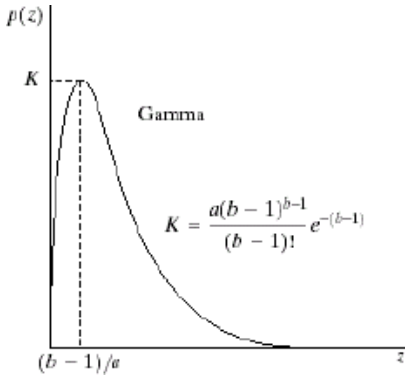
(iii) GammaNoise:

The PDF of Erlang noise is given by

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

The mean and variance of this density are given by

$$\text{mean}: \mu = \frac{b}{a} \quad \text{variance}: \sigma^2 = \frac{b}{a^2}$$



Its shape is similar to Rayleigh disruption. This equation is referred to as gamma density it is correct only when the denominator is the gamma function.

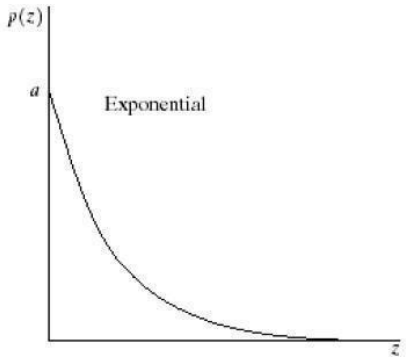
(iv) Exponential Noise:

Exponential distribution has an exponential shape. The PDF of exponential noise is given as

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

Where a>0. The mean and variance of this density are given by

$$m = \frac{1}{a}, \quad \sigma^2 = \frac{1}{a^2}$$



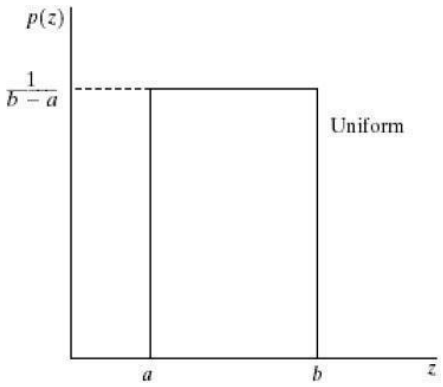
(v) Uniform Noise:

The PDF of uniform noise is given by

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

The mean and variance of this noise is

$$m = \frac{a+b}{2}, \quad \sigma^2 = \frac{(b-a)^2}{12}$$



(vi) Impulse (salt & pepper)Noise:

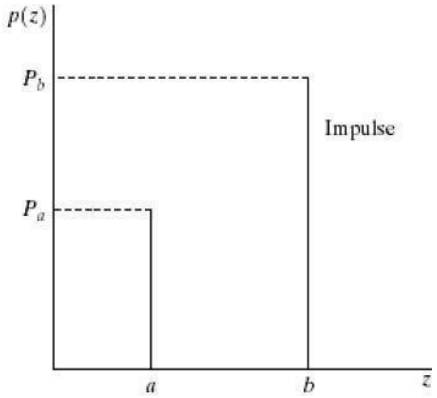
In this case, the noise is signal dependent, and is multiplied to the image.

The PDF of bipolar (impulse) noise is given by

$$p_z(z) = \begin{cases} P_a & \text{for } z = a \\ P_b & \text{for } z = b \\ 0 & \text{otherwise} \end{cases}$$

$$b > a$$

If b>a, gray level b will appear as a light dot in image. Level a will appear like a dark dot.



Restoration in the presence of Noise only- Spatial filtering:

When the only degradation present in an image is noise, i.e.

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

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:

i) MeanFilter:

ii) (a)Arithmetic Mean filter:

It is the simplest mean filter. Let Sxy 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 Sxy. 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 Sxy.

$$\hat{f}(x,y)=\frac{1}{mn}\sum_{(s,t)\in Sxy} g(s,t)$$

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. For every pixel in the image, the pixel value is replaced by the mean value of its neighboring pixels with a weight .This will resulted in a smoothing effect in the image.

(b)Geometric Mean filter:

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

$$\hat{f}(x,y)=\left[\prod_{(s,t)\in Sxy} g(s,t)\right]^{1/mn}$$

Here, each restored pixel is given by the product of the pixel in the sub image window, raised to the power 1/mn. A geometric mean filters but it to loose image details in the process.

(c) Harmonic Mean filter:

The harmonic mean filtering operation is given by the expression

$$\hat{f}(x,y) = \sum_{(s,t)\in Sxy} g(s,t)^{Q+1} \Big/ \sum_{(s,t)\in Sxy} g(s,t)^{Q}$$

The harmonic mean filter works well for salt noise but fails for pepper noise. It does well with Gaussian noise also.

(d) Order statistics filter:

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.

(e) Median filter:

It is the best order statistic filter; it replaces the value of a pixel by the median of gray levels in the Neighborhood of the pixel.

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

The original of the pixel is included in the computation of the median of the filter are quite possible because for certain types of random noise, the provide excellent noise reduction capabilities with considerably less blurring then smoothing filters of similar size. These are effective for bipolar and unipolar impulse noise.

(e) Max and Min filter:

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

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

It is used for finding the brightest point in an image. Pepper noise in the image has very low values; it is reduced by max filter using the max selection process in the sublimated area sky. 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.

(f) Midpoint filter:

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

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

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

Periodic Noise by Frequency domain filtering:

These types of filters are used for this purpose-

Band Reject Filters:

It removes a band of frequencies about the origin of the Fourier transformer.

Ideal Band reject Filter:

An ideal band reject filter is given by the expression

$$H(u,v)=\begin{cases}1 & \text{if} \quad D(u,v)< D_0-W/2\\ 0 & \text{if} \quad D_0-W/2\le D(u,v)\le D_0+W/2\\ 1 & \text{if} \quad D(u,v)> D_0+W/2\end{cases}$$

D(u,v)- the distance from the origin of the centered frequency rectangle.

W- the width of the band

Do- the radial center of the frequency rectangle.

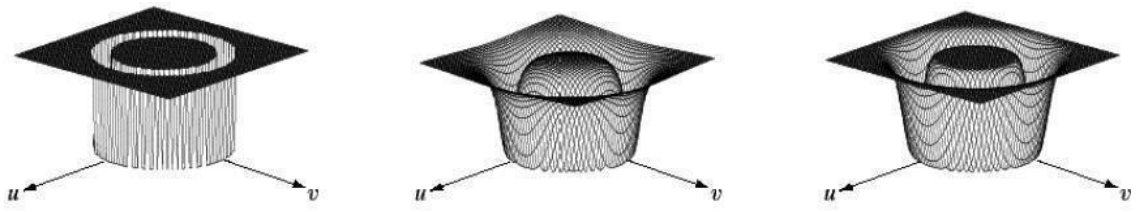
Butterworth Band reject Filter:

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

Gaussian Band reject Filter:

$$H(u,v)=1-\exp\left[-\frac{1}{2}\left(\frac{D^2(u,v)-D_0^2}{D(u,v)W}\right)^2\right]$$

These filters are mostly used when the location of noise component in the frequency domain is known. Sinusoidal noise can be easily removed by using these kinds of filters because it shows two impulses that are mirror images of each other about the origin. Of the frequency transform.



a b c

FIGURE From left to right, perspective plots of ideal, Butterworth (of order 1), and Gaussian bandreject filters.

Band pass Filter:

The function of a band pass filter is opposite to that of a band reject filter It allows a specific frequency band of the image to be passed and blocks the rest of frequencies. The transfer function of a band pass filter can be obtained from a corresponding band reject filter with transfer function Hbr(u,v) by using the equation

$$H_{BP}(u,v) = 1 - H_{BR}(u,v)$$

These filters cannot be applied directly on an image because it may remove too much details of an image but these are effective in isolating the effect of an image of selected frequency bands.

Notch Filters:

A notch filter rejects (or passes) frequencies in predefined neighborhoods about a center frequency.

Due to the symmetry of the Fourier transform notch filters must appear in symmetric pairs about the origin.

The transfer function of an ideal notch reject filter of radius D_0 with centers a (u_0, v_0) and by symmetry at $(-u_0, v_0)$ is

$$D_1(u,v) = \sqrt{(u - M/2 - u_0)^2 + (v - N/2 - v_0)^2}$$

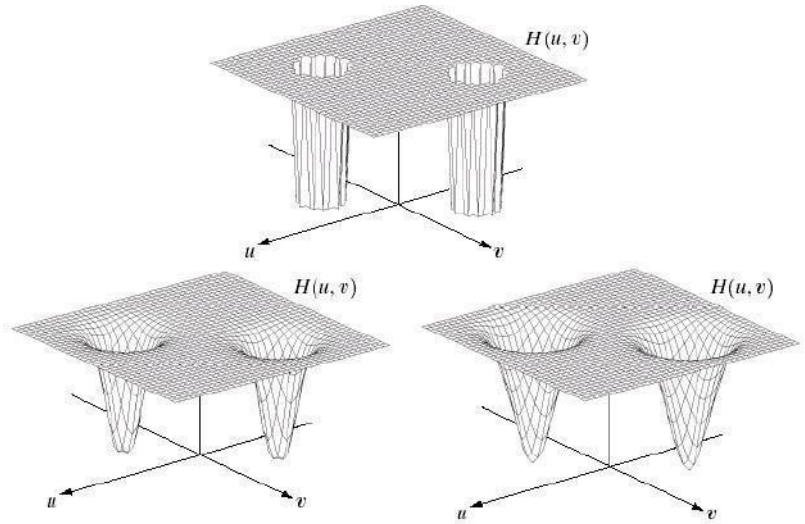
$$D_2(u,v) = \sqrt{(u - M/2 + u_0)^2 + (v - N/2 + v_0)^2}$$

Ideal, butterworth, Gaussian notch filters

$$H(u,v)=\begin{cases}0 & \text{if} \quad D_1(u,v)\le D_0 \text{ or } D_2(u,v)\le D_0\\ 1 & \text{otherwise}\end{cases}$$

$$H(u,v)=1\Big/\left[1+\left(\frac{D_0^2}{D_1(u,v)D_2(u,v)}\right)^n\right]$$

$$H(u,v)=1-\exp\left[-\frac{1}{2}\left(\frac{D_1(u,v)D_2(u,v)}{D_0^2}\right)\right]$$



a
b c

FIGURE Perspective plots of (a) ideal, (b) Butterworth (of order 2), and (c) Gaussian notch (reject) filters.

Inverse Filtering:

The simplest approach to restoration is direct inverse filtering where we complete an estimate $\hat{F}(u,v)$ of the transform of the original image simply by dividing the transform of the degraded image G(u,v) by degradation function H(u,v)

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

We know that

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

Therefore

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

From the above equation we observe that we cannot recover the undegraded image exactly because N(u,v) is a random function whose Fourier transform is not known.

One approach to get around the zero or small-value problem is to limit the filter frequencies to values near the origin.

We know that H(0,0) is equal to the average values of h(x,y).

By Limiting the analysis to frequencies near the origin we reduce the probability of encountering zero values.

Minimum mean Square Error (Wiener) filtering:

The inverse filtering approach has poor performance. The wiener filtering approach uses the degradation function and statistical characteristics of noise into the restoration process.

The objective is to find an estimate $\hat{f}$ of the uncorrupted image f such that the mean square error between them is minimized.

The error measure is given by

$$e^2 = E\{[f(x) - \hat{f}(x)]^2$$

Where E{.} is the expected value of the argument.

We assume that the noise and the image are uncorrelated one or the other has zero mean.

The gray levels in the estimate are a linear function of the levels in the degraded image.

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

$$= \left[\frac{H^*(u,v)}{|H(u,v)|^2 + S_\eta(u,v)/S_f(u,v)}\right]G(u,v)$$

$$= \left[\frac{1}{H(u,v)}\frac{|H(u,v)|^2}{|H(u,v)|^2 + S_\eta(u,v)/S_f(u,v)}\right]G(u,v)$$

Where H(u,v)= degradation function

H*(u,v)=complex conjugate of H(u,v)

$|H(u,v)|^2 = H^*(u,v)\,H(u,v)$

$S_n(u,v) = |N(u,v)|^2$= power spectrum of the noise

$S_f(u,v) = |F(u,v)|^2$= power spectrum of the underrated image

The power spectrum of the under graded image is rarely known. An approach used frequently when these quantities are not known or cannot be estimated then the expression used is

$$\hat{F}(u,v) = \left[\frac{1}{H(u,v)}\frac{|H(u,v)|^2}{|H(u,v)|^2 + K}\right]G(u,v)$$

Where K is a specified constant.

Constrained least squares filtering:

The wiener filter has a disadvantage that we need to know the power spectra of the undegraded image and noise. The constrained least square filtering requires only the knowledge of only the mean and variance of the noise. These parameters usually can be calculated from a given degraded image this is the advantage with this method. This method produces a optimal result. This method require the optimal criteria which is important we express the

$$g(x,y) = h(x,y) \star f(x,y) + \eta(x,y)$$ in vector-matrix form

$$\mathbf{g} = \mathbf{Hf} + \boldsymbol{\eta}$$

The optimality criteria for restoration is based on a measure of smoothness, such as the second derivative of an image (Laplacian).

The minimum of a criterion function C defined as

$$C = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1}[\nabla^2 f(x,y)]^2$$

Subject to the constraint

$$\|\mathbf{g} - \mathbf{H}\hat{\mathbf{f}}\|^2 = \|\boldsymbol{\eta}\|^2$$

Where $\|\mathbf{w}\|^2 \triangleq \mathbf{w}^T\mathbf{w}$ is a euclidean vector norm $\hat{\mathbf{f}}$ is estimate of the undegraded image. ∇^2 is laplacian operator.

The frequency domain solution to this optimization problem is given by

$$\hat{F}(u,v) = \left[\frac{H^*(u,v)}{|H(u,v)|^2 + \gamma|P(u,v)|^2}\right]G(u,v)$$

Where γ is a parameter that must be adjusted so that the constraint is satisfied.

P(u,v) is the Fourier transform of the laplacian operator

$$p(x, y) = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

UNIT-IV

IMAGE SEGMENTATION

Image segmentation: Detection of discontinuities. Edge linking and boundary detection, Thresholding, Region oriented segmentation

Morphological Image Processing: Dilation and Erosion, Dilation, Structuring Element Decomposition, Erosion, Combining Dilation and Erosion, Opening and Closing, the Hit or Miss Transformation.

Image segmentation

Image segmentation is the division of an image into regions or categories, which correspond to different objects or parts of objects. Every pixel in an image is allocated to one of a number of these categories.

A good segmentation is typically one in which:

• pixels in the same category have similar grey scale of multivariate values and forma connected region,

• Neighboring pixels which are in different categories have dissimilar values.

Segmentation is often the critical step in image analysis: the point at which we move from considering each pixel as a unit of observation to working with objects (or parts of objects) in the image, composed of many pixels.

Image segmentation is the key behind image understanding. Image segmentation is considered as an important basic operation for meaningful analysis and interpretation of image acquired.

It is a critical and essential component of an image analysis and/or pattern recognition system, and is one of the most difficult tasks in image processing, which determines the quality of the final segmentation.

If segmentation is done well then all other stages in image analysis are made simpler. But, as we shall see, success is often only partial when automatic segmentation algorithms are used. However, manual intervention can usually overcome these problems, and by this stage the computer should already have done most of the work.

Segmentation algorithms may either be applied to the images as originally recorded, or after the application of transformations and filters considered in chapters 2 and 3. After segmentation, methods of mathematical morphology can be used to improve the results. The

segmentation results will be used to extract quantitative information from the images.

There are ***three general approaches to segmentation***,

- Termed thresholding,
- Edge-based methods and
- Region-based methods.

In ***thresholding***, pixels are allocated to categories according to the range of values in which a pixel lies. Fig 4.1(a) shows boundaries which were obtained by thresholding the muscle fibers image. Pixels with values less than 128 have been placed in one category, and the rest have been placed in the other category. The boundaries between adjacent pixels in different categories has been superimposed in white on the original image. It can be seen that the threshold has successfully segmented the image into the two predominant fiber types.

• In ***edge-based segmentation***, an edge filter is applied to the image, pixels are classified as edge or non-edge depending on the filter output, and pixels which are not separated by an edge are allocated to the same category. Fig 4.1(b) shows the boundaries of connected regions after applying Prewitt's filter (§3.4.2) and eliminating all non-border segments containing fewer than 500 pixels. (More details will be given in§4.2.)

• Finally, ***region-based segmentation algorithms*** operate iteratively by grouping together pixels which are neighbors and have similar values and splitting groups of pixels which are dissimilar in value. Fig 4.1(c) shows the boundaries produced by one such algorithm, based on the concept of watersheds, about which we will give more details in§4.3

Note that none of the three methods illustrated in Fig 4.1 has been completely successful in segmenting the muscle fibers image by placing a boundary between every adjacent pair of fibers. Each method has distinctive faults. For example, in Fig 4.1(a) boundaries are well placed, but others are missing. In Fig 4.1(c), however, more boundaries are present, and they are smooth, but they are not always in exactly the right positions.

The following three sections will consider these three approaches in more detail. Algorithms will be considered which can either be fully automatic or require some manual intervention. The key points of the chapter will be summarized.

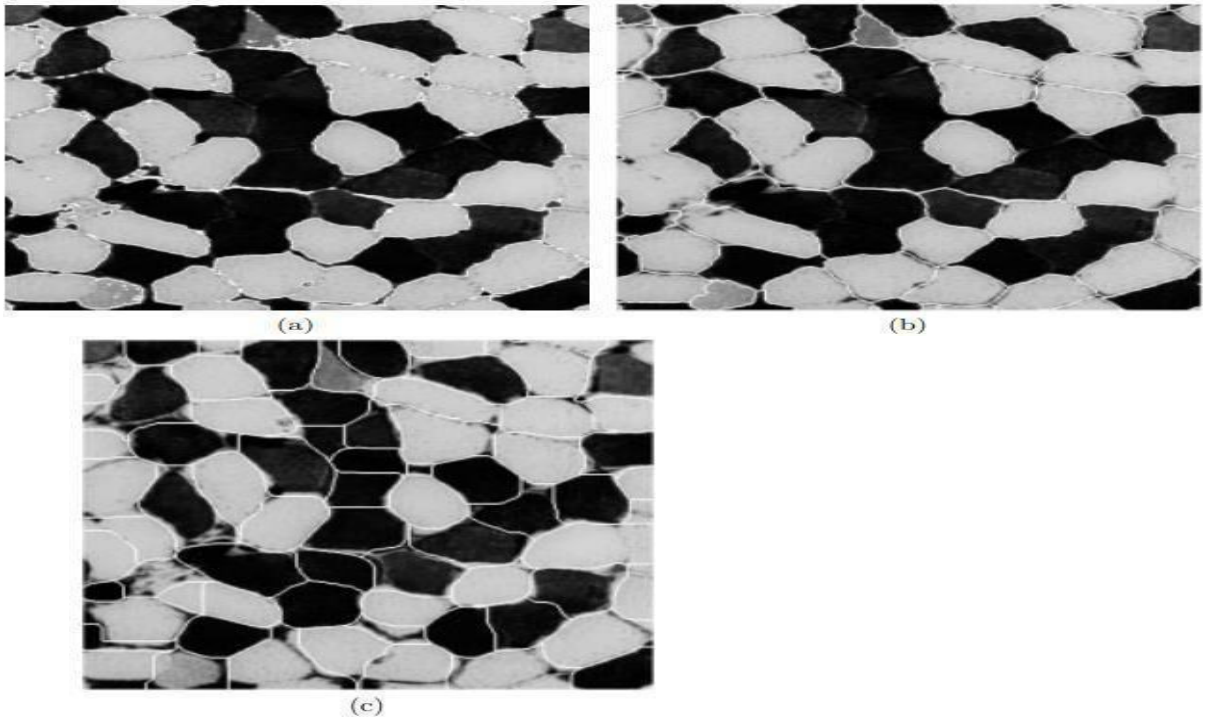


Fig. Boundaries produced by three segmentations of the muscle fibers images: (a) by thresholding, (b) connected regions after thresholding the output of Prewitt"s edge filter and removing small regions, (c) result produced by watershed algorithm on output from a variance filter with Gaussian weights ($\boldsymbol{\sigma}^2 = 96$).

