University of Technology
الجامعة التكنولوجية

Computer Science Department
قسم علوم الحاسوب

# Image Processing 2
# معالجة الصور ٢

Prof. Dr. Matheel Imad Eldeen
أ.د. مثيل عماد الدين

cs.uotechnology.edu.iq

# LECTURE 1

## FEATURE EXTRACTION AND ANALYSIS

The goal in image analysis is to extract information useful for solving **application** based problems. **The first step** to this by reducing the amount of image data with the tools we have explored. **The next step** would be to extract features that are useful in solving computer imaging problems. After we have extracted the features of interest, we can analyze the image

**Feature extraction** is part of the data reduction process and is followed by feature analysis. One of the important aspects of feature analysis is to determine exactly which features are important, so the analysis is not complete until we incorporate application-specific feedback into the system.

### Feature Vectors and Feature Spaces

**A feature vector** is one method to represent an image, or part of an image (an object), by finding measurements on a set of features. A feature vector is an *n*-dimensional vector that contains a set of values where each value represents a certain feature. This vector can be used to classify an object, or provide us with condensed higher-level information regarding the image.

### Let us consider one example:

We need to control a robotic gripper that picks parts from an assembly line and puts them into boxes (either box A or box B, depending on object type). In order to do this, we need to determine:

1) Where the object is 2) What type of object it is

The first step would be to define the feature vector that will solve this problem.

**To determine where the object is:**

Use the area and the center area of the object, defined by (r,c).

**To determine the type of object:**

Use the perimeter of object.

Therefore, the feature vector is: [area, r, c, and perimeter]

## Distance & Similarity Measures

In feature extraction process, we might need to compare two feature vectors. The primary methods to do this are either to measure the **difference** between the two or to measure the **similarity.** The difference can be measured using a ***distance measure*** in the n-dimensional space. The bigger the distance between two vectors, the greater the difference.

### a. Distance Measures

There are several metric measurements: ***Euclidean distance, Range-normalized Euclidean distance, City block*** **or** ***absolute value metric, maximum value***

- ***Euclidean distance*** is the most common metric for measuring the distance between two vectors.

  Given two vectors **A** and **B**, where:

$$\begin{aligned} A &= \begin{bmatrix} a_1 & a_2 & \dots & a_n \end{bmatrix} \\ B &= \begin{bmatrix} b_1 & b_2 & \dots & b_n \end{bmatrix} \end{aligned}$$

  The Euclidean distance is given by:

$$\sqrt{\sum_{i=1}^{n}(a_i-b_i)^2}=\sqrt{(a_1-b_1)^2+(a_2-b_2)^2+\dots+(a_n-b_n)^2}$$

  This measure may be biased as a result of the varying range on different components of the vector. For example, one component may only range from 1 to 5 and another may range from 1 to 5000, so a difference of 5 for the first component will be maximum, but a difference of 5 for the second feature may be insignificant.

- We can alleviate this problem by using the **range-normalized Euclidean distance,**

$$\sqrt{\sum_{i=1}^{n} \frac{(a_i - b_i)^2}{R_i^{\,2}}}$$

$R_i$ is the range of the

$i$th component.

- Another distance measure, called the ***city block*** **or** ***absolute value metric***, is defined as follows:

$$\sum_{i=1}^{n} |a_i - b_i|$$

This metric is computationally **faster than the Euclidean distance** but **gives similar result.**

- The final distance metric considered here is the ***maximum value*** metric defined by:

$$\max\{|a_1 - b_1|, |a_2 - b_2|, \ldots, |a_n - b_n|\}$$

## b. Similarity Measures

The second type of metric used for comparing two feature vectors is the ***similarity measure***. The most common form of the similarity measure is the vector **inner product**. Using our definition of vector **A** and **B**, the vector inner product can be defined by the following equation:

$$\sum_{i=1}^{n} a_i b_i = (a_1 b_1 + a_2 b_2 + \ldots + a_n b_n)$$

When selecting a feature for use in a computer imaging application, an important factor is the **robustness** of the feature. A feature is robust if it will provide consistent results across the entire application domain. For example, if we develop a system to work under any lightning conditions, we do not want to use features that are lightning dependent.

## Binary Object Features

In order to extract object features, we need an image that has undergone image segmentation and any necessary morphological filtering. This will provide us with a clearly defined object which can be labeled and processed independently.