A number of image segmentation techniques are available, but there is no one single technique that is suitable to all the application. Researchers have extensively worked over this fundamental problem and proposed various methods for image segmentation. These methods can be broadly classified into ***seven*** groups:

(1) Histogram thresholding,

(2) Clustering (Fuzzy and Hard),

(3) Region growing, region splitting and merging,

(4) Discontinuity-based,

(5) Physical model-based,

(6) Fuzzy approaches, and

(7) Neural network and GA (Genetic algorithm) based approaches.

Discontinuity based segmentation is one of the widely used techniques for monochrome image segmentation. In discontinuity-based approach, the partitions or subdivision of an image is based on some abrupt changes in the intensity level of images. Here, we mainly interest in identification of isolated points, lined and edges in an image.

The area of edge detection algorithms. The image segmentation based on discontinuity-based approach. Under this approach, we analyses the point detection, line detection and edge detection techniques. A number of operator which are based on first-order derivatives and second-order derivatives such as prewitt, sobel, roberts etc..

THRESHOLDING:

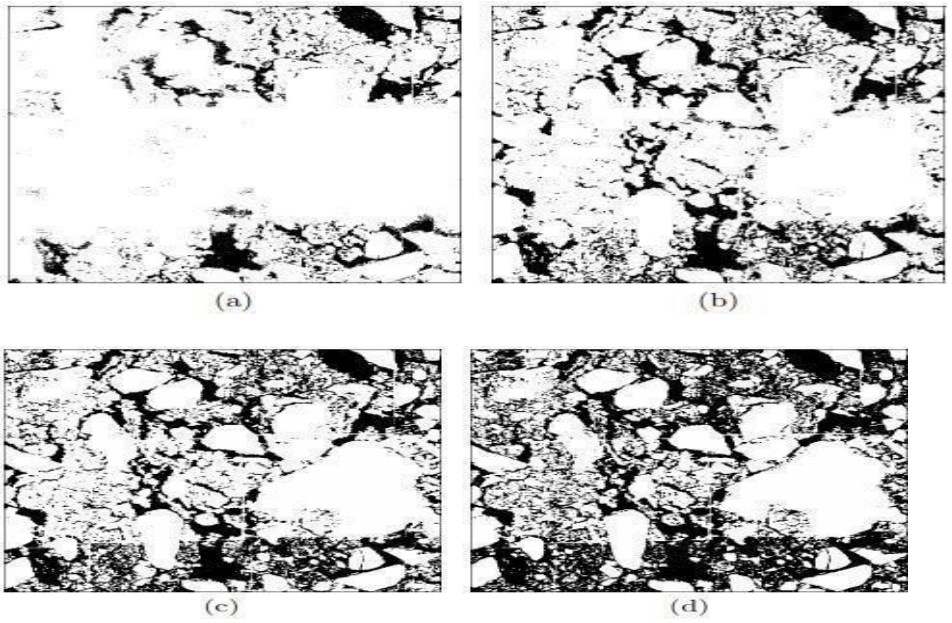
h(x) P p (x) P p(x)=

$$: \frac{P_1}{\sqrt{2\pi\sigma_1^2}} e^{-\frac{1}{2}\left(\frac{x-\mu_1}{\sigma_1}\right)^2} + \frac{P_2}{\sqrt{2\pi\sigma_2^2}} e^{-\frac{1}{2}\left(\frac{x-\mu_2}{\sigma_2}\right)^2}$$

Thresholding is the simplest and most commonly used method of segmentation. Given a single threshold, t, the pixel located at lattice position (i, j), with grayscale value fij , is allocated to category 1 if

$$fij \leq t.$$

In many cases t is chosen manually by the scientist, by trying a range of values of t and seeing which one works best at identifying the objects of interest. Fig 4.2 shows some segmentations of the soil image Thresholds of 7, 10, 13, 20, 29 and 38 were chosen in Figs 4.2(a) to (f) respectively, to identify approximately 10, 20, 30, 40, 50 and 60% of the pixels as being pores. Fig 4.2(d), with a threshold of 20, looks best because most of the connected pore network evident.



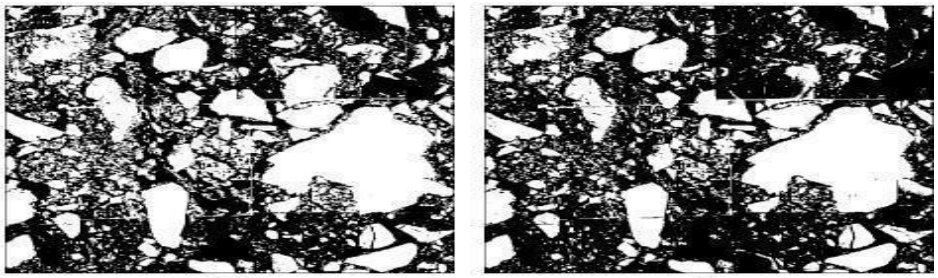


Fig.4.2: six segmentations of the soil image, obtained using manually-selected thresholds of (a)7, (b) 10, (c) 13, (d) 20, (e) 29 and (f) 38. These correspond to approximately 10%, 20%,….60%, respectively, of the image being displayed as black.

Note that:

• Although pixels in a single threshold category will have similar values (either in the range 0 to t, or in the range (t + 1) to 255), they will not usually constitute a single connected component. This is not a problem in the soil image because the object (air) is not necessarily connected, either in the imaging plane or in three-dimensions. In other cases, thresholding would be followed by dividing the initial categories into sub-categories of connected regions.

• More than one threshold can be used, in which case more than two categories are produced.

• Thresholds can be chosen automatically.

In §4.1.1 we will consider algorithms for choosing the threshold on the basis of the histogram of grayscale pixel values. In §4.1.2, manually- and automatically-selected classifiers for multivariate images will be considered. Finally, in §4.1.3, thresholding algorithms which make use of context (that is, values of neighboring pixels as well as the histogram of pixel values) will be presented.

HISTOGRAM-BASED THRESHOLDING

We will denote the histogram of pixel values by h0, h1,...,hN , where hk specifies the number of pixels in an image with grey scale value k and N is the maximum pixel value (typically 255). Ridler and Calvard (1978) and Trussell (1979) proposed a simple algorithm for choosing a single threshold. We shall refer to it as the inter means algorithm. First we will describe the algorithm in words, and then mathematically.

Initially, a guess has to be made at a possible value for the threshold. From this, the mean values of pixels in the two categories produced using this threshold are calculated. The threshold is repositioned to lie exactly half way between the two means. Mean values are calculated again

and a new threshold is obtained, and so on until the threshold stops changing value. Mathematically, the algorithm can be specified as follows.

1. Make an initial guess at t: for example, set it equal to the median pixel value, that is, the value for which

$$\sum_{k=0}^{t} h_k \geq \frac{n^2}{2} > \sum_{k=0}^{t-1} h_k,$$

where n^2 is the number of pixels in the n × n image.

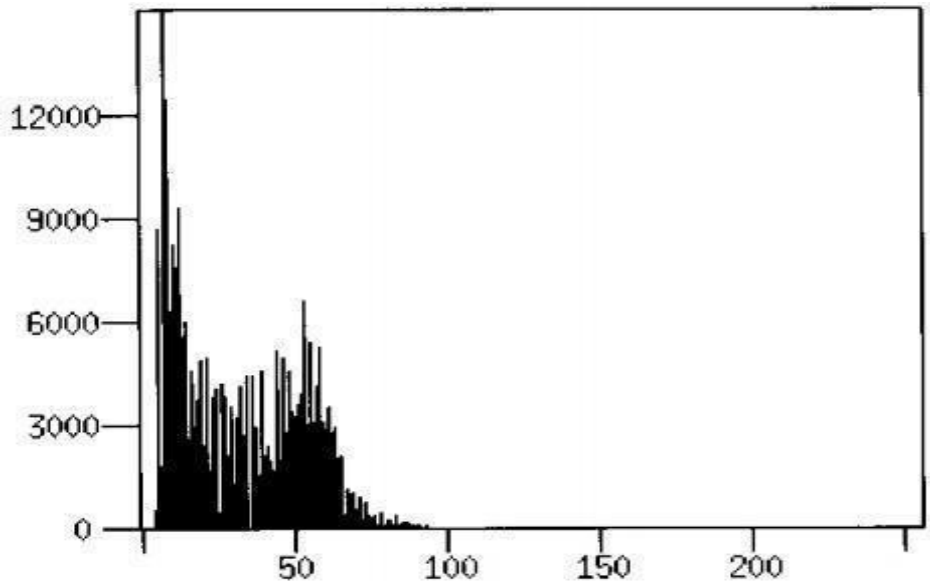


Figure 4.3: Histogram of soil image.

2. Calculate the mean pixel value in each category. For values less than or equal to t, this is givenby:

$$\mu_1 = \sum_{k=0}^{t} k h_k \Big/ \sum_{k=0}^{t} h_k$$

Whereas, for values greater than t, it is given by:

$$\mu_2 = \sum_{k=t+1}^{N} k h_k \Big/ \sum_{k=t+1}^{N} h_k \,.$$

3. Re-estimate t as half-way between the two means ,i.e.

$$t = \left[\frac{\mu_1 + \mu_2}{2}\right]$$

where [ ] denotes 'the integer part of' the expression between the brackets.

4. Repeat steps (2) and (3) until 't' stops changing value between consecutive evaluations.

Fig 4.3 shows the histogram of the soil image. From an initial value of t = 28 (the median pixel value), the algorithm changed t to 31, 32, and 33 on the first three iterations, and then t remained unchanged. The pixel means in the two categories are 15.4 and 52.3. Fig 4.4(a) shows the result of using this threshold. Note that this value of t is considerably higher than the threshold value of 20 which we favored in the manual approach.

The inter means algorithm has a tendency to find a threshold which divides the histogram in two, so that there are approximately equal numbers of pixels in the two categories. In many applications, such as the soil image, this is not appropriate. One way to overcome this drawback is to modify the algorithm as follows.

Consider a distribution which is a mixture of two Gaussian distributions. Therefore, in the absence of sampling variability, the histogram is given by:

$$h_k = n^2\{p_1\phi_1(k) + p_2\phi_2(k)\}, \qquad \text{for } k = 0, 1, \ldots, N.$$

Here, p_1 and p_2 are proportions (such that p1 + p2 = 1) and $\varphi_l(k)$ denotes the probability density of a Gaussian distribution, thatis

$$\phi_l(k) = \frac{1}{\sqrt{2\pi\sigma_l^2}} \exp\left\{\frac{-(k-\mu_l)^2}{2\sigma_l^2}\right\} \qquad \text{for } l = 1, 2,$$

where μ_l and σ^2_l are the mean and variance of pixel values in category l. The best classification criterion, i.e. the one which misclassifies the least number of pixels, allocates pixels with value k to category 1 if p

$$p_1\phi_1(k) \geq p_2\phi_2(k),$$

and otherwise classifies them as 2. After substituting for φ and taking logs, the inequality becomes

$$k^2\left\{\frac{1}{\sigma_1^2} - \frac{1}{\sigma_2^2}\right\} - 2k\left\{\frac{\mu_1}{\sigma_1^2} - \frac{\mu_2}{\sigma_2^2}\right\} + \left\{\frac{\mu_1^2}{\sigma_1^2} - \frac{\mu_2^2}{\sigma_2^2} + \log\frac{\sigma_1^2 p_2^2}{\sigma_2^2 p_1^2}\right\} \leq 0.$$

The left side of the inequality is a quadratic function in k. Let A, B and C denote the three terms in curly brackets, respectively. Then the criterion for allocating pixels with value k to category 1 is:

$$k^2A - 2kB + C \leq 0.$$

There are three cases to consider:

(a) If A = 0 (i.e. $\sigma_1^2 = \sigma_2^2$), the criterion simplifies to one of allocating pixels with value k to category 1 if

$$2kB \geq C.$$

(If, in addition, $p_1 = p_2$ and $\mu_1 < \mu_2$, the criterion becomes $k \leq 1/2\ \{\mu_1 + \mu_2\}$. Note that this is the intermeans criterion, which implicitly assumes that the two categories are of equal size.)

(b) If B < AC, then the quadratic function has no real roots, and allpixels are classified as 1 if A < 0 (i.e. $\sigma_1^2 > \sigma_2^2$), or as 2 if A > 0

(c) Otherwise, denote the roots t_1 and t_2, where $t_1 \leq t_2$and

$$t_1, t_2 = \frac{B \pm \sqrt{B^2 - AC}}{A}.$$

The criteria for category 1 are

$$t_1 < k \leq t_2 \qquad \text{if } A > 0,$$

$$k \leq t_1 \text{ or } k > t_2 \qquad \text{if } A < 0.$$

In practice, cases (a) and (b) occur infrequently, and if $\mu_1 < \mu_2$ the rule simplifies to thethreshold:

$$\text{category 1 if a pixel value} \quad k \leq \frac{B + \sqrt{B^2 - AC}}{A}.$$

Kittler and Illingworth (1986) proposed an iterative minimum-error algorithm, which is based on this threshold and can be regarded as a generalization of the inter means algorithm. Again, we will describe the algorithm in words, and then mathematically.

From an initial guess at the threshold, the proportions, means and variances of pixel values in the two categories are calculated. The threshold is repositioned according to the above criterion, and proportions, means and variances are recalculated. These steps are repeated until there are no changes in values between iterations.

Mathematically:

1. Make an initial guess at a value fort.
2. Estimate p_1, μ_1 and σ_1^2 for pixels with values less than or equal to t,by

$$p_1 = \frac{1}{n^2} \sum_{k=0}^{t} h_k,$$

$$\mu_1 = \frac{1}{n^2 p_1} \sum_{k=0}^{t} k h_k,$$

$$\text{and } \sigma_1^2 = \frac{1}{n^2 p_1} \sum_{k=0}^{t} k^2 h_k \quad - \quad \mu_1^2.$$

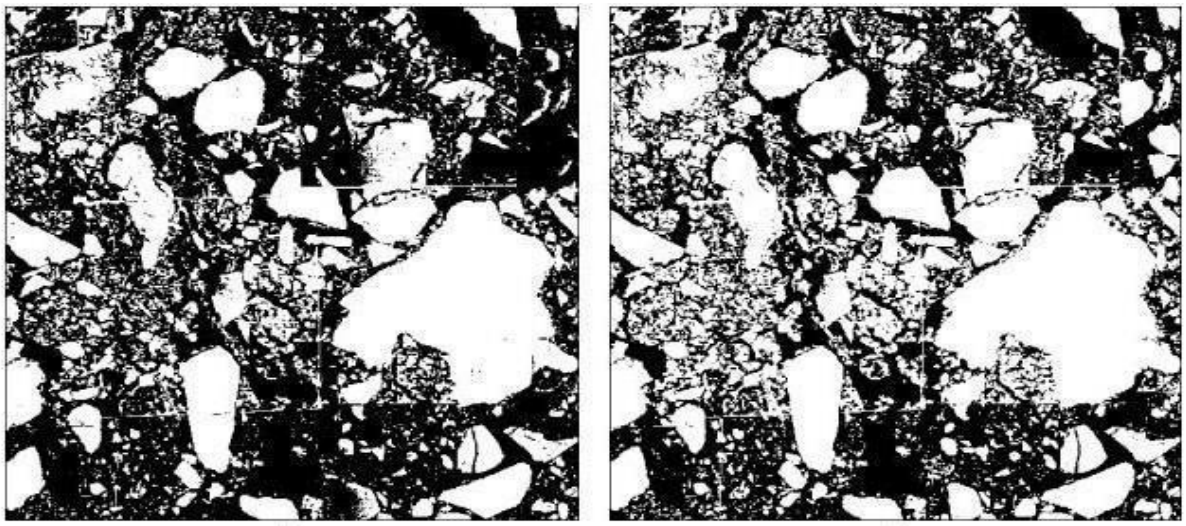
Similarly, estimate p_2, μ_2 and σ_2^2 for pixels in the range t + 1 to N.

3. Re-estimate tby

$$t = \left[\frac{B + \sqrt{B^2 - AC}}{A} \right],$$

where A, B, C and [ ] have already been defined.

4. Repeat steps (2) and (3) until *t* converges to a stable value.



(a) (b)

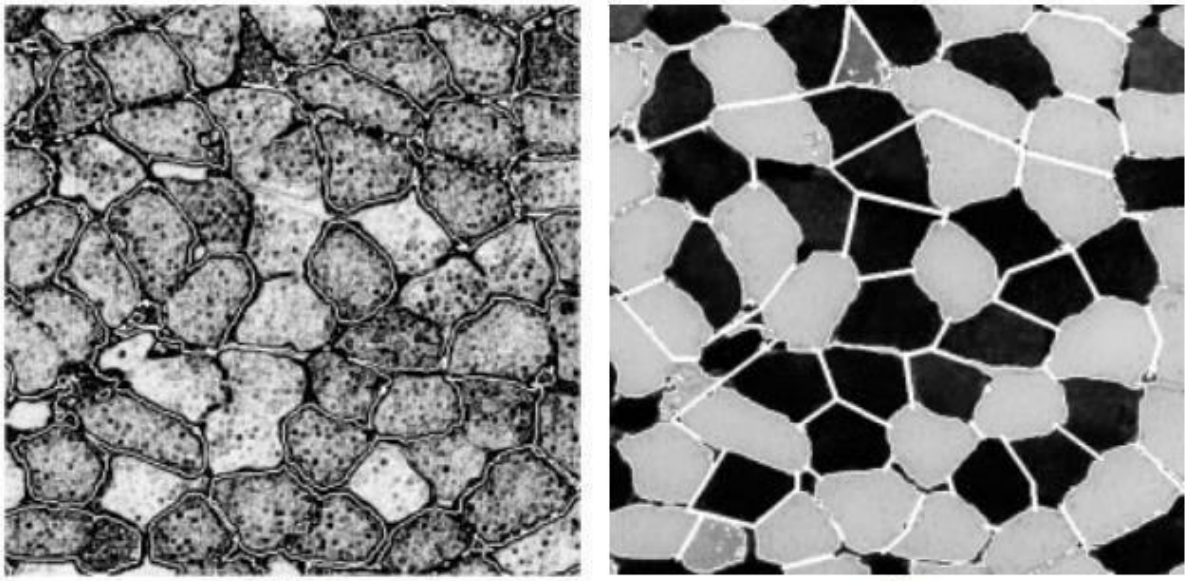
Figure 4.4: Segmentations of the soil image, obtained by thresholding at: **(a)** 33, selected by the intermeans algorithm, **(b)** 24, selected by the minimum-error algorithm.

When applied to the soil image, the algorithm converged in 4 iterations to t = 24. Fig 4.4(b) shows the result, which is more satisfactory than that produced by the inter means

Algorithm because it has allowed for a smaller proportion of air pixels ($p_1 = 0.45$, as compared with $p_2 = 0.55$). The algorithm has also taken account of the air pixels being less variable in value than those for the soil matrix ($\sigma_1^2 = 30$, whereas $\sigma_2^2 = 186$). This is in accord with the left- most peak in the histogram plot (Fig 4.3) being quite narrow.

EDGE-BASED SEGMENTATION

As we have seen, the results of threshold-based segmentation are usually less than perfect. Often, a scientist will have to make changes to the results of automatic segmentation. One simple way of doing this is by using a computer mouse to control a screen cursor and draw boundary lines between regions. Fig 4.10(a) shows the boundaries obtained by thresholding the muscle fibres image (as already displayed in Fig 4.1(a)), superimposed on the output from Prewitt's edge filter (§3.4.2), with the contrast stretched so that values between 0 and 5 are displayed as shades of grey ranging from white to black and values exceeding 5 are all displayed as black. This display can be used as an aid to determine where extra boundaries need to be inserted to fully segment all muscle fibers. Fig 4.10(b) shows the result after manually adding 71 straight lines.



(a) (b)

Figure 4.10: **(a)** Boundaries obtained by thresholding the muscle fibres image, superimposed on output for Prewitt's filter, with values between 0 and 5 displayed in progressively darker shades of grey and values in excess of 5 displayed as black. **(b)** Manual segmentation of the image by addition of extra lines to boundaries obtained by thresholding, superimposed in white on the original image.