After all the binary objects in the image are labeled, we can treat each object as a binary image. The labeled object has a value of '1' and everything else is '0'. The labeling process goes as follows:

- Define the desired connectivity.
- Scan the image and label connected objects with the same symbol.

After we have labeled the objects, we have an image filled with object numbers. This image is used to extract the features of interest. Among the binary object features include

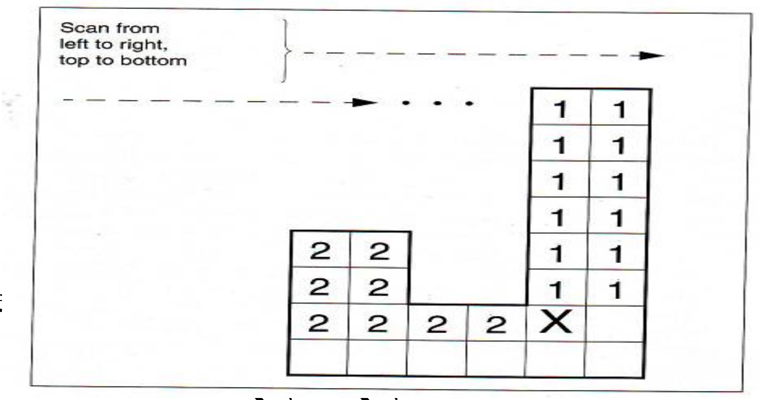
1. area,
2. center of area,
3. axis of least second moment,
4. perimeter,
5. Euler number,
6. projections,
7. thinness ration and
8. Aspect ratio.

The first four tell us something about **where the object is**, and the latter four tell us something about **the shape of the object**.

In order to provide general equations for area, center of area, and axis of least second moment, we define a function, I,(r, c):

$$I_i(r, c) = \begin{cases} 1 & \text{if } I(r, c) = i\text{th } \textit{object number} \\ 0 & \textit{otherwise} \end{cases}$$

Multiple labeling can occur during sequential scanning, as shown above on the J shaped object. We label two different objects until we reach the pixel marked X, where we discover that objects 1 and 2 are connected. As show in the figure below



1. **Area**

   The area of the

$$A_i = \sum_{r=0} \sum_{c=0} I_i(r,c)$$

   The area A$_i$ is measured in pixels and indicates the relative size of the object.

2. **Center of Area**

   The center of area is defined as follows:

$$\overline{r}_i = \frac{1}{A_i} \sum_{r=0}^{height-1} \sum_{c=0}^{width-1} rI_i(r,c)$$

$$\overline{c}_i = \frac{1}{A_i} \sum_{r=0}^{height-1} \sum_{c=0}^{width-1} cI_i(r,c)$$

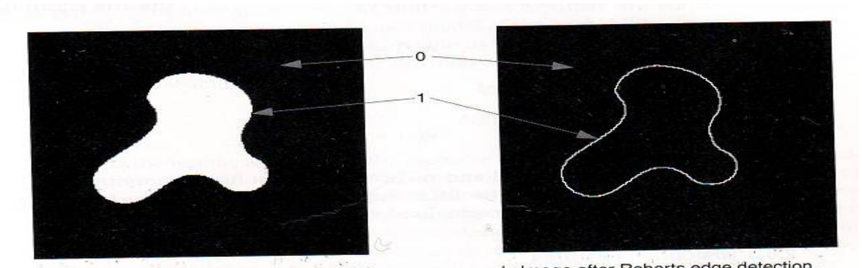
These correspond to the row and column coordinates of the center of the *i*th object

3. **Axis of Least Second Moment**

   The Axis of Least Second Moment is expressed as θ - the angle of the axis relatives to the vertical axis.

4. **Perimeter**

   The perimeter is defined as the total pixels that constitute the edge of the object. Perimeters can be found by counting the number of '1' pixels that have '0' pixels as neighbors. Perimeter can also be found by applying an edge detector to the object, followed by counting the '1' pixels.



a. Image with binary object. We can find the perimeter by counting the '1' pixels that have a '0' neighbor.

b. Image after Roberts edge detection. We find the perimeter by counting the '1' pixels.

5. **Thinness Ratio**

   The thinness ratio, *T*, can be calculated from perimeter and area.

6. **Aspect Ratio**

   This can be found by scanning the image and finding the minimum and maximum values on the row and column where the object lies.

   The equation for aspect ratio is as follows:

$$\frac{c_{\max} - c_{\min} + 1}{r_{\max} - r_{\min} + 1}$$

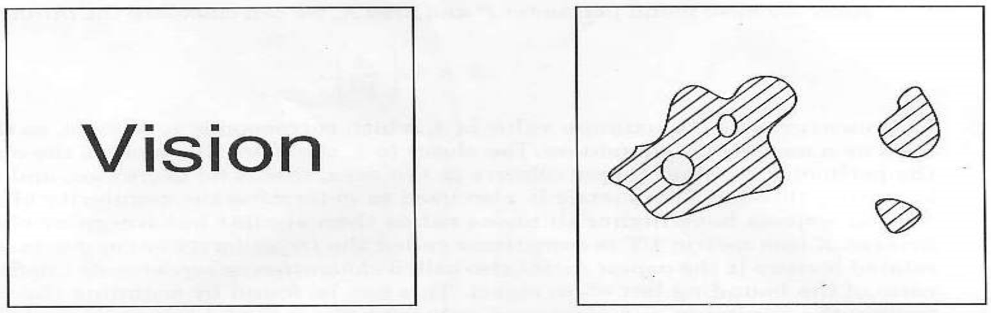
High aspect ratio indicates the object spread more towards **horizontal direction.**

7. **Euler Number**

Euler number is defined as the difference between the number of objects and the number of holes.

**Euler number = num of object – number of holes**

In the case of a single object, the Euler number indicates how many closed curves (holes) the object contains. Euler number can be used in tasks such as optical character recognition (OCR).



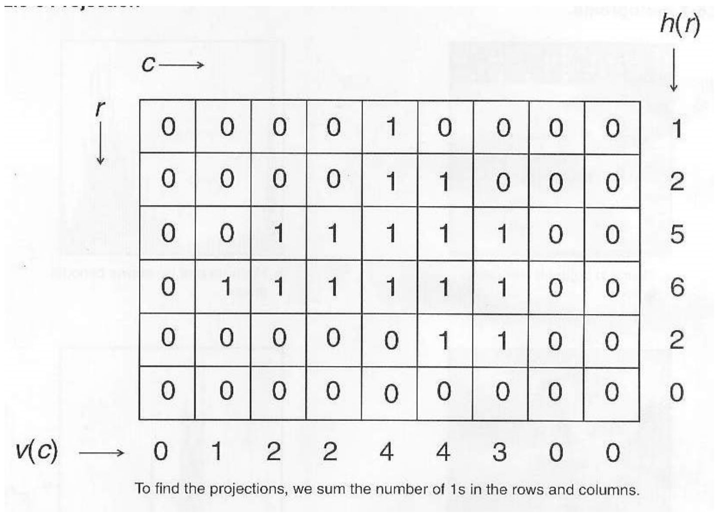
a. This image has eight objects and one hole, so its Euler number is 8 – 1 = 7. The letter *V* has Euler number of 1, *i* = 2, *s* = 1, *o* = 0, and *n* = 1.

b. This image has three objects and two holes, so the Euler number is 3 – 2 = 1.

8. **Projection**

The projection of a binary object, may provide useful information related to object's shape. It can be found by summing all the pixels along the rows or columns.

- Summing the rows give horizontal projection.
- Summing the columns give the vertical projection.



$$\begin{array}{c|ccccccccc|c}
 & c \longrightarrow & & & & & & & & & h(r) \downarrow \\
r \downarrow & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\
 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 2 \\
 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 5 \\
 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 6 \\
 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 2 \\
 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\hline
v(c) \longrightarrow & 0 & 1 & 2 & 2 & 4 & 4 & 3 & 0 & 0 &
\end{array}$$

To find the projections, we sum the number of 1s in the rows and columns.

# LECTURE 2

## Histogram features

The histogram features that we are considered are statically based features where the histogram is used as a model of the probability distribution of the gray levels. These statistical features provide us with information about the characteristic of the gray – level distribution for the image or sub image. We define the **first–order histogram probability P(a)** as :

$$P(g) = \frac{N(g)}{M}$$

M is the number of pixels in the image or sub image (if the entire image is under consideration, then M= $N^2$ for N×N), and N (g) is the number of pixels at gray level g. As with any probability distribution, all values for P (g) are less than or equal to 1. Histogram features are mean, standard deviation, skew, energy and entropy.