Algorithms are available for semi-automatically drawing edges, whereby the scientist's rough lines are smoothed and perturbed to maximize some criterion of match with the image (see, for example, Samadani and Han, 1993). Alternatively, edge finding can be madefully

automatic, although not necessarily fully successful. Fig 4.11(a) shows the result of applying Prewitt's edge filter to the muscle fiber image. In this display, the filter output has been threshold at a value of 5: all pixels exceeding 5 are labeled as edge pixels and displayed as black. Connected chains of edge pixels divide the image into regions. Segmentation can be achieved by allocating to a single category all non-edge pixels which are not separated by an edge. Rosenfeld and Pfaltz (1966) gave an efficient algorithm for doing this for 4- and 8- connected regions, termed a connected components algorithm. We will describe this algorithm in words, and then mathematically.

The algorithm operates on a raster scan, in which each pixel is visited in turn, starting at the top-left corner of the image and scanning along each row, finishing at the bottom-right corner. For each non-edge pixel, (i, j), the following conditions are checked. If its already visited neighbors — (i − 1, j) and (i, j − 1) in the 4-connected case, also (i − 1, j − 1) and (i − 1, j + 1) in the 8-connected case — are all edge pixels, then a new category is created and (i, j) is allocated to it. Alternatively, if all its non-edge neighbors are in a single category, then (i, j) is also placed in that category. The final possibility is that neighbors belong to two or more categories, in which case (i, j) is allocated to one of them and a note is kept that these categories are connected and therefore should from then on be considered as a single category. More formally, for the simpler case of 4-connectedregions:

• Initialize the count of the number of categories by setting K =0.

• Consider each pixel (i, j) in turn in a raster scan, proceeding row by row (i = 1,...,n), and for each value of i taking j =1,...,n.

• One of four possibilities apply to pixel (i, j):

1. If (i, j) is an edge pixel then nothing needs to bedone.
2. If both previously-visited neighbors, (i − 1, j) and (i, j − 1), are edge pixels, then a new category has to be created for (i,j):

$$K \to K+1, \quad h_K = K, \quad g_{ij} = K,$$

where the entries in h1,...,hK are used to keep track of which categories are equivalent, and gij records the category label for pixel (i, j).

3. If just one of the two neighbors is an edge pixel, then (i, j) is assigned the same label as the other one:

$$g_{ij} = \begin{cases} g_{i-1,j} & \text{if } (i, j-1) \text{ is the edge pixel,} \\ g_{i,j-1} & \text{otherwise.} \end{cases}$$

4. The final possibility is that neither neighbor is an edge pixel, in which case (i, j) is given the same label as one of them:

$$g_{ij} = g_{i-1,j},$$

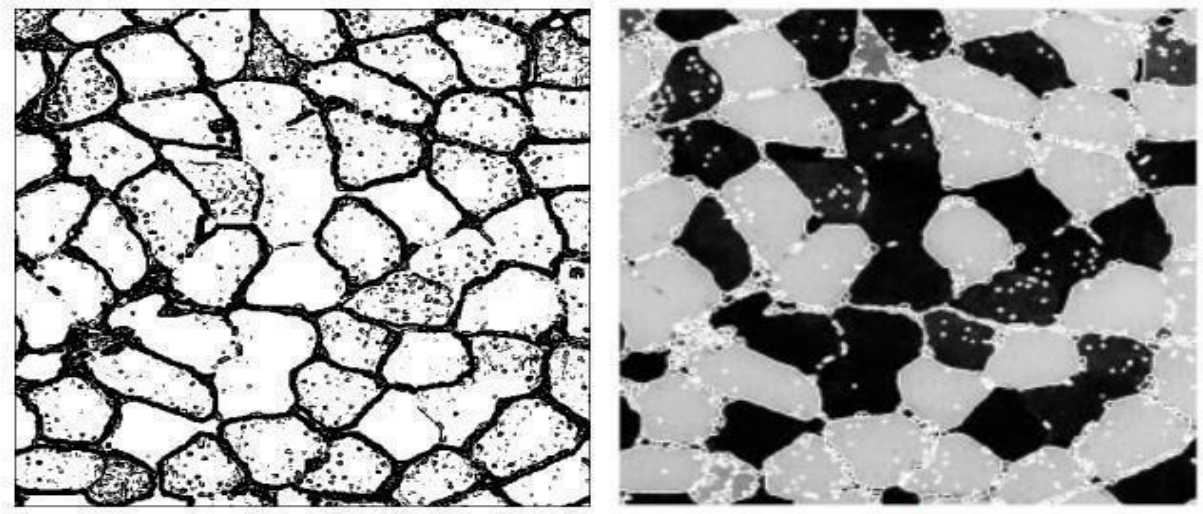
And if the neighbors have labels which have not been marked as equivalent, i.e. hgi−1,j 6= hgi,j−1 , then this needs to be done (because they are connected at pixel (i, j)). The equivalence is recorded by changing the entries in $h_1,...,h_K$, as follows: – Set l1 = min(hgi−1,j , hgi,j−1) and l2 = max(hgi−1,j , hgi,j−1). – For each value of k from 1 to K, if hk = l2 then hk → l1.

• Finally, after all the pixels have been considered, the array of labels is revised, taking into account which categories have been marked for amalgamation:

gij → hgij for i, j = 1, . . . , n

After application of the labeling algorithm, superfluous edge pixels — that is, those which do not separate classes — can be removed: any edge-pixel which has neighbors only of one category is assigned to that category. Fig 4.11(b) shows the result of applying the labeling algorithm with edges as shown in Fig 4.11(a), and removing superfluous edge pixels. The white boundaries have been superimposed on the original image.

Similarly, small segments (say less than 500 pixels in size) which do not touch the borders of the image can be removed, leading to the previously displayed Fig 4.1(b). The segmentation has done better than simple thresholding, but has failed to separate all fibers because of gaps in output from Prewitt's edge filter. Martello (1976), among others, has proposed algorithms for bridging these gaps.



(a) (b)

Figure 4.11: **(a)** Thresholded output from Prewitt's edge filter applied to muscle fibres image: values greater than 5 are displayed as black, those less than or equal to 5 as white. **(b)** Boundaries produced from connected regions in (a), superimposed in white on the original image.

REGION-BASED SEGMENTATION

Segmentation may be regarded as spatial clustering:

- clustering in the sense that pixels with similar values are grouped together, and
- spatial in that pixels in the same category also form a single connected component.

Clustering algorithms may be agglomerative, divisive or iterative (see, for example, Gordon, 1981). Region-based methods can be similarly categorized into:

- those which merge pixels,
- those which split the image into regions, and
- those which both split-and-merge in an iterative search scheme

The distinction between edge-based and region-based methods is a little arbitrary. For example, in §4.2 one of the algorithms we considered involved placing all neighboring non-edge pixels in the same category. In essence, this is a merging algorithm.

Seeded region growing is a semi-automatic method of the merge type. We will explain it by way of an example. Fig 4.13(a) shows a set of seeds, white discs of radius 3, which have been placed inside all the muscle fibres, using an on-screen cursor controlled by a computer mouse. Fig 4.13(b) shows again the output from Prewitt's edge filter. Superimposed on it in white are the seeds and the boundaries of segmentation produced by a form of watershed algorithm. The boundaries are also shown superimposed on the original muscle fibers image in Fig 4.13(c). The watershed algorithm operates as follows (we will explain the namelater).

For each of a sequence of increasing values of a threshold, all pixels with edge strength less than this threshold which form a connected region with one of the seeds are allocated to the corresponding fiber. When a threshold is reached for which two seeds become connected, the pixels are used to label the boundary. A mathematical representation of the algorithm is too complicated to be given here. Instead, we refer the reader to Vincent and Soille (1991) for more details and an efficient algorithm. Meyer and Beucher (1990) also consider the watershed algorithm, and added some refinements to the method.

Note that:

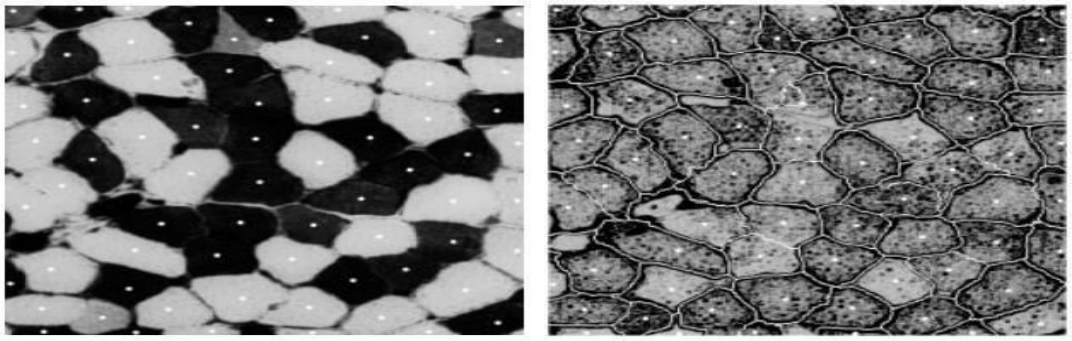
- The use of discs of radius 3 pixels, rather than single points, as seeds make the watershed

results less sensitive to fluctuations in Prewitt's filter output in the middle of fibres.

• The results produced by this semi-automatic segmentation algorithm are almost as good as those shown in Fig 4.10(b), but the effort required in positioning seeds inside muscle fibers is far less than that required to draw boundaries.

• Adams and Bischof (1994) present a similar seeded region growing algorithm, but based directly on the image grey scale, not on the output from an edge filter.

The watershed algorithm, in its standard use, is fully automatic. Again, we will demonstrate this by illustration. Fig 4.14 shows the output produced by a variance filter (§3.4.1) with Gaussian weights ($\sigma^2 = 96$) applied to the muscle fibers image after histogram-equalization (as shown in Fig 2.7(d)). The white seeds overlie all the local minima of the filter output, that is, pixels whose neighbors all have larger values and so are shaded lighter. Note that it is necessary to use a large value of σ^2 to ensure that the filter output does not have many more local minima. The boundaries produced by the watershed algorithm have been added to Fig 4.14. An intuitive way of viewing the watershed algorithm is by considering the output from the variance filter as an elevation map: light areas are high ridges and dark areas are valleys. Each local minimum can be thought of as the point to which any water falling on the region drains, and the segments are the catchments for them. Hence, the boundaries, or watersheds, lie along tops of ridges. The previously mentioned Fig 4.1(c) shows this segmentation superimposed on the original image.



(a) (b)

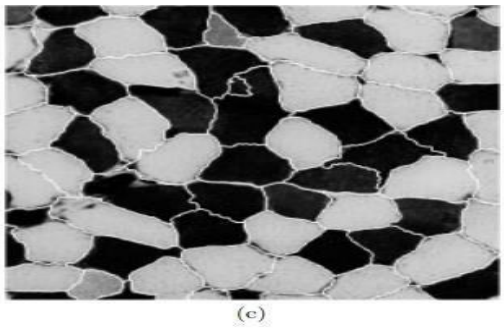


Fig.: Manual segmentation of muscle fibers image by use of watersheds algorithm (a) manually positioned 'seeds' in centers of all fibers, (b) output from Prewitt's edge filter together with watershed boundaries, (c) watershed boundaries superimposed on the image.

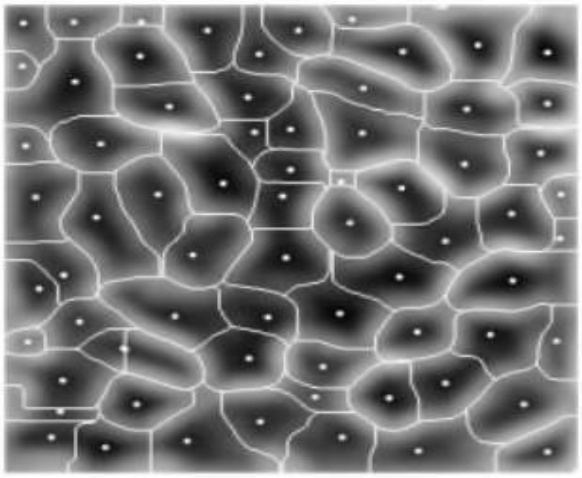


Figure: Output of variance filter with Gaussian weights ($\sigma 2 = 96$) applied to muscle fibers image, together with seeds indicating all local minima and boundaries produced by watershed algorithm.

There are very many other region-based algorithms, but most of them are quite complicated. In this section we will consider just one more, namely an elegant split-and- merge algorithm proposed by Horowitz and Pavlidis (1976). We will present it in a slightly modified form to segment the log-transformed SAR image (Fig 2.6), basing our segmentation decisions on variances, whereas Horowitz and Pavlidis based theirs on the range of pixel values. The algorithm operates in two stages, and requires a limit to be specified for the maximum variance in pixel values in a region.

The first stage is the splitting one. Initially, the variance of the whole image is calculated. If this variance exceeds the specified limit, then the image is subdivided into four quadrants. Similarly, if the variance in any of these four quadrants exceeds the limit it is further subdivided into four. This continues until the whole image consists of a set of squares of varying sizes, all of which have variances below

the limit. (Note that the algorithm must be capable of achieving this because at the finest resolution of each square consisting of a single pixel the variances are taken to be zero.)

Fig 4.15(a) shows the resulting boundaries in white, superimposed on the log- transformed SAR image, with the variance limit set at 0.60. Note that:

• Squares are smaller in non-uniform parts of theimage.

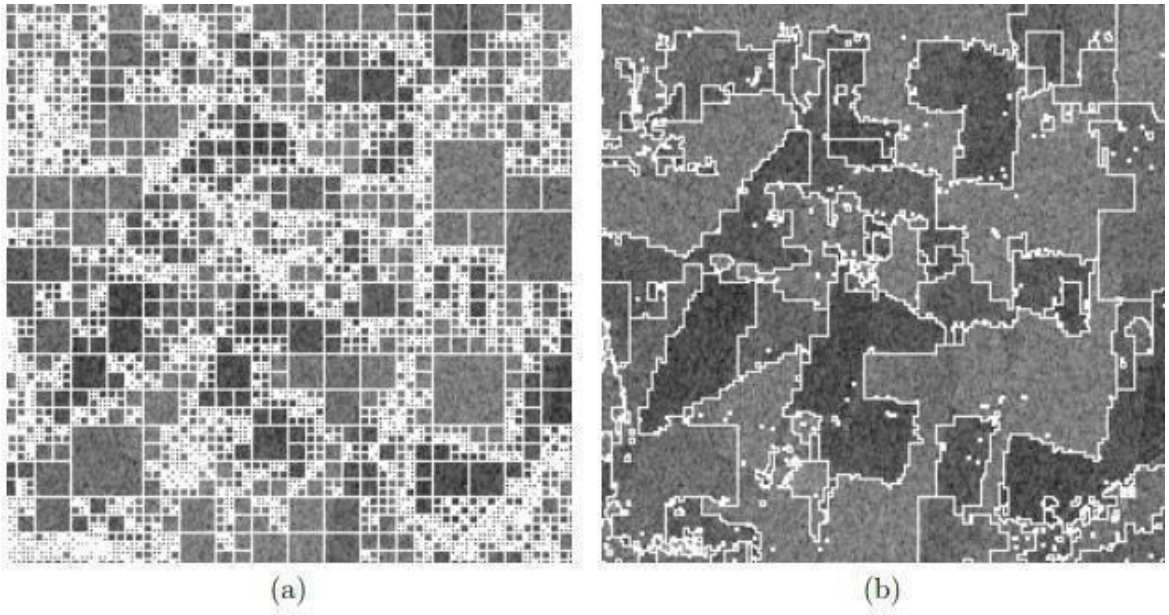


Figure 4.15: Region-growing segmentation of log-transformed SAR image: (a) division of image into squares with variance less than 0.60, obtained as first step in algorithm, (b) final segmentation, after amalgamation of squares, subject to variance limit of 0.60.

• The variance limit was set to 0.60, rather than to the speckle variance value of 0.41 (Horgan, 1994), because in the latter case the resulting regions were verysmall.

• The algorithm requires the image dimension, n, to be a power of 2. Therefore, the 250 × 250 SAR image was filled out to 256 × 256 by adding borders of width3.

The second stage of the algorithm, the merging one, involves amalgamating squares which have a common edge, provided that by so doing the variance of the new region does not exceed the limit. Once all amalgamations have been completed, the result is a segmentation in which every region has a variance less than the set limit. However, although the result of the first stage in the algorithm is unique, that from the second is not — it depends on the order of which squares are considered.

Fig 4.15(b) shows the boundaries produced by the algorithm, superimposed on the SAR

image. Dark and light fields appear to have been successfully distinguished between, although the boundaries are rough and retain some of the artifacts of the squares in Fig 4.15(a).

Pavlidis and Liow (1990) proposed overcoming the deficiencies in the boundaries produced by the Horowitz and Pavlidis algorithm by combining the results with those from an edge-based segmentation. Many other ideas for region-based segmentation have been proposed (see, for example, the review of Haralick and Shapiro, 1985), and it is still an active area of research.

One possibility for improving segmentation results is to use an algorithm which over-segments an image, and then apply a rule for amalgamating these regions. This requires 'high-level' knowledge, which falls into the domain of artificial intelligence. (All that we have considered in this chapter may be termed 'low-level'.) For applications of these ideas in the area of remote sensing, see Tailor, Cross, Hogg and Mason (1986) and Ton, Sticklen and Jain (1991). It is possible that such domain-specific knowledge could be used to improve the automatic segmentations of the SAR and muscle fibres images, for example by constraining boundaries to be straight in the SAR image and by looking only for convex regions of specified size for the muscle fibers.

We briefly mention some other, more-complex techniques which can be used to segment images.

• The Hough transform (see, for example, Leavers, 1992) is a powerful technique for finding straight lines, and other parameterized shapes, in images.

• Boundaries can be constrained to be smooth by employing roughness penalties such as bending energies. The approach of varying a boundary until some such criterion is optimized is known as the fitting of snakes (Kass, Witkin and Terzopoulos1988).

• Models of expected shapes can be represented as templates and matched to images. Either the templates can be rigid and the mapping can be flexible (for example, the thinplate spline of Bookstein, 1989), or the template itself can be flexible, as in the approach of Amit, Grenander and Piccioni (1991).

• Images can be broken down into fundamental shapes, in a way analogous to the decomposition of a sentence into individual words, using syntactic methods (Fu, 1974).

Basic Formulation

Let R represent the entire image region. We want to partition R into n sub regions, R_1 , R_2 , . . ., Rn , such that:

(a) $\bigcup_{i=1}^{n} Ri=R$

(b) Ri is a connected region for i=1, 2, . . , n

(C) Ri ∩ Rj = ϕfor all i and j, i =j

(d) P(Ri) = TRUE for i=1,2, . ..n

(e) P(Ri ∪Rj) = FALSE for i ≠j

Where P(Ri) is a logic predicate over the points in set Ri and ϕis the null set.

The symbols ∪ and ∩ represent set union and intersection, respectively.

The two regions Ri and Rj are said to be adjacent if their union forms a connected set.

Condition (a) indicates that the segmentation must be complete; that is every pixel must be in a region.

Condition (b) requires that points in a region be connected in some predefined sense.

Condition (c) indicates that the regions must be disjoint.

Condition (d) deals with the properties that must be satisfied by the pixels in a segmented region. For ex: P(Ri) = TRUE if all pixels in Ri have the same intensity level. Finally, condition (e) indicates that two adjacent regions Ri and Rj must be different in the sense of predicate P.

Point Detection

A point is the most basic type of discontinuity in a digital image. The most common approach to finding discontinuities is to run an (n n) mask over each point in the image. The mask is as shown in figure 2.