1. **Mean:** the mean is the average value, so it tells us something about the general brightness of the image. A bright image will have a high mean, and a dark image will have a low mean. We will use L as the total number of gray levels available, so the gray levels range from 0 to L-1. For example, for typical 8-bit image data, L is 256 and ranges from 0 to 255. We can define the mean as follows:

$$\overline{g} = \sum_{g=0}^{L-1} gP(g) = \sum_{r}\sum_{c}\frac{I(r,c)}{M}$$

If we use the second form of the equation, we sum over the rows and columns corresponding to the pixels in the image or sub image under consideration.

2. **Standard deviation:**Which is also known as the square root of the **variance**, tell us something about the contrast. It describes the spread in the data, so a high contrast image will have a high variance, and a low – contrast image will have a low variance. It is defined as follows:

$$\sigma = \sqrt{\sum_{g=0}^{L-1} (g-\bar{g})^2 P(g)}$$

3. **Skew** :the skew measure the asymmetry about the mean in the gray-level distribution .it is defined as:

$$\text{SKEW} = \frac{1}{\sigma_g^3} \sum_{g=0}^{L-1} (g-\bar{g})^3 P(g)$$

This method of measuring skew is more computationally efficient, especially considering that, typically, the mean and standard deviation have already been calculated.

4. The **energy** measure tell us something about how the gray level are distributed

$$\text{Energy} = \sum_{g=0}^{L-1} [P(g)]^2$$

The energy measure has a maximum value of 1 for an image with a constant value and gets increasingly smaller as the pixel values are distributed across more gray level values(remember that all the P(g) values are less than or equal to 1). The lager this value is, the easier it is to compress the image data. If the energy is high, it tells us that the number of gray levels in the image is few, that is, the distribution is concentrated in only a small number of different gray levels.

**5. Entropy:** the entropy is a measure that tells us how many bits we need to code the image data and given by :

$$Entropy = -\sum_{g=0}^{L-1} P(g)\log_2[P(g)]$$

As the pixel values in the image are distributed among more gray levels, the entropy increases. This measure tends to vary inversely with the energy. Note (log2=log10 × 3.33).

# 3. Segmentation

Image segmentation is important in many computer visions and image processing application .The goal of image segmentation is to find regions that represent objects or meaningful parts of objects.

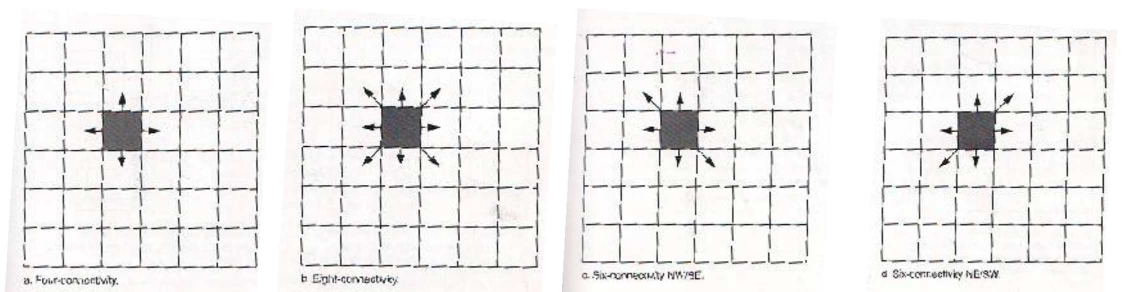
Division of the image into regions corresponding to objects of interest is necessary before any processing can be done at a level higher than that the pixel. Identifying real objects, pseudo—objects, and shadows or actually finding anything of interest within the image requires some form of segmentation.

## Image Segmentation Methods

Image segmentation methods will look for objects that either have some measure of homogeneity within themselves or have some measure of contrast with the objects on their border. Most image segmentation algorithms are modifications, extensions, or combinations of these two basic concepts. The homogeneity and contrast measures can include features such as gray level, color, and texture. After we have performed some preliminary segmentation, we may incorporate higher-level object properties, such as perimeter and shape, into the segmentation process.

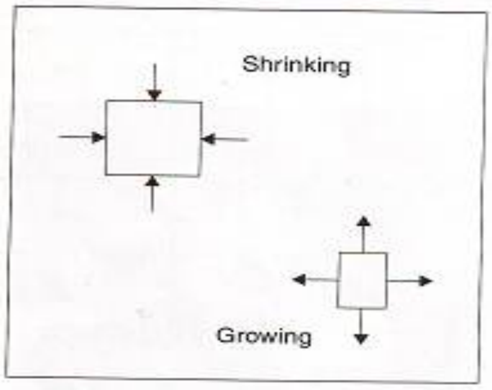
Before we look at the different segmentation methods, we need to consider some of the problems associated with image segmentation. The major problems are a result of noise in the image and digitization of a continuous image. Noise is typically caused by the camera, the lenses, the lighting, or the signal path and can be reduced by the use of the preprocessing methods previously discussed. Spatial digitization can cause problems regarding connectivity of objects. These problems can be resolved with careful connectivity definitions and heuristics applicable to the specific domain.

Connectivity refers to the way in which we define an object. After we have segmented an image, which segments should be connected to form an object? Or, at a lower level, when searching the image for homogeneous regions, how do we define which pixels are connected? We must define which of the surrounding pixels are considered to be neighboring pixels. A pixel has eight possible neighbors: two horizontal neighbors, two vertical neighbors, and four diagonal neighbors. We can define connectivity in three different ways: 1) four-connectivity, 2) eight-connectivity, and 3) six-connectivity. Figure below illustrates these three definitions.

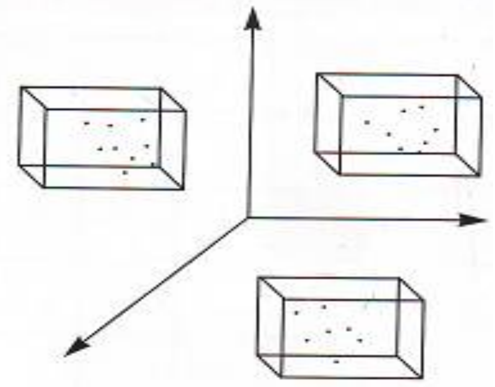


a. Four-connectivity. b Eight-connectivity. c. Six-connectivity NW/SE. d Six-connectivity NE/SW.

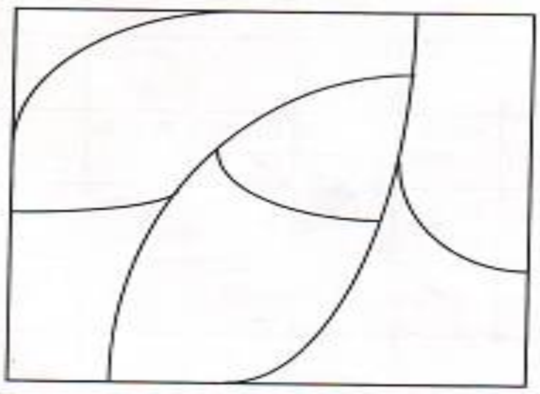
We can divide image segmentation techniques into three main categories (see Figure below): 1) region growing and shrinking, 2) clustering methods, and 3) boundary detection. The region growing and shrinking methods use the row and column (rc) based image space, whereas the clustering techniques can be applied to any domain spatial domain, color space, feature space, etc.



a. Region growing/shrinking is performed by finding homogeneous regions and changing them until they no longer meet the homogeneity criteria.



b. Clustering looks for data that can be grouped in domains other than the spatial domain.



c. Boundary detection is often achieved using a differentiation operator to find lines or edges, followed by postprocessing to connect the points into borders.

## 1.Region Growing and Shrinking

Region growing and shrinking methods segment the image into regions by operating principally in the RC—based image space. Some of the techniques used are local, in which small areas of the image are processed at a time; others are global, with the entire considered during processing. Methods that can combine local and global techniques, such as split and merge, are referred to as state space techniques and use graph structures to represent the regions and their boundaries.

In general, the split and merge technique proceeds as follows:

1. Define a homogeneity test. This involves defining a homogeneity measure, which may incorporate brightness, color, texture, or other application-specific information, and determining a criterion the region must meet to pass the homogeneity test.

2. Split the image into equally sized regions.

3. Calculate the homogeneity measure for each region.

4. If the homogeneity test is passed for a region, then a merge is attempted with its neighbor(s). If the criterion is not met, the region is split.

5. Continue this process until all regions pass the homogeneity test.

There are many variations of this algorithm. For example, we can start out at the global level, where we consider the entire image as our initial region, and then follow an algorithm similar to the preceding algorithm, but without any region merging. Algorithms based on splitting only are called multi resolution algorithms. Alternately we can start at the smallest level and only merge, with no region splitting. This merge-only approach is one example of region growing methods. Often the results from all these approaches will be quite similar, with the differences apparent only in computation time. Parameter choice, such as the minimum block size allowed for splitting, will heavily influence the computational burden as well as the resolution available in the results.