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

Figure 2. A mask for point detection

The point is detected at a location (x, y) in an image where the mask is centered. If the corresponding value of R such that

Where R is the response of the mask at any point in the image and T is non-negative threshold value. It means that isolated point is detected at the corresponding value (x, y). This formulation serves to measures the weighted differences between the center point and its neighbors since the gray level of an isolated point will be very different from that of its neighbors [ ]. The result of point detection mask is as shown in figure 3

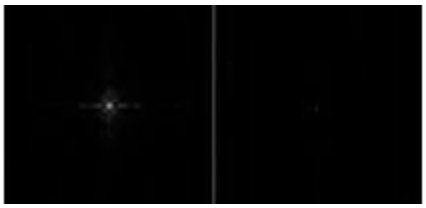


Figure 3. (a) Gray-scale image with a nearly invisible isolated black point (b) Image showing the detected point

Line Detection

Line detection is the next level of complexity in the direction of image discontinuity. For any point in the image, a response can be calculated that will show which direction the point of a line is most associated with. For line detection, we use two masks, and, mask. Then, we have

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

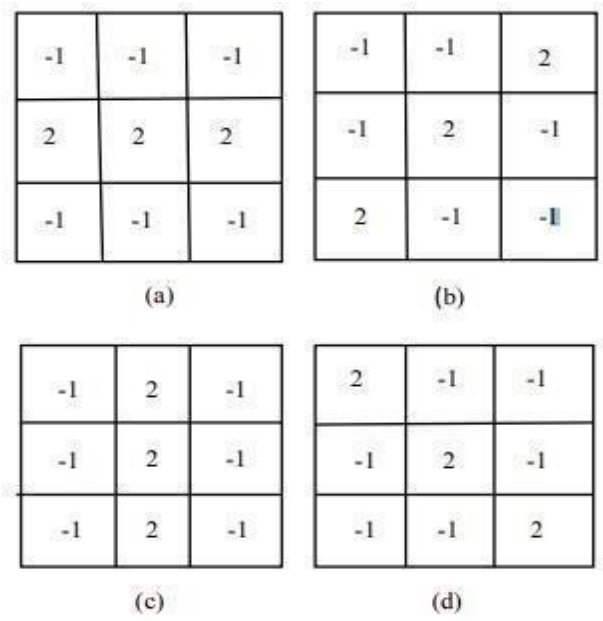


Figure 4. Line Detector masks in (a) Horizontal direction (b) 45° direction (c) Vertical direction (d) - 45° direction The greatest response calculation from these matrices will yield the direction of the given pixel []. The result of line detection mask is as shown in figure 5

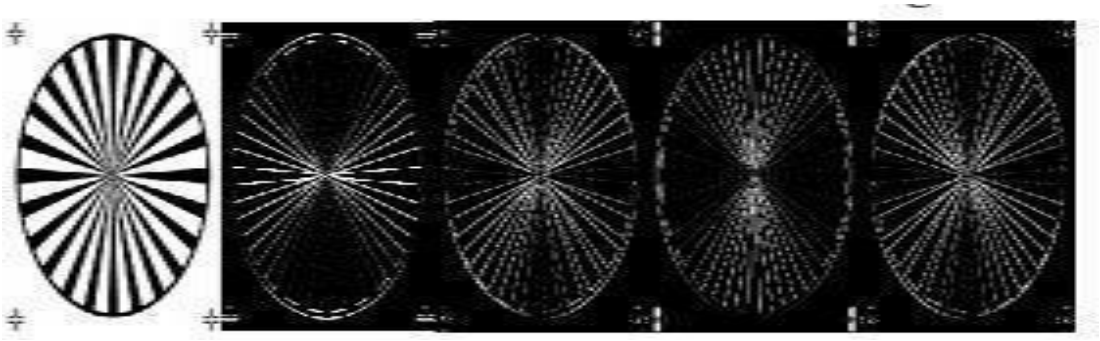


Figure 5. (a) Original Image (b) result showing with horizontal detector (c) with 45° detector (d) with vertical detector (e) with -45° detector

With the help of lines detector masks, we can detect the lines in a specified direction. For example, we are interesting in finding all the lines that are one pixel thick, oriented at -45°. For that, we take a digitized (binary portion of a wire-bond mask for an electronics circuit. The results are shown as in figure6.

Edge detection

Since isolated points and lines of unitary pixel thickness are infrequent in most practical application, edge detection is the most common approach in gray level discontinuity segmentation. An edge is a boundary between two regions having distinct intensity level. It is very useful in detecting of discontinuity in an image. When the image changes from dark to white or vice-versa. The changes of intensity, first-order derivative and second-order derivative are shown in figure7.

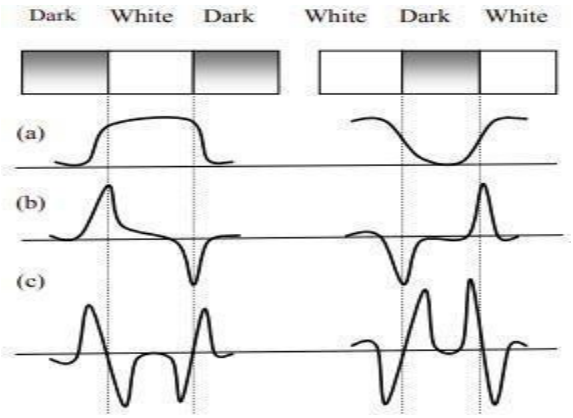


Figure 7. (a) Intensity profile (b) First-order derivatives (c) Second-order derivatives

First-order derivatives. First-order derivatives responds whenever there is discontinuity in intensity level. It is positive at the leading edge and negative at the trailing edge. Suppose we have an image *f(x, y)* and gradient operator *f.*

$$f = \begin{bmatrix} Gx \\ Gy \end{bmatrix} = \begin{bmatrix} \partial f/\partial x \end{bmatrix} \quad \text{----------(4)}$$ strength of ∇f isgivenby

$$\begin{bmatrix} \partial f / \partial y \end{bmatrix}$$

∇f = magnitude of (∇f)

$$= \sqrt{Gx2+Gy2}$$

$\cong \| Gx \| + \|Gy\|$ ---------- (5) It gives the strength of edge at location(x,y)

Direction of ∇f is given by

$$\alpha(x,y) = \tan^{-1}\left(\frac{|Gx|}{Gy}\right) \quad (6)$$

Where $\alpha(x,y)$ gives the direction of ∇f.

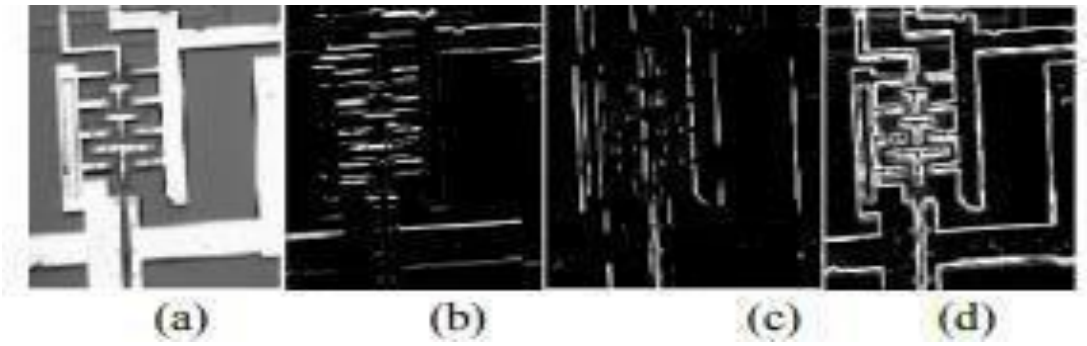


Figure 8. (a) Original Image (b) $\|Gx\|$component of the gradient along x-direction (c) $\|$ Gy $\|$component of the gradient along y-direction (d) Gradient Image $\|Gx\|+ \|Gy\|$

There is several ways to calculate the image gradient:

Prewitt Edge operator

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

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

Figure 9. Masks used for Prewitt Edge operator

The mask finds the horizontal edges is equivalent to gradient in the vertical direction and the mask compute the vertical edges is equivalent to gradient in the horizontal direction. Using these two masks passing to the intensity image, we can find out and component at different location in an image. So, we can find out the strength and direction of edge at that particular 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 10. Masks used for Sobel Edge operator

It gives the averaging effect over an image. It considers the effect due to the spurious noise in the image. It is preferable over prewitt edge operator because it gives the smoothing effect and by which we can reduce spurious edge which are generated because of noise present 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. There are several second-order derivative operators: 3.3.2.1. Laplacian operator. The Laplacian mask

Laplacian operator. The Laplacian mask is given by:

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

Figure 11. Masks used for Laplacian operator

$$\nabla^2(f) = \frac{\partial 2\, f}{\partial \chi 2} + \frac{\partial 2 f}{\partial y 2} \qquad \text{----------} (7)$$

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 12. Masks used for Laplacian operator using 8-connectivity

It is not used for edge detection because it is very sensitive to noise and also leads to double edge. But, it is very useful for extracting secondary information. To reduce the effect of noise, first image will be smooth using the Gaussian operator and then it is operated by Laplacian operator. These two operations together is called LoG (Laplacian of Gaussian) operator.

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 13. Masks used for LoG operator

The Gaussian operator is given by:

$$h(x,y) = \exp\left(-\frac{x^2+y^2}{2\sigma^2}\right) \quad (8)$$

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

$$\nabla^2 h = \left(\frac{r^2-\sigma^2}{\sigma^4}\right)\exp\left(-\frac{r^2}{2\sigma^2}\right) \quad (9)$$

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.

SUMMARY

The key points about image segmentation are

• Segmentation is the allocation of every pixel in an image to one of a number of categories, which correspond to objects or parts of objects. Commonly, pixels in a single category should:

- have similar pixel values,
- form a connected region in the image,
- Be dissimilar to neighboring pixels in other categories.

• Segmentation algorithms may either be applied to the original images or after the application of transformations and filters (considered in chapters 2 and3).

• Three general approaches to segmentation are:

– thresholding,

– edge-based methods,

– Region-based methods.

• Methods within each approach may be further divided into those which:

– require manual intervention, or

– Are fully automatic.

A single threshold, t, operates by allocating pixel (i, j) to category 1 if fij ≤ t, and otherwise putting it in category 2. Thresholds may be obtained by:

- manual choice, or
- applying an algorithm such as inter means or minimum-error to the histogram of pixel values.

– Inter means positions t half-way between the means in the two categories.
– Minimum-error chooses t to minimize the total number of misclassifications on the assumption that pixel values in each category are Normally distributed.

- Thresholding methods may also be applied to multivariate images. In this case, two possibilities are:
  – manually selecting a training set of pixels which are representative of the different categories, and then using linear discrimination,

– k-means clustering, in which the categories are selected automatically from the data.

• The context of a pixel, that is the values of neighboring pixels, may also be used to modify the threshold value in the classification process. We considered three methods:

– restricting the histogram to those pixels which have similarly valued neighbours,

– post-classification smoothing,

- using Bayesian image restoration methods, such as the iterated conditional modes (ICM)algorithm.

In edge-based segmentation, all pixels are initially labeled as either being on an edge or not, then non-edge pixels which form connected regions are allocated to the same category. Edge labeling may be:

• manual, by using a computer mouse to control a screen cursor and draw boundary lines between regions,

• Automatic, by using an edge-detection filter. Edges can be located either:

– where output from a filter such as Prewitt's exceeds a threshold,or

– at zero crossings from a Laplacian-of-Gaussianfilter.

Region-based algorithms act by grouping neighboring pixels which have similar values and splitting groups of pixels which are heterogeneous in value. Three methods were considered:

• Regions may be grown from manually-positioned 'seed' points, for example, by applying a watershed algorithm to output from Prewitt'sfilter.

• The watershed algorithm may also be run fully automatically, for example, by using local minima from a variance filter as seed points.

• One split-and-merge algorithm finds a partition of an image such that the variance in pixel values within every segment is below a specified threshold, but no two adjacent segments can be amalgamated without violating thethreshold.

Finally, the results from automatic segmentation can be improved by:

• Using methods of mathematical morphology (chapter 4 of secondpart).

• Using domain-specific knowledge, which is beyond the scope of this syllabus.

MORPHOLOGICAL IMAGE PROCESSING

Till now discussed segmentation based on three principal concepts

(a) edge detection

(b) thresholding, and

(c) region growing.

Each of these approaches was found to have advantages and disadvantages. In this section we discuss an approach based on the concept of so-called morphological watersheds. These often produce more stable segmentation results, including connected segmentation boundaries. This approach also provides a simple framework for incorporating knowledge-based constraints in the segmentation process.

Background:

The concept of watersheds is based on visualizing an image in three dimensions: two spatial coordinates versus intensity. In such a "topographic" interpretation, we consider three types of points:

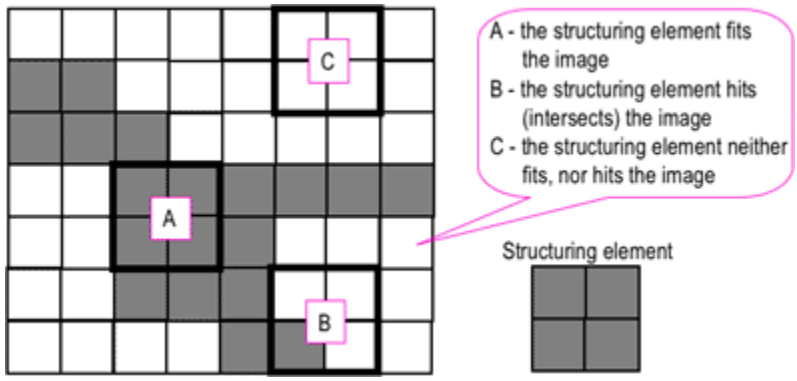
(a) points belonging to a regional minimum;
(b) points at which a drop of water, if places at the location of any of those points, would fall with certainty to a single minimum; and
(c) Points at which water would be equally likely to fall to more than one such minimum.

For a particular regional minimum, the set of points satisfying condition (b) is called the ***catchment basin*** or ***watershed*** of that minimum.

Binary images may contain numerous imperfections. In particular, the binary regions produced by simple thresholding are distorted by noise and texture. Morphological image processing pursues the goals of removing these imperfections by accounting for the form and structure of the image. These techniques can be extended to grey scale images. Morphology is a broad set of image processing operations that process images based on shapes. Morphological operations apply a structuring element to an input image, creating an output image of the same size. In a morphological operation, the value of each pixel in the output image is based on a comparison of the corresponding pixel in the input image with its neighbors. By choosing the size and shape of the neighborhood, you can construct a morphological operation that is sensitive to specific shapes in the input image.

Morphological techniques probe an image with a small shape or template called a **structuring element**.

The structuring element is positioned at all possible locations in the image and it is compared with the corresponding neighborhood of pixels. Some operations test whether the element "fits" within the neighborhood, while others test whether it "hits" or intersects the neighborhood:

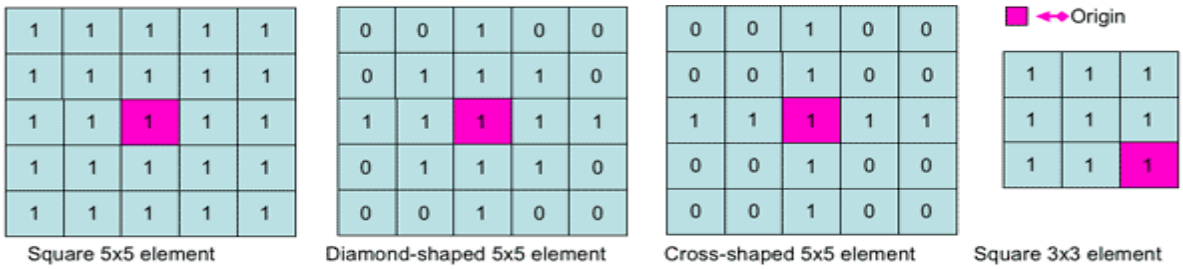


Probing of an image with a structuring element
(white and grey pixels have zero and non-zero values, respectively).

A morphological operation on a binary image creates a new binary image in which the pixel has a non-zero value only if the test is successful at that location in the input image.

The **structuring element** is a small binary image, i.e. a small matrix of pixels, each with a value of zero or one:

- The matrix dimensions specify the *size* of the structuring element.
- The pattern of ones and zeros specifies the *shape* of the structuring element.
- An *origin* of the structuring element is usually one of its pixels, although generally the origin can be outside the structuring element.



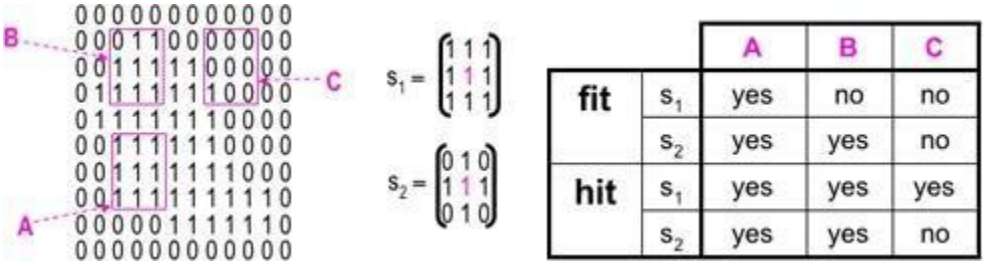
Examples of simple structuring elements.

A common practice is to have odd dimensions of the structuring matrix and the origin defined as the centre of the matrix. Structuring elements play in morphological image processing the same role as convolution kernels in linear image filtering.

When a structuring element is placed in a binary image, each of its pixels is

associated with the corresponding pixel of the neighborhood under the structuring element. The structuring element is said to **fit** the image if, for each of its pixels set to 1, the corresponding image pixel is

also 1. Similarly, a structuring element is said to **hit**, or intersect, an image if, at least for one of its pixels set to 1 the corresponding image pixel is also 1.



		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

Fitting and hitting of a binary image with structuring elements s_1 and s_2.

Zero-valued pixels of the structuring element are ignored, i.e. indicate points where the corresponding image value is irrelevant.

Fundamental operations:

The most basic morphological operations are dilation and erosion

Dilation adds pixels to the boundaries of objects in an image, while erosion removes pixels on object boundaries.The number of pixels added or removed from the objects in an image depends on the size and shape of the structuring element used to process the image.

In the morphological dilation and erosion operations, the state of any given pixel in the output image is determined by applying a rule to the corresponding pixel and its neighbors in the input image. The rule used to process the pixels defines the operation as dilation or erosion. This table lists the rules for both dilation and erosion.

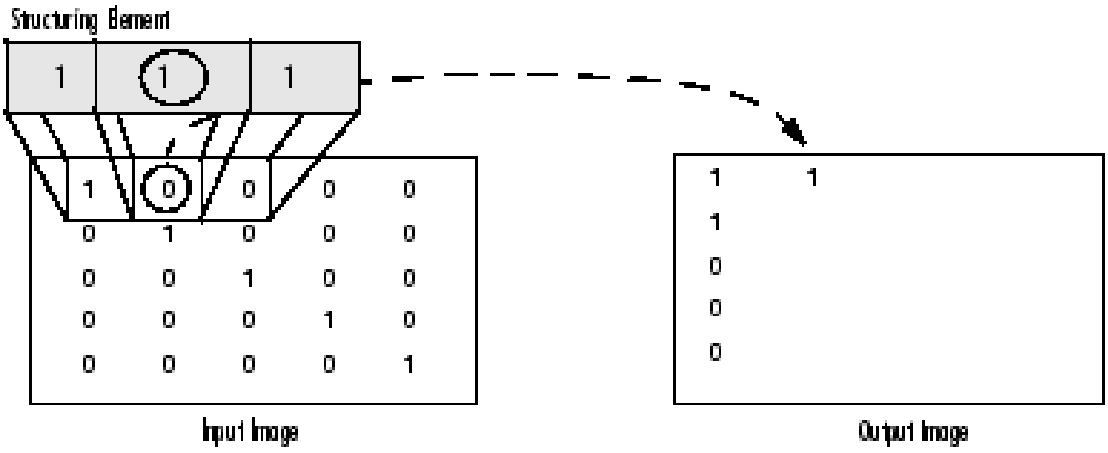
Rules for Dilation and Erosion

Operation	Rule
Dilation	The value of the output pixel is the *maximum* value of all the pixels in the input pixel's neighborhood. In a binary image, if any of the pixels is set to the value 1, the output pixel is set to 1.
Erosion	The value of the output pixel is the *minimum* value of all the pixels in the input pixel's neighborhood. In a binary image, if any of the pixels is set to 0, the output pixel is set to 0.

The following figure illustrates the dilation of a binary image. Note how the

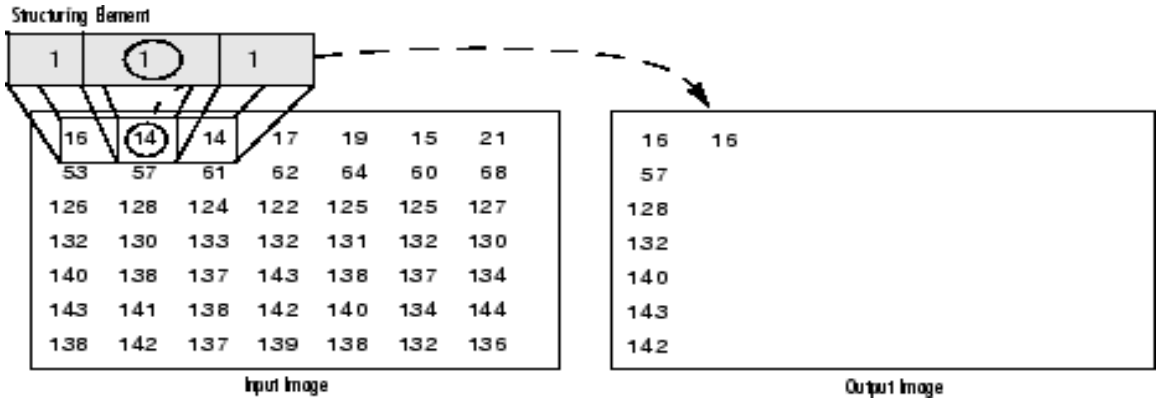
structuring element defines the neighborhood of the pixel of interest, which is circled. The dilation function applies the appropriate rule to the pixels in the neighborhood and assigns a value to the corresponding pixel in the output image. In the figure, the morphological dilation function sets the value of the output pixel to 1 because one of the elements in the neighborhood defined by the structuring element is on.

Morphological Dilation of a Binary Image



The following figure illustrates this processing for a grayscale image. The figure shows the processing of a particular pixel in the input image. Note how the function applies the rule to the input pixel's neighborhood and uses the highest value of all the pixels in the neighborhood as the value of the corresponding pixel in the output image.

Morphological Dilation of a Grayscale Image



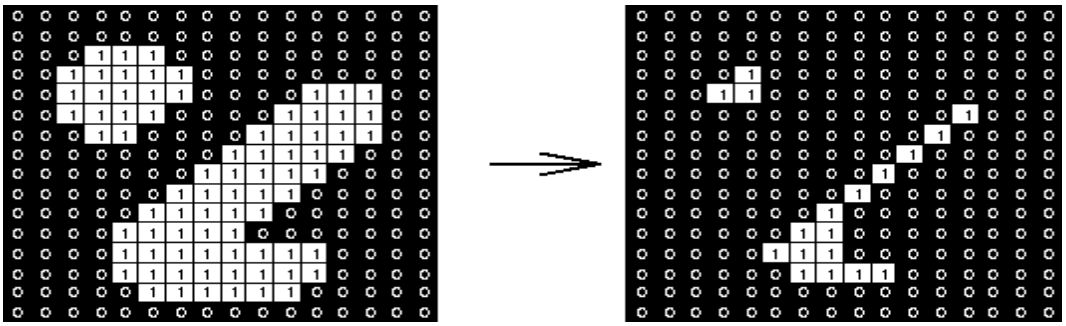
The **erosion** of a binary image f by a structuring element s (denoted $f \ominus s$) produces a new binary image $g = f \ominus s$ with ones in all locations (x,y) of a structuring element's origin at

which that structuring element s fits the input image f, i.e. $g(x,y) = 1$ is s fits f and 0 otherwise, repeating for all pixel coordinates (x,y).

Greyscale image | Binary image by thresholding | Erosion: a 2×2 square structuring element

http://documents.wolfram.com/applications/digitalimage/UsersGuide/Morphology/ImageProcessing6.3.html

Erosion with small (e.g. 2×2 - 5×5) square structuring elements shrinks an image by stripping away a layer of pixels from both the inner and outer boundaries of regions. The holes and gaps between different regions become larger, and small details are eliminated:



Erosion: a 3×3 square structuring element

Larger structuring elements have a more pronounced effect, the result of erosion with a large structuring element being similar to the result obtained by iterated erosion using a smaller structuring element of the same shape. If s_1 and s_2 are a pair of structuring elements identical in shape, with $s_2$twice the size of s_1, then

The **dilation** of an image f by a structuring element s (denoted $f \oplus s$) produces a new binary image $g = f \oplus s$ with ones in all locations (x,y) of a structuring element's origin at which that structuring element s hits the input image f, i.e. $g(x,y) = 1$ if s hits f and 0 otherwise, repeating for all pixel coordinates (x,y). Dilation has the opposite effect to erosion -- it adds a layer of pixels to both the inner and outer boundaries of regions.

Binary image Dilation: a 2×2 square structuring element

Results of dilation or erosion are influenced both by the size and shape of a structuring element. Dilation and erosion are *dual* operations in that they have opposite effects. Let f^c denote the complement of an image f, i.e., the image produced by replacing 1 with 0 and vice versa.

Formally, the duality is written as

$$f \ominus s = (f^c \oplus s_{rot})^c$$

Where s_{rot} is the structuring element s rotated by 180^0. If a structuring element is symmetrical with respect to rotation, then s_{rot} does not differ from s. If a binary image is considered to be a collection of connected regions of pixels set to 1 on a background of pixels set to 0, then erosion is the fitting of a structuring element to these regions and dilation is the fitting of a structuring element (rotated if necessary) into the background, followed by inversion of the result.

Processing Pixels at Image Borders (Padding Behavior)

Morphological functions position the origin of the structuring element, its center element, over the pixel of interest in the input image. For pixels at the edge of an image, parts of the neighborhood defined by the structuring element can extend past the border of the image.

To process border pixels, the morphological functions assign a value to these undefined pixels, as if the functions had padded the image with additional rows and columns. The value of these padding pixels varies for dilation and erosion operations. The following table describes the padding rules for dilation and erosion for both binary and gray scale images.

Rules for Padding Images

Operation	Rule
Dilation	Pixels beyond the image border are assigned the minimum value afforded by the data type. For binary images, these pixels are assumed to be set to 0. For grayscale images, the minimum value for `uint8` images is `0`.
Erosion	Pixels beyond the image border are assigned the *maximum* value afforded by the data type. For binary images, these pixels are assumed to be set to `1`. For grayscale images, the maximum value for `uint8` images is `255`.

Compound operations

Many morphological operations are represented as combinations of erosion, dilation, and simple set-theoretic operations such as the **complement** of a binary image:

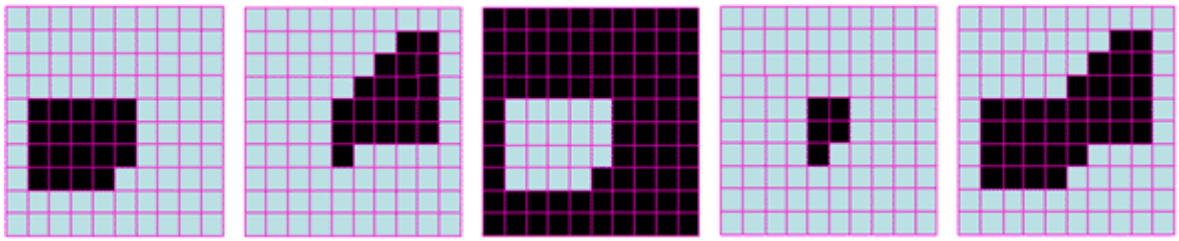
$f^c(x,y) = 1$ if $f(x,y) = 0$, and $f^c(x,y) = 0$ if $f(x,y) = 1$,

The **intersection** $h = f \cap g$ of two binary images f and g:

$$h(x,y) = 1 \text{ if } f(x,y) = 1 \text{ and } g(x,y) = 1, \text{ and } h(x,y) = 0$$

otherwise, and the **union** $h = f \cup g$ of two binary images f and g:

$$h(x,y) = 1 \text{ if } f(x,y) = 1 \text{ or } g(x,y) = 1, \text{ and } h(x,y) = 0 \text{ otherwise:}$$



Set operations on binary images: from left to right: a binary image f, a binary image g, the complement f^c of f, the intersection $f \cap g$, and the union $f \cup g$.

The **opening** of an image f by a structuring element s (denoted by $f \circ s$) is an erosion followed by a dilation:

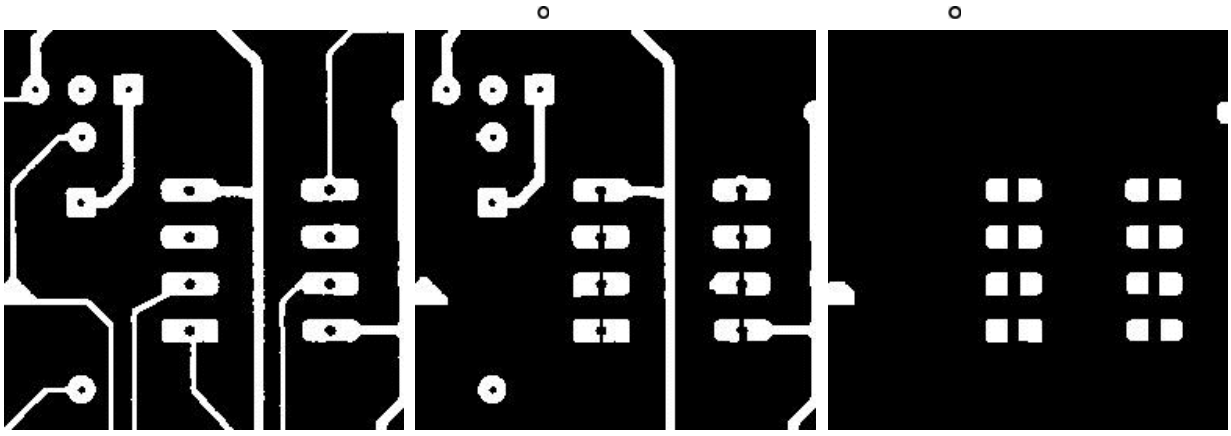
$$f \circ s = (f \ominus s) \oplus s$$

Binary image

Opening: a 2×2 square structuring element

http://documents.wolfram.com/applications/digitalimage/UsersGuide/Morphology/ImageProcessing6.3.html

Opening is so called because it can open up a gap between objects connected by a thin bridge of pixels. Any regions that have survived the erosion are restored to their original size by the dilation:



Binaryimage*f* *f* ∘ *s*(5×5square) *f* ∘ *s* (9×9 square)

Opening is an **idempotent** operation: once an image has been opened, subsequent openings with the same structuring element have no further effect on that image:

$$(f \circ s) \circ s = f \circ s$$

The **closing** of an image f by a structuring element s (denoted by $f \bullet s$) is a dilation followed by erosion:

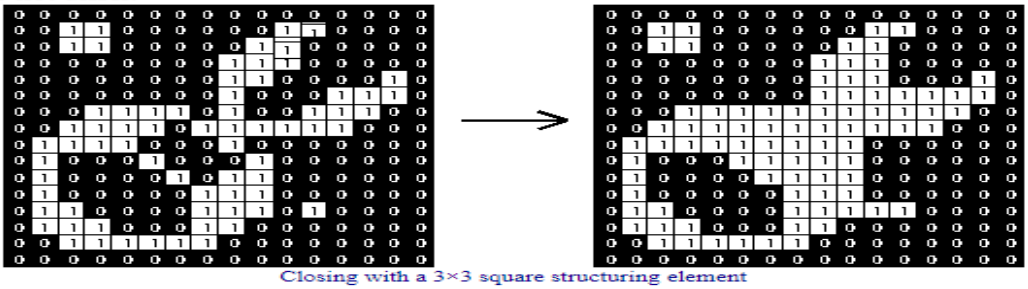
$$f \bullet s = (f \oplus s_{rot}) \ominus s_{rot}$$

Binary image

Closing: a 2×2 square structuring element

http://documents.wolfram.com/applications/digitalimage/UsersGuide/Morphology/ImageProcessing6.3.html

rosion should be performed with a rotated by 180° structuring element. Typically, the latte
it do not differ.



Closing with a 3×3 square structuring element
(www.cs.princeton.edu/~pshilane/class/mosaic/).

Closing is so called because it can fill holes in the regions while keeping the initial region sizes. Like opening, closing is idempotent: $(f \bullet s) \bullet s = f \bullet s$, and it is dual operation of opening (just as opening is the dual operation of closing):

$$f \bullet s = (f^c \circ s)^c; \qquad f \circ s = (f^c \bullet s)^c.$$

In other words, closing (opening) of a binary image can be performed by taking the complement of that image, opening (closing) with the structuring element, and taking the complement of the result.

The **hit and miss transform** allows to derive information on how objects in a binary image are related to their surroundings. The operation requires a matched pair of structuring elements, $\{s_1, s_2\}$, that probe the inside and outside, respectively, of objects in the image:

$$f \circledast \{s_1, s_2\} = (f \ominus s_1) \cap (f^c \ominus s_2)$$

Binary image

Hit and miss transform:
an elongated 2×5 structuring element

http://documents.wolfram.com/applications/digitalimage/UsersGuide/Morphology/ImageProcessing6.3.html

A pixel belonging to an object is preserved by the hit and miss transform if and only if s_1 translated to that pixel fits inside the object AND $s_2$translated to that pixel fits outside the object. It is assumed that s_1 and s_2 do not intersect; otherwise it would be impossible for both fits to occur simultaneously.

It is easier to describe it by considering s_1 and s_2 as a single structuring element with 1s for pixels of s_1 and 0s for pixels of s_2; in this case the hit-and-miss transform assigns 1 to an output pixel only if the object (with the value of 1) and background (with the value of 0) pixels in the structuring element exactly match object (1) and background (0) pixels in the input image. Otherwise that pixel is set to the background value (0).

The hit and miss transform can be used for detecting specific shapes (spatial arrangements of object and background pixel values) if the two structuring elements present the desired shape, as well as for thinning or thickening of object linear elements.

Morphological filtering of a binary image is conducted by considering compound operations like opening and closing as filters. They may act as filters of shape. For example, opening with a disc structuring element smoothes corners from the inside, and closing with a disc smoothes corners from the outside. But also these operations can filter out from an image any details that are smaller in size than the structuring element, e.g. opening is filtering the binary image at a scale defined by the size of the structuring element.

Only those portions of the image that fit the structuring element are passed by the

filter; smaller structures are blocked and excluded from the output image. The size of the structuring element is most important to eliminate noisy details but not to damage objects of interest.

Opening (morphology)

In mathematical morphology, **opening** is the dilation of the erosion of a set A by a structuring element B:

$$A \circ B = (A \ominus B) \oplus B,$$

Where $\ominus$ and $\oplus$ denote erosion and dilation, respectively.

Together with closing, the opening serves in computer vision and image processing as a basic workhorse of morphological noise removal. Opening removes small objects from the foreground (usually taken as the bright pixels) of an image, placing them in the background, while closing removes small holes in the foreground, changing small islands of background into foreground. These techniques can also be used to find specific shapes in an image. Opening can be used to find things into which a specific structuring element can fit (edges, corners, ...).

One can think of B sweeping around the inside of the boundary of A, so that it does not extend beyond the boundary, and shaping the A boundary around the boundary of the element.

Properties

- Opening is idempotent, thatis, $(A \circ B) \circ B = A \circ B$.
- Opening is increasing, that is, if $A \subseteq C$, then $A \circ B \subseteq C \circ B$. .
- Opening is anti-extensive,i.e., $A \circ B \subseteq A$.
- Opening is translation invariant.

Opening and closing satisfy the duality $A \bullet B = (A^c \circ B^c)^c$,where $\bullet$ denotes closing.

Closing (morphology)

Closing is an important operator from the field of mathematical morphology. Like its dual operator opening, it can be derived from the fundamental operations of erosion and dilation. Like those operators it is normally applied to binary images, although there are gray level versions. Closing is similar in some ways to dilation in that it tends to enlarge the

boundaries of foreground (bright) regions in an image (and shrink background color holes in such regions), but it is less destructive of the original boundary shape. As with other morphological operators, the exact operation is determined by a structuring element. The effect of the operator is to preserve *background* regions that have a similar shape to this structuring element, or that can completely contain the structuring element, while eliminating all other regions of back ground pixels.

Closing is opening performed in reverse. It is defined simply as dilation followed by erosion *using the same structuring element for both operations*. See the sections on erosion and dilation for details of the individual steps. The closing operator therefore requires two inputs: an image to be closed and a structuring element.

Gray level closing consists straightforwardly of a gray level dilation followed by a gray level erosion.

Closing is the dual of opening, *i.e.* closing the foreground pixels with a particular structuring element, is equivalent to closing the background with the same element.

One of the uses of dilation is to fill in small background color holes in images, *e.g.* `pepper noise'. One of the problems with doing this, however, is that the dilation will also distort *all* regions of pixels indiscriminately. By performing erosion on the image after the dilation, *i.e.* a closing, we reduce some of this effect. The effect of closing can be quite easily visualized. Imagine taking the structuring element and sliding it around *outside* each foreground region, without changing its orientation. For any background boundary point, if the structuring element can be made to touch that point, without any part of the element being inside a foreground region, then that point remains background. If this is not possible, then the pixel is set to foreground. After the closing has been carried out the background region will be such that the structuring element can be made to cover any point in the background without any part of it also covering a foreground point, and so further closings will have no effect. This property is known as *idempotence*. The effect of a closing on a binary image using a 3×3 square structuring element is illustrated in Figure1.

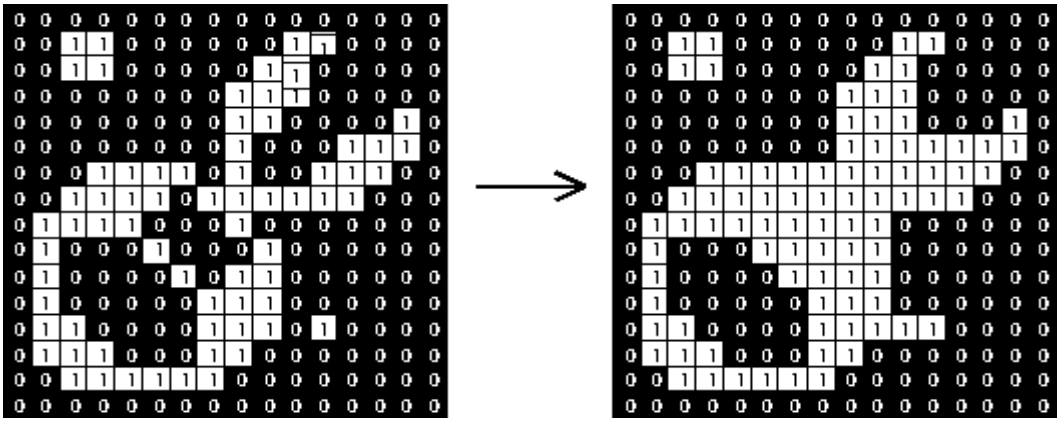


Fig: Effect of closing using a 3×3 square structuring element

As with erosion and dilation, this particular 3×3 structuring element is the most commonly used, and in fact many implementations will have it hardwired into their code, in which case it is obviously not necessary to specify a separate structuring element. To achieve the effect of a closing with a larger structuring element, it is possible to perform multiple dilations followed by the same number of erosions.

Closing can sometimes be used to selectively fill in particular background regions of an image. Whether or not this can be done depends upon whether a suitable structuring element can be found that fits well inside regions that are to be preserved, but doesn't fit inside regions that are to be removed.

The image is the result of a closing with a 22 pixel diameter disk. Note that the thin black ring has also been filled in as a result of the closing operation.

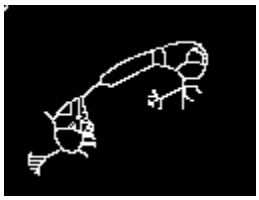
In real world applications, closing can, for example, be used to enhance binary images of

objects obtained from thresholding. Consider that we want compute the skeleton of

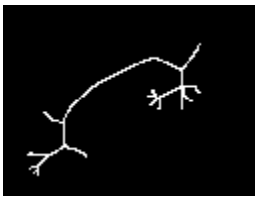
To do this we first need to transform the gray level image into a binary image. Simply thresholding the image at a value of *100* yields

We can see that the threshold classified some parts of the receiver as background. The image

is the result of closing the thresholded, image with a circular structuring element of size *20*. The merit of this operator becomes obvious when we compare the skeletons of the two binary images. The image



is the skeleton of the image which was only thresholded and

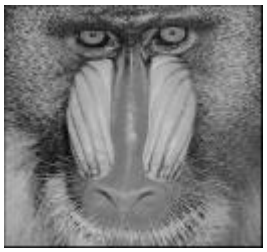


is the skeleton of the image produced by the closing operator. We can see that the latter skeleton is less complex and it better represents the shape of the object.