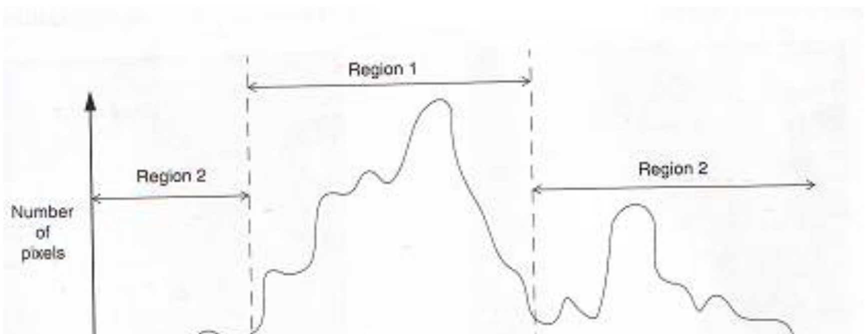
## 2.Clustering Techniques

Clustering techniques are image segmentation methods by which individual elements are placed into groups; these groups are based on some measure of similarity within the group. The major difference between these techniques and the region growing techniques is that domains other than the rc-based image space (the spatial domain) may be considered as the primary domain for clustering. Some of these other domains include color spaces, histogram spaces, or complex feature spaces. The simplest method is to divide the space of interest into regions by selecting the center or median along each dimension and splitting it there; this can be done iteratively until the space is divided into the specific number of regions needed. This method is used in the SCT Center and PCT/Median segmentation algorithms.

The next level of complexity uses an adaptive and intelligent method to decide where to divide the space. These methods include histogram thresholding and other, more complex feature-space-based statistical methods.

**Recursive region splitting** is a clustering method that has become a standard technique. This method uses a thresholding of histograms technique to segment the image. A set of histograms is calculated for a specific set of features, and then each of these histograms is searched for distinct peaks.

**Histogram peak finding**



The best peak is selected and the image is split into regions based on this thresholding of the histogram. One of the first algorithms based on these concepts proceeds as follows:

1. Consider the entire image as one region and compute histograms for each component of interest (for example, red, green, and blue for a color image).

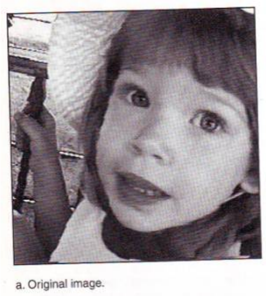
2. Apply a peak finding test to each histogram. Select the best peak and put thresholds on either side of the peak. Segment the image into two regions based on this peak.

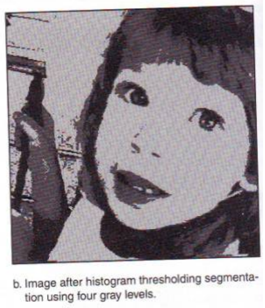
3. Smooth the binary thresholded image so that only a single connected sub region is left.

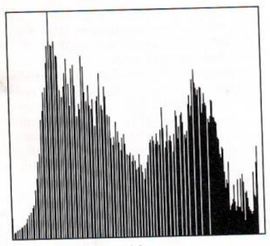
4. Repeat steps 1-3 for each region until no new sub regions can be created, that is, no histograms have significant peaks.

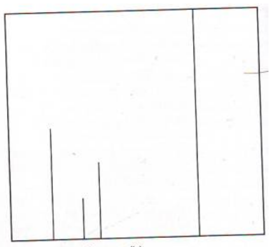
Two thresholds are selected, one on each side of the best peak. The image is then split into two regions. Region 1 corresponds to those pixels with feature values between the selected thresh olds, known as those in the peak. Region 2 consists of those pixels with feature values outside the threshold.

**Histogram Thresholding Segmentation**


a. Original image.


b. Image after histogram thresholding segmentation using four gray levels.