Unlike erosion and dilation, the position of the origin of the structuring element does not really matter for opening and closing. The result is independent of it.

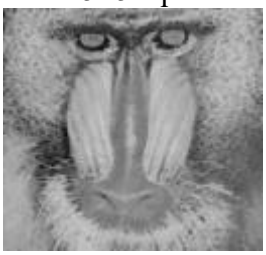
Gray level closing can similarly be used to select and preserve particular intensity patterns while attenuating others.

The image



is our starting point.

The result of gray level closing with a flat 5×5 square structuring element is shown in



Notice how the dark specks in between the bright spots in the hair have been largely filled in to the same color as the bright spots, while the more uniformly colored nose area is largely the same intensity as before. Similarly the gaps between the white whiskers have been filled in.

Closing can also be used to remove `pepper noise' in images.The image is an image containing pepper noise.

The result of a closing with a 3×3 square structuring element is shown .The noise has been

completely removed with only a little degradation to the underlying image. If, on the other hand, the noise consists of bright spots (*i.e.* `salt noise'), as can be seen in closing yields

Here, no noise has been removed. The noise has even been increased at locations where two nearby noise pixels have merged together into one larger spot. Compare these results with the ones achieved on the same image using opening.

Although closing can sometimes be used to preserve particular intensity patterns in an image while attenuating others, this is not always the case. Some aspects of this problem are discussed under opening.

Common Variants

Opening and closing are themselves often used in combination to achieve more subtle results. If we represent the closing of an image f by $C(f)$, and its opening by $O(f)$, then some common combinations include:

Proper Opening

$$\text{Min}(f, /\text{em}\{\text{C}\}(O(C(f))))$$

Proper Closing

$$\text{Max}(f, O(C(O(f))))$$

Auto median Filter

Max($O(C(O(f))$, Min(f, $C(O(C(f))))$)

THE HIT OR MISS TRANSFORMATION:

It is a morphological operation where we mainly focus on finding patterns of foreground and background object.

For this we use two conditions:

(1) Hit condition: If the foreground and background pixels in the structuring elements exactly matches with the foreground and background pixels in the image then it is ***hit condition***, and the pixels below the origin of structuring element is bet to the foreground color.
(2) Miss condition: If foreground and background pixels in element does not match exactly then it is ***miss condition***.

The hit or miss transformation of an image A by B is denoted by $A \circledast B$. B is a pair of structuring elements $B = (B_1, B_2)$ rather than a single element.

B1: set of elements of B associated with an object

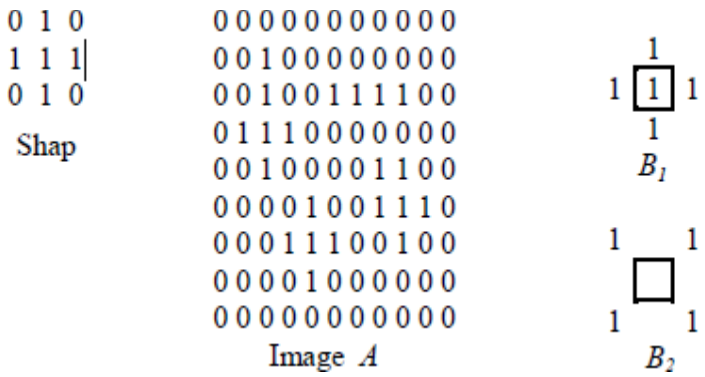
B2: set of elements of B associated with the

background The Hit or Miss transform is defined as follows:

$$A \circledast B = (A \ominus B_1) \cap (A^c \ominus B_2)$$

This transform is useful in locating all pixel configurations that match the B1 structure (i.e. a hit) but do not match that B2 (i.e. a miss). Thus, the hit-or-miss transform is used for shape detection.

Example : Use the hit-or-miss transform to identify the locations of the following shape pixel configuration in the image below using the two structuring elements B1 and B2.



Solution:

$A \ominus B_1 =$

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 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 0 1 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 0 0 0 0

$A^c =$

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

$A^c \ominus B_2 =$

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

$A \circledast B =$

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 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 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 0 0 0 0

The figure below shows an example of applying the hit-or-miss transform on the image in the previous example.

Fig. (a)Binaryimage. (b) Result of applying hit-or-misstransform.

$A \circledast B = (A \ominus B) \cap (A^c \ominus B_2)$

Foreground Background

$= (A \ominus B) \cap (A^c \ominus (W-B))$ where B2 is W-B, let Wis

1	1
1	1

Let A as

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

And B as

1	0
0	1

$A \ominus B =$

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

Let W =

1	1
1	1

W – B =

0	1
1	0

A^c =

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

Then $A^c \ominus (W-B)$=

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

THEN $A \otimes B$ =

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

$=(A \ominus B) \cap (Ac \ominus (W-B))$

$A \otimes B = \{ a/B_1 \in A \ \& \ B_2 \in A^c \}$ where A^c is the complement of A. (originalimage)

$=(A \ominus B_1) \cap (A^c \ominus B_2)$

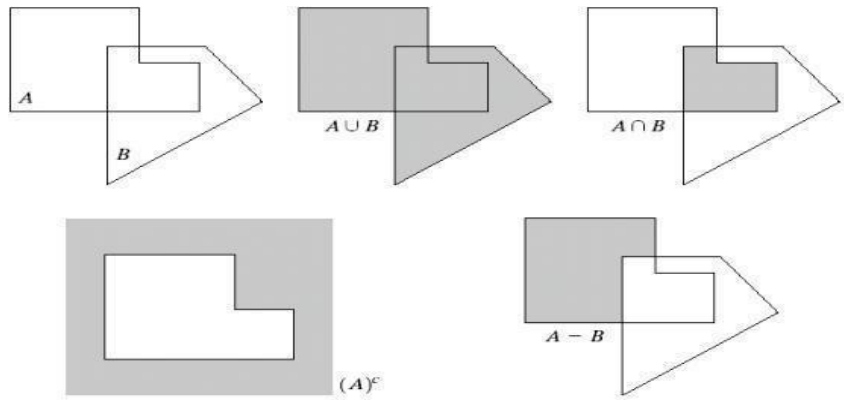


Fig. shows some of the basic operations in pictorial form.

Basic Morphological Algorithms (Applications)

The principle application of morphology is extracting image components that are useful in the representation and description of shape. Morphological algorithms are used for boundaries extraction, skeletonization (i.e. extracting the skeleton of an object), and thinning.

Boundary Extraction

The boundary of a set A, denoted by $\beta(A)$, can be obtained by:

$$\beta(A) = A - (A \ominus B)$$

where B is the structuring element.

The figure below shows an example of extracting the boundary of an object in a binary image.

(a) (b)

Figure 12.2 (a) Binary image. (b) Object boundary extracted using the previous equation and 3×3 square structuring element.

Note that, because the size of structuring element is 3×3 pixels, the resulted boundary is one pixel thick. Thus, using 5×5 structuring element will produce a boundary between 2 and 3 pixels thick as shown in the next figure.

Figure 12.3 Object boundary extracted using 5×5 square structuring element

Thinning

Thinning means reducing binary objects or shapes in an image to strokes that are a single pixel wide. The thinning of a set A by a structuring element B, is defined as:

$$A \otimes B = A - (A \circledast B)$$
$$= A \cap (A \circledast B)^c$$

Since we only match the pattern (shape) with the structuring elements, no background operation is required in the hit-or-miss transform.

Here, B is a sequence of structuring elements:

$$\{B\} = \{B^1, B^2, B^3, \dots, B^n\}$$

where B^i is the rotation of B^{i-1}. Thus, the thinning equation can be written as:

$$A \otimes \{B\} = ((\dots((A \otimes B^1) \otimes B^2) \dots) \otimes B^n)$$

The entire process is repeated until no further changes occur. The next figure shows an example of thinning the fingerprint ridges so that each is one pixel thick.

Figure 12.4 (a) Original fingerprint image. (b) Image thinned once. (c) Image thinned twice. (d) Image thinned until stability (no changes occur).

Skeletonization (Skeleton Extraction)

is another way to reduce binary objects to thin strokes that retain important structural information about the shapes of the original objects. The skeleton of A can be expressed in terms of erosions and openings as follows:

$$S(A) = \bigcup_{k=0}^{K} S_k(A)$$

with

$$S_k(A) = (A \ominus kB) - (A \ominus kB) \circ B$$

where B is a structuring element, and $(A \ominus kB)$ indicates k successive erosions of A:

The figure below shows the result of dilating a gray image using a 3×3 square structuring element.

(a) (b)

Figure 12.6 (a) Original gray image. (b) Dilated image.

We can see that gray-scale dilation produces the following:

1. Bright and slightly blurred image.
2. Small, dark details have been reduced.

Gray-scale Morphology

The basic morphological operations of dilation, erosion, opening and closing can also be applied to gray images.

Gray-scale Dilation

The *gray-scale dilation* of a gray-scale image f by a structure element b is defined as:

$$(f \oplus b)(x, y) = \max\{f(x - x', y - y') + b(x', y') \mid (x', y') \in D_b\}$$

where D_b is the domain of the structuring element b. This process operates in the same way as the spatial convolution.

Gray-scale Erosion

The *gray-scale erosion* of a gray-scale image f by a structure element b is defined as:

$$(f \ominus b)(x,y) = \min\{f(x+x', y+y') - b(x',y') \mid (x',y') \in D_b\}$$

The next figure shows the result of eroding a gray image using a 3×3 square structuring element.

(a) (b)

Figure 12.7 (a) Original gray image. (b) Eroded image.

We can see that gray-scale erosion produces the following:

1. Dark image.
2. Small, bright details were reduced.

Gray-scale Opening and Closing

The opening and closing of gray-scale images have the same form as in the binary images:

$$\text{Opening}: \quad f \circ b = (f \ominus b) \oplus b$$

$$\text{Closing}: \quad f \bullet b = (f \oplus b) \ominus b$$

The figure below shows the result of opening a gray image.

(a) (b)

Figure 12.8 (a) Original gray image. (b) Opened image.

UNIT-V

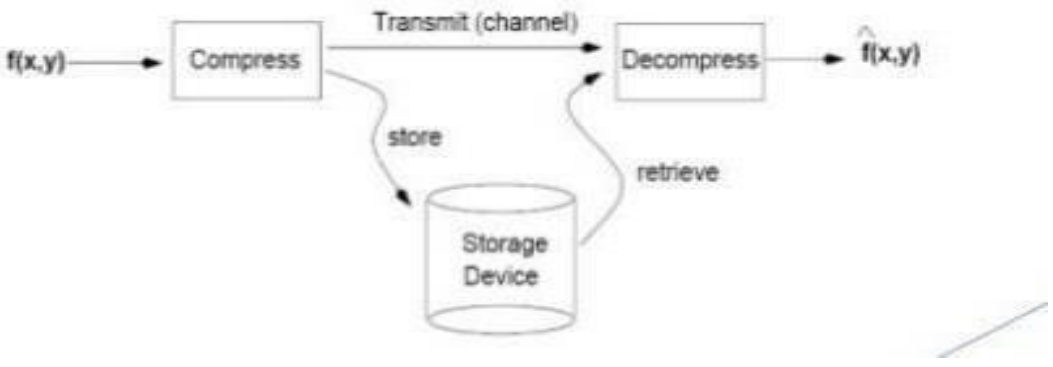
IMAGE COMPRESSION

Redundancies and their Removal Methods, Fidelity Criteria, Image Compression Models, Huffman and Arithmetic Coding, Error Free Compression, Lossy Compression, Lossy and Lossless Predictive Coding, Transform Based Compression, JPEG 2000 Standards.

Definition: Image compression deals with reducing the amount of data required to represent a digital image by removing of redundant data.

Images can be represented in digital format in many ways. Encoding the contents of a 2-D image in a raw bitmap (raster) format is usually not economical and may result in very large files. Since raw image representations usually require a large amount of storage space (and proportionally long transmission times in the case of file uploads/ downloads), most image file formats employ some type of compression. The need to save storage space and shorten transmission time, as well as the human visual system tolerance to a modest amount of loss, have been the driving factors behind image compression techniques.

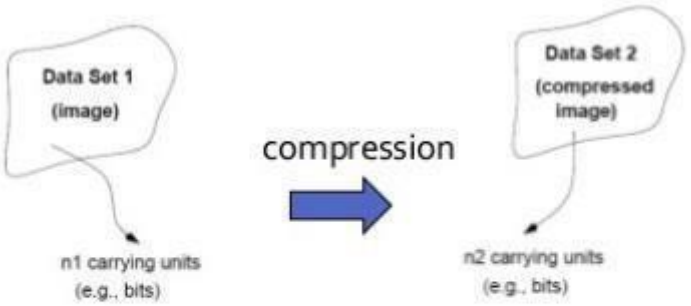
Goal of image compression: The goal of image compression is to reduce the amount of data required to represent a digital image.



Data ≠Information:

- Data and information are not synonymous terms!
- Data is the means by which information is conveyed.
- Datacompressionaimstoreducetheamountofdatarequiredtorepresentagiven quality of information while preserving as much information as possible.
- The same amount of information can be represented by various amount of data. Ex1: You have an extra class after completion of 3.50p.m
  Ex2: Extra class have been scheduled after 7th hour for you.
  Ex3: After 3.50 p.m you should attended extra class.

Definition of compression ratio:



Compression ratic $C_R = \frac{n_1}{n_2}$

Definitions of Data Redundancy:

➢ Relative data redundancy: $R_D = 1 - \frac{1}{C_R}$

Example:

If $C_R = \frac{10}{1}$, then $R_D = 1 - \frac{1}{10} = 0.9$

(90% of the data in dataset 1 is redundant)

if $n_2 = n_1$, then C_R=1, R_D=0

if $n_2 \ll n_1$, then $C_R \to \infty$, $R_D \to 1$

Coding redundancy:

- Code: a list of symbols (letters, numbers, bitsetc.,)
- Code word: a sequence of symbol used to represent a piece of information or anevent (e.g., graylevels).
- Code word length: number of symbols in each codeword.

Example: (binary code, symbols: 0,1, length: 3)

0: 000	4: 100
1: 001	5: 101
2: 010	6: 110
3: 011	7: 111

N x M image
r_k: k-th gray level
$P(r_k)$: probability of r_k
$l(r_k)$: # of bits for r_k

Expected value:

$$E(X) = \sum_x xP(X = x)$$

Average # of bits: $L_{avg} = E(l(r_k)) = \sum_{k=0}^{L-1} l(r_k)P(r_k)$

Total # of bits: NML_{avg}

Coding Redundancy (con'd)

Case 1: $l(r_k)$ = constant length

Example:

r_k	$p(r_k)$	Code 1	$l(r_k)$
$r_0 = 0$	0.19	000	3
$r_1 = 1/7$	0.25	001	3
$r_2 = 2/7$	0.21	010	3
$r_3 = 3/7$	0.16	011	3
$r_4 = 4/7$	0.08	100	3
$r_5 = 5/7$	0.06	101	3
$r_6 = 6/7$	0.03	110	3
$r_7 = 1$	0.02	111	3

Assume an image with $L = 8$

Assume $l(r_k) = 3$, $L_{avg} = \sum_{k=0}^{7} 3P(r_k) = 3\sum_{k=0}^{7} P(r_k) = 3$ bits

Total number of bits: $3NM$

Case 2: $l(r_k)$ = variable length

Table 6.1 Variable-Length Coding Example

r_k	$p(r_k)$	Code 1	$l(r_k)$	Code 2	$l(r_k)$
$r_0 = 0$	0.19	000	3	11	2
$r_1 = 1/7$	0.25	001	3	01	2
$r_2 = 2/7$	0.21	010	3	10	2
$r_3 = 3/7$	0.16	011	3	001	3
$r_4 = 4/7$	0.08	100	3	0001	4
$r_5 = 5/7$	0.06	101	3	00001	5
$r_6 = 6/7$	0.03	110	3	000001	6
$r_7 = 1$	0.02	111	3	000000	6

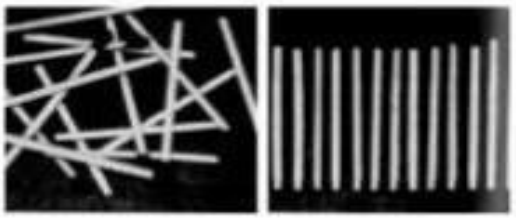
$$L_{avg} = \sum_{k=0}^{7} l(r_k)P(r_k) = 2.7 \text{ bits}$$

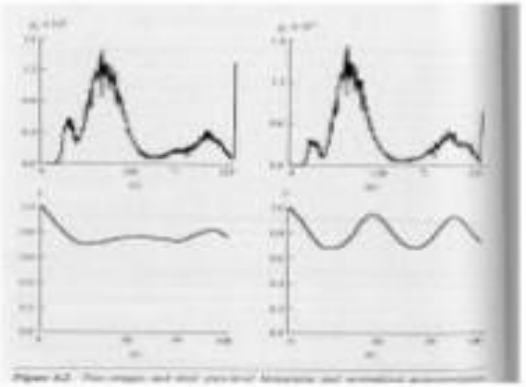
$$C_R = \frac{3}{2.7} = 1.11 \text{ (about 10\%)}$$

$$R_D = 1 - \frac{1}{1.11} = 0.099$$

Interpixel redundancy

- Interpixel redundancy implies that pixel values are correlated (i.e., a pixel value can be reasonably predicted by its neighbors).





$$f(x) \circ g(x) = \int_{-\infty}^{\infty} f(x) g(x+a) da$$

autocorrelation: f(x)=g(x)

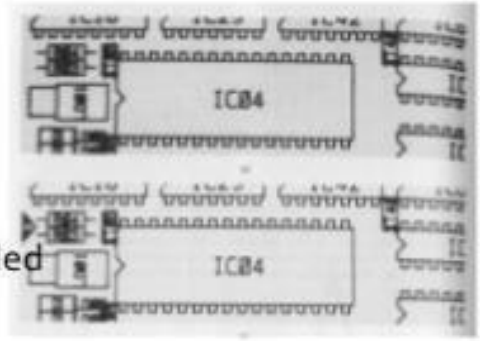
Interpixel redundancy (cont'd)

To reduce interpixel redundancy, the data must be transformed in another format (i.e., using a transformation)

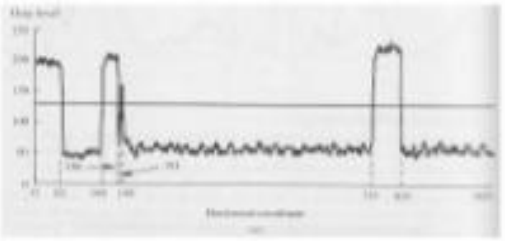
- e.g., thresholding, DFT, DWT, etc.

Example:

origin
al



thresholded



Run-length encoding:

(1,63) (0,87) (1,37) (0,5) (1,4) (0, 556) (1,62) (0,218)

Using 11 bits/pair:

88 bits are required (compared to 1024 ?!)

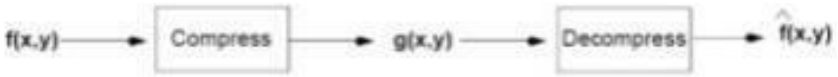
Psychovisual redundancy

The human eye does not respond with equal sensitivity to all visual information.

It is more sensitive to the lower frequencies than to the higher frequencies in the visual spectrum.

Idea: discard data that is perceptually insignificant!

Fidelity Criteria



$$\hat{f}(x,y) = f(x,y) + e(x,y)$$

- How close $f(x,y)$ $\hat{f}(x,y)$?

- Criteria
  - Subjective: based on human observers
  - Objective: mathematically defined criteria

Subjective Fidelity Criteria

Value	Rating	Description
1	Excellent	An image of extremely high quality, as good as you could desire.
2	Fine	An image of high quality, providing enjoyable viewing. Interference is not objectionable.
3	Passable	An image of acceptable quality. Interference is not objectionable.
4	Marginal	An image of poor quality; you wish you could improve it. Interference is somewhat objectionable.
5	Inferior	A very poor image, but you could watch it. Objectionable interference is definitely present.
6	Unusable	An image so bad that you could not watch it.

Objective Fidelity Criteria

- Root mean square error (RMS)

$$e_{rms} = \sqrt{\frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}(\hat{f}(x,y) - f(x,y))^2}$$

- Mean-square

$$SNR_{ms} = \frac{\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}(\hat{f}(x,y))^2}{\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}(\hat{f}(x,y) - f(x,y))^2}$$

COMPRESSION METHODS OF IMAGES:

Compression methods can be ***lossy,***

when a tolerable degree of deterioration in the visual quality of the resulting image is acceptable, or ***lossless,***

when the image is encoded in its full quality. The overall results of the compression process, both in terms of storage savings – usually expressed numerically in terms of compression ratio (CR) or bits per pixel (bpp) – as well as resulting quality loss (for the case of lossy techniques) may vary depending on the technique, format, options (such as the quality setting for JPEG), and the image contents.

As a general guideline, ***lossy compression*** should be used for general purpose photographic images.

Whereas ***lossless compression*** should be preferred when dealing with line art, technical drawings, cartoons, etc. or images in which no loss of detail may be tolerable (most notably, space images and medical images).

Fundamentals of visual data compression

The general problem of image compression is to reduce the amount of data required to represent a digital image or video and the underlying basis of the reduction process is the removal of redundant data. Mathematically, visual data compression typically involves

transforming (encoding) a 2-D pixel array into a statistically uncorrelated data set. This transformation is applied prior to storage or transmission. At some later time, the compressed image is decompressed to reconstruct the original image information (preserving or lossless techniques) or an approximation of it (lossy techniques).

Redundancy

Data compression is the process of reducing the amount of data required to represent a given quantity of information. Different amounts of data might be used to communicate the same amount of information. If the same information can be represented using different amounts of data, it is reasonable to believe that the representation that requires more data contains what is technically called ***data redundancy.***

Image compression and coding techniques explore three types of redundancies: ***coding*** redundancy, ***interpixel***(spatial) redundancy, and *psychovisual* redundancy. The way each of them is explored is briefly described below.

- **Coding redundancy:** consists in using variable-length code words selected as to match the statistics of the original source, in this case, the image itself or a processed version of its pixel values. This type of coding is always reversible and usually implemented using look-up tables (LUTs). Examples of image coding schemes that explore coding redundancy are the Huffman codes and the arithmetic coding technique.
- **Interpixel redundancy:** this type of redundancy – sometimes called spatial redundancy, inter frame redundancy, or geometric redundancy – exploits the fact that an image very often contains strongly correlated pixels, in other words, large regions whose pixel values are the same or almost the same. This redundancy can be explored in several ways, one of which is by predicting a pixel value based on the values of its neighboring pixels. In order to do so, the original 2-D array of pixels is usually mapped into a different format, e.g., an array of differences between adjacent pixels. If the original image pixels can be reconstructed from the transformed data set the mapping is said to be reversible. Examples of compression techniques that explore the interpixel redundancy include: Constant Area Coding (CAC), (1- D or 2-D) Run-Length Encoding (RLE) techniques, and many predictive coding algorithms such as Differential Pulse Code Modulation (DPCM).
- **Psycho visual redundancy: many experiments on the psychophysical aspects of human** vision have proven that the human eye does not respond with equal sensitivity to all incoming visual information; some pieces of information are more important than others. The

knowledge of which particular types of information are more or less relevant to the final human user have led to image and video compression techniques that aim at eliminating or reducing any amount of data that is psycho visually redundant. The end result of applying these techniques is a compressed image file, whose size and quality are smaller than the original information, but whose resulting quality is still acceptable for the application at hand.

The loss of quality that ensues as a byproduct of such techniques is frequently called *quantization,* as to indicate that a wider range of input values is normally mapped into a narrower range of output values thorough an irreversible process. In order to establish the nature and extent of information loss, different fidelity criteria (some objective such as root mean square (RMS) error, some subjective, such as pair wise comparison of two images encoded with different quality settings) can be used. Most of the image coding algorithms in use today exploit this type of redundancy, such as the Discrete Cosine Transform (DCT)- based algorithm at the heart of the JPEG encoding standard.

IMAGE COMPRESSION AND CODING MODELS

Figure shows a general image compression model. It consists of a source encoder, a channel encoder, the storage or transmission media (also referred to as *channel)*, a channel decoder, and a source decoder. The source encoder reduces or eliminates any redundancies in the input image, which usually leads to bit savings. Source encoding techniques are the primary focus of this discussion. The channel encoder increase noise immunity of source encoder's output, usually adding extra bits to achieve its goals. If the channel is noise-free, the channel encoder and decoder may be omitted. At the receiver's side, the channel and source decoder perform the opposite functions and ultimately recover (an approximation of) the original image.

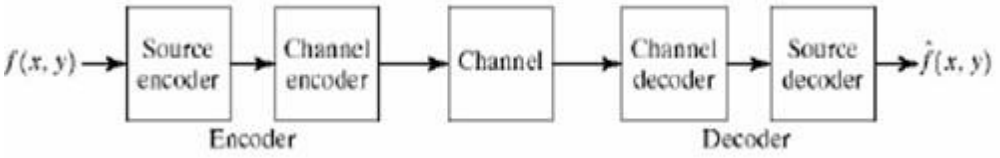


Figure Image compression model

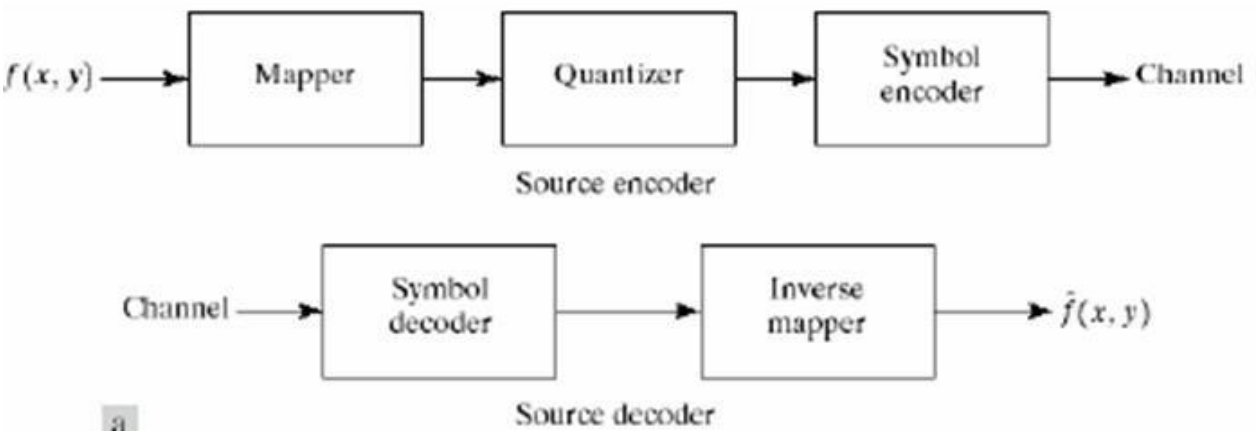


Figure shows the source encoder in further detail. Its main components are:

- **Mapper:** transforms the input data into a (usually nonvisual) format designed to reduce interpixel redundancies in the input image. This operation is generally reversible and may or may not directly reduce the amount of data required to represent the image.

- **Quantizer:** reduces the accuracy of the mapper's output in accordance with some pre-established fidelity criterion. Reduces the psychovisual redundancies of the input image. This operation is not reversible and must be omitted if lossless compression is desired.

- **Symbol (entropy) encoder:** creates a fixed- or variable-length code to represent the quantizer's output and maps the output in accordance with the code. In most cases, a variable-length code is used. This operation is reversible.

Error-free compression

Error-free compression techniques usually rely on entropy-based encoding algorithms. The concept of entropy is mathematically described in equation (1):

Where:

a_j is a symbol produced by the information source

$P(a_j)$ is the probability of that symbol

J is the total number of different symbols

$H(z)$ is the entropy of the source.

The concept of entropy provides an upper bound on how much compression can be achieved, given the probability distribution of the source. In other words, it establishes a theoretical limit on the amount of lossless compression that can be achieved using entropy encoding techniques alone.

Variable Length Coding (VLC)

Most entropy-based encoding techniques rely on assigning variable-length codewords to each symbol, whereas the most likely symbols are assigned shorter codewords. In the case of image coding, the symbols may be raw pixel values or the numerical values obtained at the output of the mapper stage (e.g., differences between consecutive pixels, run-lengths, etc.). The most popular entropy-based encoding technique is the Huffman code. It provides the least amount of information units (bits) per source symbol. It is described in more detail in a separate short article.

Run-length encoding (RLE)

RLE is one of the simplest data compression techniques. It consists of replacing a sequence (run) of identical symbols by a pair containing the symbol and the run length. It is used as the primary compression technique in the 1-D CCITT Group 3 fax standard and in conjunction with other techniques in the JPEG image compression standard (described in a separate short article).

Differential coding

Differential coding techniques explore the interpixel redundancy in digital images. The basic idea consists of applying a simple difference operator to neighboring pixels to calculate a difference image, whose values are likely to follow within a much narrower range than the original gray-level range. As a consequence of this narrower distribution – and consequently reduced entropy – Huffman coding or other VLC schemes will produce shorter code words for the difference image.

Predictive coding

Predictive coding techniques constitute another example of exploration of interpixel

redundancy, in which the basic idea is to encode only the new information in each pixel. This new information is usually defined as the difference between the actual and the predicted value of that pixel.

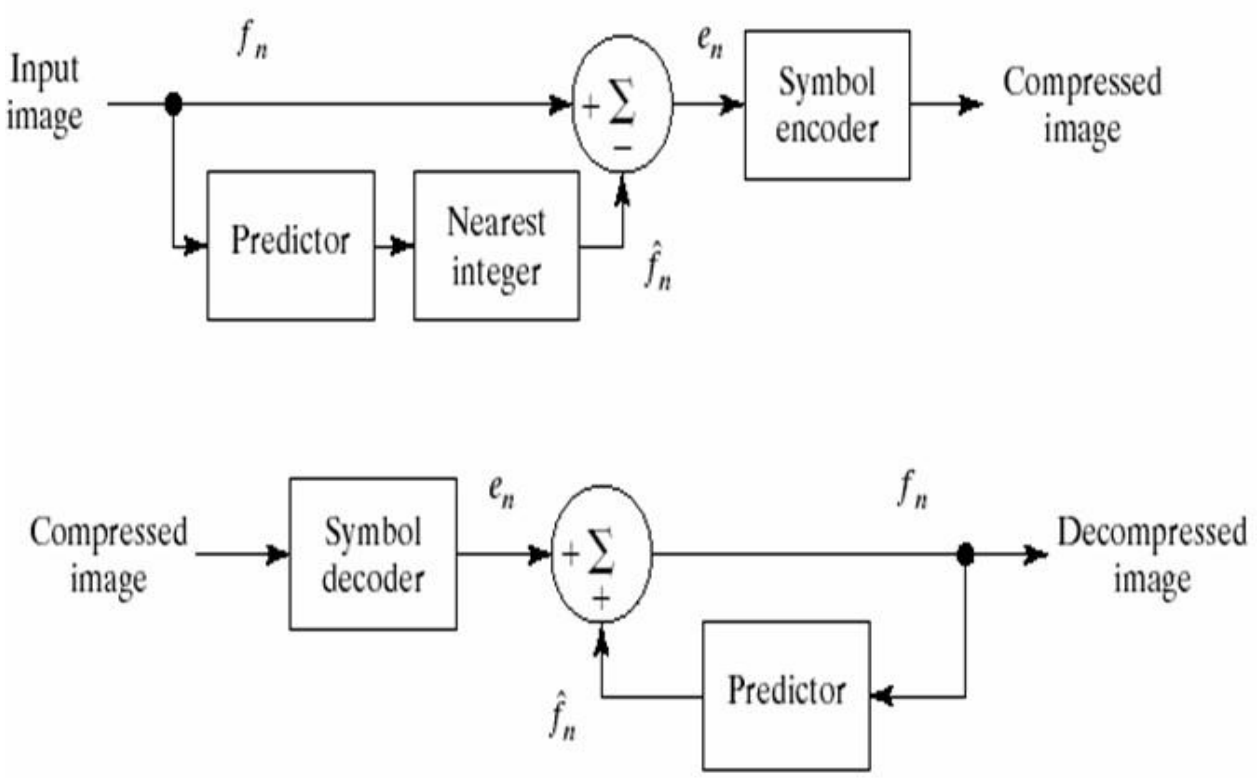


Figure 3 shows the main blocks of a lossless predictive encoder. The key component is the predictor, whose function is to generate an estimated (predicted) value for each pixel from the input image based on previous pixel values. The predictor's output is rounded to the nearest integer and compared with the actual pixel value: the difference between the two – called *prediction error* – is then encoded by a VLC encoder. Since prediction errors are likely to be smaller than the original pixel values, the VLC encoder will likely generate shorter code words.

There are several local, global, and adaptive prediction algorithms in the literature. In most cases, the predicted pixel value is a linear combination of previous pixels.

Dictionary-based coding

Dictionary-based coding techniques are based on the idea of incrementally building a dictionary (table) while receiving the data. Unlike VLC techniques, dictionary-based techniques use fixed-length code words to represent variable-length strings of symbols that commonly occur together. Consequently, there is no need to calculate, store, or transmit the probability

distribution of the source, which makes these algorithms extremely convenient and popular. The best-known variant of dictionary-based coding algorithms is the LZW (Lempel-Ziv-Welch) encoding scheme, used in popular multimedia file formats such as GIF, TIFF, andPDF.

Lossy compression

Lossy compression techniques deliberately introduce a certain amount of distortion to the encoded image, exploring the psychovisual redundancies of the original image. These techniques must find an appropriate balance between the amount of error (loss) and the resulting bit savings.

Quantization

The quantization stage is at the core of any lossy image encoding algorithm. Quantization, in at the encoder side, means partitioning of the input data range into a smaller set of values. There are two main types of quantizers: scalar quantizers and vector quantizers. A scalar quantizer partitions the domain of input values into a smaller number of intervals. If the output intervals are equally spaced, which is the simplest way todo it, the process is called *uniform scalar quantization;* otherwise, for reasons usually related to minimization of total distortion, it is called *non uniform scalar quantization.* One of the most popular non uniform quantizers is the Lloyd-Max quantizer. Vector quantization (VQ) techniques extend the basic principles of scalar quantization to multiple dimensions. Because of its fast lookup capabilities at the decoder side, VQ-based coding schemes are particularly attractive to multimedia applications.

Transform coding

The techniques discussed so far work directly on the pixel values and are usually called *spatial domain techniques.* Transform coding techniques use a reversible, linear mathematical transform to map the pixel values onto a set of coefficients, which are then

quantized and encoded. The key factor behind the success of transform-based coding schemes many of the resulting coefficients for most natural images have small magnitudes and can be quantized (or discarded altogether) without causing significant distortion in the decoded image. Different mathematical transforms, such as Fourier (DFT), Walsh-Hadamard (WHT), and Karhunen-Loeve (KLT), have been considered for the task. For compression purposes, the higher the capability of compressing information in fewer coefficients, the better the transform; for that reason, the Discrete Cosine Transform (DCT) has become the most widely used transform coding technique.

Wavelet coding

Wavelet coding techniques are also based on the idea that the coefficients of a transform that decorrelates the pixels of an image can be coded more efficiently than the original pixels themselves. The main difference between wavelet coding and DCT-based coding (Figure 4) isthe omission of the first stage. Because wavelet transforms are capable of representing an input signal with multiple levels of resolution, and yet maintain the useful compaction properties of the DCT, the subdivision of the input image into smaller sub images is no longer necessary. Wavelet coding has been at the core of the latest image compression standards, most notably JPEG 2000, which is discussed in a separate short article.

Image compression standards

Work on international standards for image compression started in the late 1970s with the CCITT (currently ITU-T) need to standardize binary image compression algorithms for Group 3 facsimile communications. Since then, many other committees and standards have been formed to produce *de jure* standards (such as JPEG), while several commercially successful initiatives have effectively become *de facto* standards (such as GIF). Image compression standards bring about many benefits, such as: (1) easier exchange of image files between different devices and applications; (2) reuse of existing hardware and software for a wider array of products; (3) existence of benchmarks and reference data sets for new and alternative developments.

Binary image compression standards

Work on binary image compression standards was initially motivated by CCITT Group 3 and 4 facsimile standards. The Group 3 standard uses a non-adaptive, 1-D RLE technique in which the last K-1 lines of each group of K lines (for K = 2 or 4) are optionally coded in a 2- D manner,

using the *Modified Relative Element Address Designate* (MREAD) algorithm. The Group 4 standard uses only the MREAD coding algorithm. Both classes of algorithms are non-adaptive and were optimized for a set of eight test images, containing a mix of representative documents, which sometimes resulted in data expansion when applied to different types of documents (e.g., half-tone images).. The Joint Bilevel Image Group (JBIG)– a joint committee of the ITU-T and ISO – has addressed these limitations and proposed two new standards (JBIG and JBIG2) which can be used to compress binary and gray-scale images of up to 6 gray-coded bits/pixel.

Continuous tone still image compression standards

For photograph quality images (both grayscale and color), different standards have been proposed, mostly based on lossy compression techniques. The most popular standard in this category, by far, is the JPEG standard, a lossy, DCT-based coding algorithm. Despite its great popularity and adoption, ranging from digital cameras to the World Wide Web, certain limitations of the original JPEG algorithm have motivated the recent development of two alternative standards, JPEG 2000 and JPEG-LS (lossless). JPEG, JPEG 2000, and JPEG-LS are described in separate short articles.

Encode each pixel ignoring their inter-pixel dependencies. Among methods are:

1. ***Entropy Coding:*** Every block of an image is entropy encoded based upon the Pk's within a block. This produces variable length code for each block depending on spatial activities within the blocks.
2. ***Run-Length Encoding:*** Scan the image horizontally or vertically and while scanning assign a group of pixel with the same intensity into a pair (gi ,li) where gi is the intensity and li is the length of the "run". This method can also be used for detecting edges and boundaries of an object. It is mostly used for images with a small number of gray levels and is not effective for highly textured images.

Example 1: Consider the following 8×8 image.

4	4	4	4	4	4	4	0
4	5	5	5	5	5	4	0
4	5	6	6	6	5	4	0
4	5	6	7	6	5	4	0
4	5	6	6	6	5	4	0
4	5	5	5	5	5	4	0
4	4	4	4	4	4	4	0
4	4	4	4	4	4	4	0

The run-length codes using vertical (continuous top-down) scanning mode are:

(4,9)	(5,5)	(4,3)	(5,1)	(6,3)
(5,1)	(4,3)	(5,1)	(6,1)	(7,1)
(6,1)	(5,1)	(4,3)	(5,1)	(6,3)
(5,1)	(4,3)	(5,5)	(4,10)	(0,8)

i.e. total of 20 pairs $= 40$ numbers. The horizontal scanning would lead to 34 pairs $= 68$ numbers, which is more than the actual number of pixels (i.e. 64).

Example 2: Let the transition probabilities for run-length encoding of a binary image (0:black and 1:white) be $p_0 = P(0/1)$ and $p1 = P(1/0)$. Assuming all runs are independent, find (a) average run lengths, (b) entropies ofwhite and black runs, and (c) compression ratio.

Solution:

A run of length $l \geq 1$ can be represented by a Geometric random variable (Grv) X_i with PMF $P(X_i = l) = p_i (1\text{-}p_i)^{t-1}$ with i = 0,1 which corresponds to happening of 1st occurrences of 0 or 1 after l independent trials. (Note that (1-P(0/1)) = P(1/1) and(1-P(1/0)) = P(0/0)) and Thus, for the average we have

$$\mu_{X_i} = \sum_{l=1}^{\infty} lP(X_i = l) = \sum_{l=1}^{\infty} lp_i(1-p_i)^{l-1}$$

which using series $\sum_{n=1}^{\infty} na^{n-1} = \frac{1}{(1-a)^2}$ reduces to $\mu_{X_i} = \frac{1}{p_i}$. The entropy is given by

$$\begin{aligned} H_{X_i} &= -\sum_{l=1}^{\infty} P(X_i = l) log_2 P(X_i = l) \\ &= -p_i \sum_{l=1}^{\infty} (1-p_i)^{l-1} [log_2 p_i + (l-1) log_2 (1-p_i)] \end{aligned}$$

Using the same series formula, we get

$$H_{X_i} = -\frac{1}{p_i}[p_i log_2 p_i + (1-p_i) log_2 (1-p_i)]$$

. The achievable compression ratio is

$$C = \frac{H_{X_0} + H_{X_1}}{\mu_{X_0} + \mu_{X_1}} = \frac{H_{X_0} P_0}{\mu_{X_0}} + \frac{H_{X_1} P_1}{\mu_{X_1}}$$

where $P_i = \frac{p_i}{p_0 + p_i}$ are the *a priori* probabilities of black and white pixels.

Huffman Encoding Algorithm*:* It consists of the following steps.

1. Arrange symbols with probability Pk's in a decreasing order and consider them as "leaf nodes" of atree.
2. Merge two nodes with smallest prob to form a new node whose prob is the sum of the two merged nodes. Go to Step 1 and repeat until only two nodes are left ("root nodes"). 3 Arbitrarily assign 1's and 0's to each pair of branches merging into a node. 4 Read sequentially from root node to the leaf nodes to form the associated code for eachsymbol.

Example 3: For the same image in the previous example, which requires 3 bits/pixel using standard PCM we can arrange the table .

Gray levels	# occurrences	P_k	C_k	β_k	$P_k\beta_k$	$-P_k \log_2 P_k$
0	8	0.125	0000	4	0.5	0.375
1	0	0	-	0	-	-
2	0	0	-	0	-	-
3	0	0	-	0	-	-
4	31	0.484	1	1	0.484	0.507
5	16	0.25	01	2	0.5	0.5
6	8	0.125	001	3	0.375	0.375
7	1	0.016	0001	4	0.64	0.095
	64	1			R	H

Codewords C_ks are obtained by constructing the binary tree as in Fig. 5.

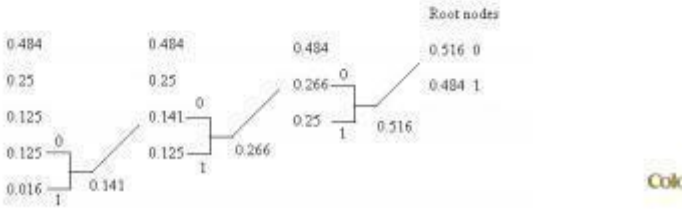


Fig. Tree structure for Huffman Encoding

Note that in this case, we have

$$R = \sum_{k=1}^{8} \beta_k P_k = 1.923 \text{ bits/pixel}$$

$$H = -\sum_{k=1}^{8} P_k \log_2 P_k = 1.852 \text{ bits/pixel}$$

Thus,

$$1.852 \leq R = 1.923 \leq H + \frac{1}{L} = 1.977$$

i.e., an average of 2 bits/pixel (instead of 3 bits/pixel using PCM) can be used to code the image. However, the drawback of the standard Huffman encoding method is that the codes have variable lengths.

PREDICTIVE ENCODING:

Idea: Remove mutual redundancy among successive pixels in a region of support (ROS) or neighborhood and encode only the new information. This mehtod is based upon linear prediction. Let us start with 1-D linear predictors. An N^{th} order linear prediction of x(n) based on N previous samples is generated using a 1-D autoregressive (AR)model.

$$\hat{x}(n) = a_1 x(n-1) + a_2 x(n-2) + \cdots + a_N x(n-N)$$

a_is are model coefficients determined based on some sample signals. Now instead of encoding x(n) the prediction error.

$$e(n) = x(n) - \hat{x}(n)$$

Is encoded as it requires substantially small number of bits. Then, at the receiver we reconstruct x(n) using the previous encoded values x(n-k) and the encoded error signal, i.e.,

$$x(n) = \hat{x}(n) + e(n)$$

This method is also referred to as differential PCM (DPCM).

Minimum Variance Prediction

The predictor

$$\hat{x}(n) = \sum_{i=1}^{N} a_i x(n-i)$$

is the best N^{th} order linear mean-squared predictor of $x(n)$, which minimizes the MSE

$$\epsilon = E\left[\left(x(n) - \hat{x}(n)\right)^2\right]$$

This minimization wrt a_k's results in the following "orthogonal property"

$$\frac{\partial \epsilon}{\partial a_k} = -2E\left[\left(x(n) - \hat{x}(n)\right) x(n-k)\right] = 0, \qquad 1 \le k \le N$$

which leads to the normal equation

$$r_{xx}(k) - \sum_{i=1}^{N} a_i r_{xx}(k-i) = \sigma_e^2 \delta(k), \qquad 0 \le k \le N$$

where $r_{xx}(k)$ is the autocorrelation of the data $x(n)$ and σ_e^2 is the variance of the driving process $e(n)$.

To understand the need for compact image representation, consider the amount of data required to represent a **2** hour standard Definition (SD) using 720 x 480 x 24 bit pixel arrays.

A video is a sequence of video frames where each frame is full color still image.

Because video player must display the frames sequentially at rates near 30 fps. Standard definition data must be accessed 30fps x (720 x 480) ppf x 3bpp = 31,104,000 bps.

fps: frames per second, ***ppf***: pixels per frame, ***bpp***: bytes per pixel, ***bps***: bytes per second.

Thus a 2 hour movie consists of : = 31,104,000 bps x (60^2) sph x 2hrs
where sph is second per hour = 2.24 x 10^{11} bytes = 224 GB of data.
TWENTY SEVEN 8.5 GB dual layer DVD's are needed to store it.
To put 2 hours movie on a single DVD, each frame must be compressed by a factor of around 26.3.

The compression must be even higher for HD, where image resolution reaches 1920 x 1080 x 24 bits per image.

Webpage images & High-resolution digital camera photos also are compressed to save storage space & reduce transmission time.

Residential Internet connection delivers data at speeds ranging from 56kbps (conventional phone line) to more than 12 mbps (broadband).

Time required to transmit a small 128 x 128 x 24 bit full color image over this range of speed is from 7.0 to 0.03 sec.

Compression can reduce the transmission time by a factor of around 2 to 10 or more.

Similarly, number of uncompressed full color images that an 8 Megapixel digital camera can store on a 1GB Memory card can be increased.

Data compression: It refers to the process of reducing the amount of data required to represent a given quantity of information.

Data Vs Information:

Data and information is not the same thing; data are the means by which information is conveyed.

Because various amounts of data can be used to represent the same amount of information, representations that contain irrelevant or repeated information are said to contain redundant.

In today's multimedia wireless communication, major issue is bandwidth needed to satisfy real time transmission of image data. Compression is one of the good solutions to address this issue.

Transform based compression algorithms are widely used in the field of compression, because of their de-correlation and other properties, useful in compression. In this paper, comparative study of compression methods is done based on their types. This paper addresses the issue of importance of transform in image compression and selecting particular transform for image compression. A comparative study of performance of a variety of different image transforms is done base on compression ratio, entropy and time factor.

THE FLOW OF IMAGE COMPRESSION CODING:

Image compression coding is to store the image into bit-stream as compact as possible and to display the decoded image in the monitor as exact as possible. Now consider an encoder and a decoder as shown in Fig. 1.3. When the encoder receives the original image file, the image file will be converted into a series of binary data, which is called the bit-stream. The decoder then receives the encoded bit-stream and decodes it to form the decoded image. If the total data quantity of the bit-stream is less than the total data quantity of the original image, then this is called image compression. The full compression flow is as shown in Fig.1.3.

Fig. 1.3 The basic flow of image compression coding

The compression ratio is defined as follows:

$$Cr = \frac{n1}{n2}, \tag{1.2}$$

where $n1$ is the data rate of original image and $n2$ is that of the encoded bit-stream.

In order to evaluate the performance of the image compression coding, it is necessary to define a measurement that can estimate the difference between the original image and the decoded image. Two common used measurements are the Mean Square Error (MSE) and the Peak Signal to Noise Ratio (PSNR), which are defined in (1.3) and (1.4), respectively. f(x,y) is the pixel value of the original image, and f'(x,y)is the pixel value of the decoded image. Most image compression systems are designed to minimize the MSE and maximize the PSNR.

$$MSE = \sqrt{\frac{\sum_{x=0}^{W-1}\sum_{y=0}^{H-1}[f(x,y)-f'(x,y)]^2}{WH}} \tag{1.3}$$

$$PSNR = 20\log_{10}\frac{255}{MSE} \tag{1.4}$$

The general encoding architecture of image compression system is shown is Fig. 1.4. The fundamental theory and concept of each functional block will be introduced in the following sections.

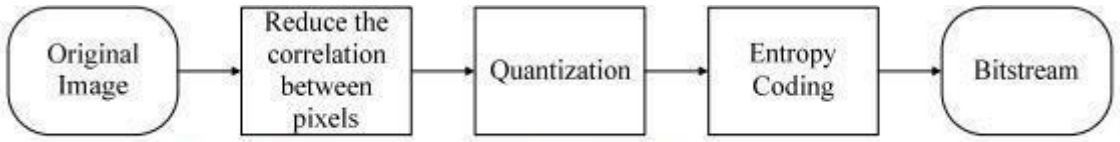


Fig. 1.4 The general encoding flow of image compression

Reduce the Correlation between Pixels

The reason is that the correlation between one pixel and its neighbor pixels is very high, or we can say that the values of one pixel and its adjacent pixels are very similar. Once the correlation between the pixels is reduced, we can take advantage of the statistical characteristics and the variable length coding theory to reduce the storage quantity. This is the most important part of the image compression algorithm; there are a lot of relevant processing methods being proposed. The best-known methods are as follows:

- Predictive Coding: Predictive Coding such as DPCM (Differential Pulse Code Modulation) is a lossless coding method, which means that the decoded image and the original image have the same value for every corresponding element.
- Orthogonal Transform: Karhunen-Loeve Transform (KLT) and Discrete Cosine Transform (DCT) are the two most well-known orthogonal transforms. The DCT- based image compression standard such as JPEG is a lossy coding method that will result in some loss of details and unrecoverable distortion.
- Subband Coding: Subband Coding such as Discrete Wavelet Transform (DWT) is also a lossy coding method. The objective of subband coding is to divide the spectrum of one image into the low pass and the high pass components. JPEG 2000 is a 2- dimension DWT based image compression standard.

QUANTIZATION

The objective of quantization is to reduce the precision and to achieve higher compression ratio. For instance, the original image uses 8 bits to store one element for every pixel; if we use less bits such as 6 bits to save the information of the image, then the storage quantity will be reduced, and the image can be compressed. The shortcoming of quantization is that it is a lossy operation, which will result into loss of precision and unrecoverable distortion. The image compression standards such as JPEG

and JPEG 2000 have their own quantization methods, and the details of relevant theory will be introduced in the chapter 2.

ENTROPY CODING

The main objective of entropy coding is to achieve less average length of the image. Entropy coding assigns codewords to the corresponding symbols according to the probability of the symbols. In general, the entropy encoders are used to compress the data by replacing symbols represented by equal-length codes with the code words whose length is inverse proportional to corresponding probability. The entropy encoder of JPEG and JPEG 2000 will also be introduced.

IMAGE COMPRESSION STANDARD:

In this chapter, we will introduce the fundamental theory of two well-known image compression standards –JPEG and JPEG 2000.

JPEG – JOINT PICTURE EXPERT GROUP

Fig. 2.1 and 2.2 shows the Encoder and Decoder model of JPEG. We will introduce the operation and fundamental theory of each block in the following sections.

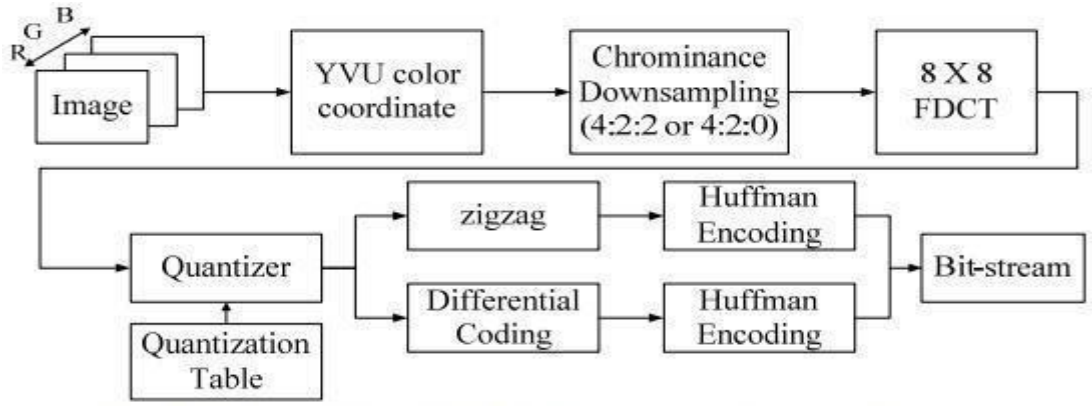


Fig. 2.1 The Encoder model of JPEG compression standard

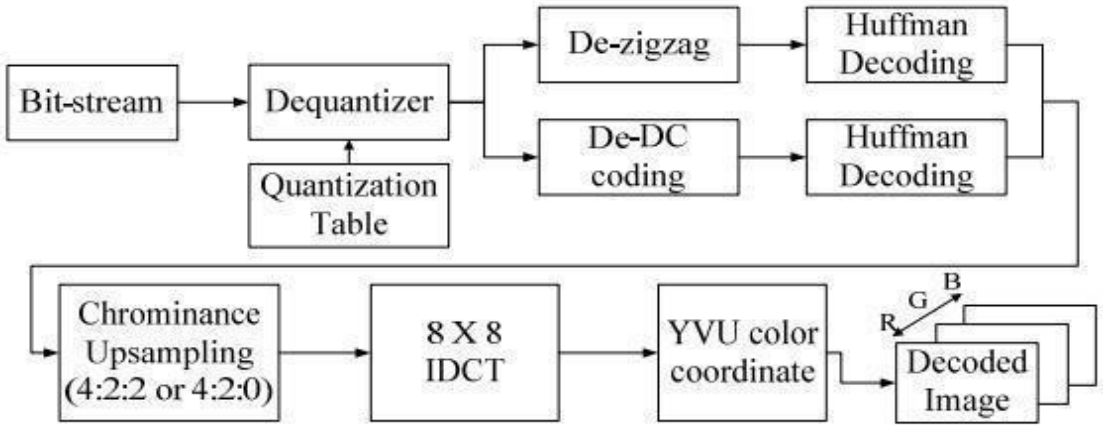


Fig. 2.2 The Decoder model of JPEG compression standard

DISCRETE COSINE TRANSFORM

The next step after color coordinate conversion is to divide the three color components of the image into many 8×8 blocks. The mathematical definition of the Forward DCT and the Inverse DCT are as follows:

Forward DCT

$$F(u,v)=\frac{2}{N}C(u)C(v)\sum_{x=0}^{N-1}\sum_{y=0}^{N-1}f(x,y)\cos\left[\frac{\pi(2x+1)u}{2N}\right]\cos\left[\frac{\pi(2y+1)v}{2N}\right]$$

for $u=0,\ldots,N-1$ and $v=0,\ldots,N-1$ **(2.1)**

where $N=8$ and $C(k)=\begin{cases}1/\sqrt{2} & \text{for } k=0\\ 1 & \text{otherwise}\end{cases}$

Inverse DCT

$$f(x,y)=\frac{2}{N}\sum_{u=0}^{N-1}\sum_{v=0}^{N-1}C(u)C(v)F(u,v)\cos\left[\frac{\pi(2x+1)u}{2N}\right]\cos\left[\frac{\pi(2y+1)v}{2N}\right] \quad \textbf{(2.2)}$$

for $x=0,\ldots,N-1$ and $y=0,\ldots,N-1$ where $N=8$

The f(x,y) is the value of each pixel in the selected 8×8 block, and the F(u,v) is the DCT coefficient after transformation. The transformation of the 8×8 block is also a 8×8 block composed of F(u,v).

The DCT is closely related to the DFT. Both of them taking a set of points from the spatial domain and transform them into an equivalent representation in the frequency domain. However, why DCT is more appropriate for image compression than DFT. The two main reasons are:

1. The DCT can concentrate the energy of the transformed signal in low frequency, whereas the DFT can not. According to Parseval's theorem, the

energy is the same in the spatial domain and in the frequency domain. Because the human eyes are less sensitive to the low frequency component, we can focus on the low frequency component and reduce the contribution of the high frequency component after taking DCT.

2. For image compression, the DCT can reduce the blocking effect than the DFT.

After transformation, the element in the upper most left corresponding to zero frequency in both directions is the "DC coefficient" and the rest are called "AC coefficients."

Quantization in JPEG:

Quantization is the step where we actually throw away data. The DCT is a lossless procedure. The data can be precisely recovered through the IDCT (this isn't entirely true because in reality no physical implementation can compute with perfect accuracy). During Quantization every coefficients in the 8×8 DCT matrix is divided by a corresponding quantization value. The quantized coefficient is defined in (2.3), and the reverse the process can be achieved by the (2.4).

$$F(u,v)_{Quantization} = round\left(\frac{F(u,v)}{Q(u,v)}\right) \qquad (2.3)$$

$$F(u,v)_{deQ} = F(u,v)_{Quantization} \times Q(u,v) \qquad (2.4)$$

The goal of quantization is to reduce most of the less important high frequency DCT coefficients to zero, the more zeros we generate the better the image will compress. The matrix Q generally has lower numbers in the upper left direction and large numbers in the lower right direction. Though the high-frequency components are removed, the IDCT still can obtain an approximate matrix which is close to the original 8×8 block matrix. The JPEG committee has recommended certain Q matrix that work well and the performance is close to the optimal

$$Q_Y = \begin{pmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{pmatrix} \qquad (2.5)$$

condition, the Q matrix for luminance and chrominance components is defined in (2.5) and(2.6)

$$Q_C = \begin{pmatrix} 17 & 18 & 24 & 47 & 99 & 99 & 99 & 99 \\ 18 & 21 & 26 & 66 & 99 & 99 & 99 & 99 \\ 24 & 26 & 56 & 99 & 99 & 99 & 99 & 99 \\ 47 & 66 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \end{pmatrix} \quad \textbf{(2.6)}$$

ZIGZAG SCAN:

After quantization, the DC coefficient is treated separately from the 63 AC coefficients. The DC coefficient is a measure of the average value of the original 64 image samples. Because there is usually strong correlation between the DC coefficients of adjacent 8×8 blocks, the quantized DC coefficient is encoded as the difference from the DC term of the previous block. This special treatment is worthwhile, as DC coefficients frequently contain a significant fraction of the total image energy. The other 63 entries are the AC components. They are treated separately from the DC coefficients in the entropy coding process.

0	1	5	6	14	15	27	28
2	4	7	13	16	26	29	42
3	8	12	17	25	30	41	43
9	11	18	24	31	40	44	53
10	19	23	32	39	45	52	54
20	22	33	38	46	51	55	60
21	34	37	47	50	56	59	61
35	36	48	49	57	58	62	63

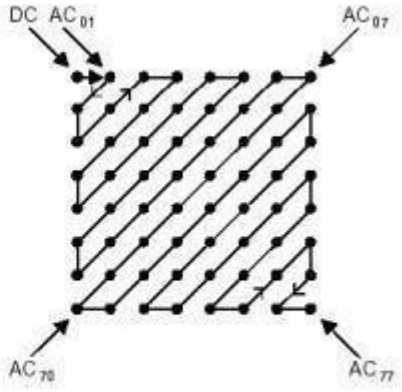
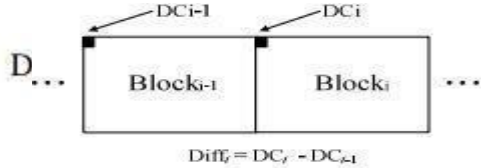


Fig. 2.3 The zigzag scan order

Entropy Coding in JPEG Differential Coding:

The mathematical representation of the differential coding is:



(2.7)

Fig. 2.4 Differential Coding

We set DC0 = 0. DC of the current block DCi will be equal to DCi-1 + Diffi . Therefore, in the JPEG file, the first coefficient is actually the difference of DCs. Then the difference is encoded

with Huffman coding algorithm together with the encoding of AC coefficients.