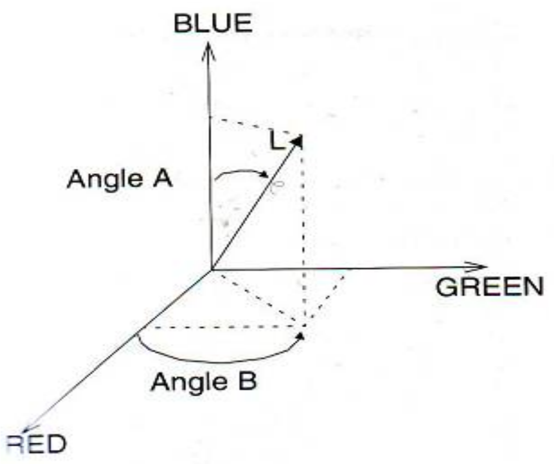

c. Histogram of image (a).


d. Histogram of image (b).

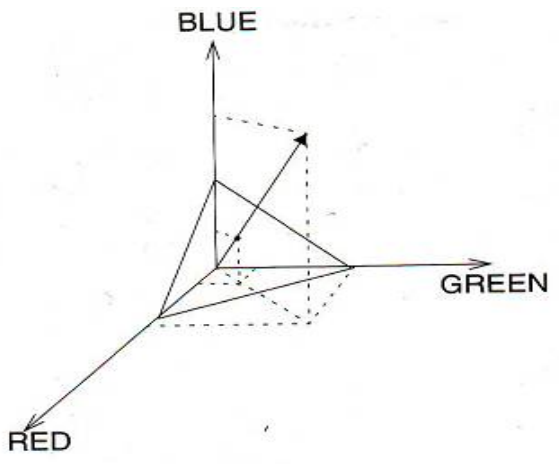
## SCT/Center algorithm:

The SCT/Center algorithm (Spherical Coordinate Transform) by using the two-dimensional color subspace defined by two angles we have a more robust algorithm. If we slice a plane through the RGB color space, we can model a color triangle.

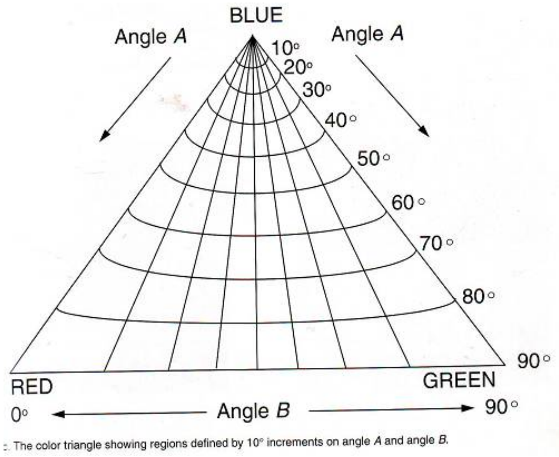
### SCT/ center and color Triangle



a. The spherical coordinate transform separates the red, green, and blue information into a two-dimensional color space defined by angles *A* and *B*, and a one-dimensional brightness space defined by *L*.



b. The color triangle.

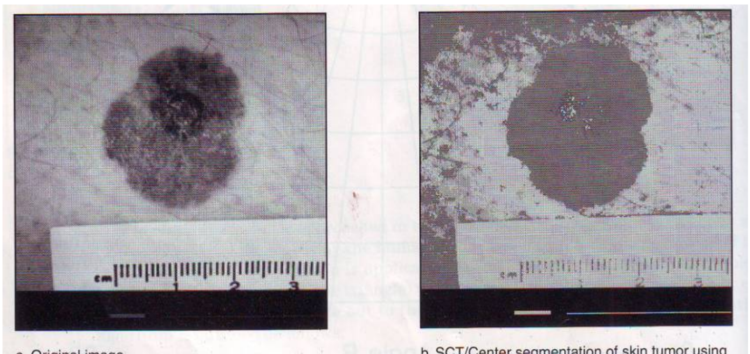


The color triangle showing regions defined by 10° increments on angle *A* and angle *B*.

We can segment the image by taking the color triangle and dividing it into blocks based on limits on the two angles. Figure up shows the shape of the resulting blocks. We can see that for a region defined by a range of minima and maxima on the two angles, the side of the region that is closest to the blue vertex is shorter than the side that is closest to the line that joins the red and green vertices. The SCT/Center segmentation algorithm is outlined as follows:

1. Convert the (R, G, B) triple into spherical coordinates (L, angle A, angle B).
2. Find the minima and maxima of angles A and B.
3. Divide the subspace, defined by the maxima and minima, into equally sized blocks.
4. Calculate the RGB means for the pixel values in each block.
5. Replace the original pixel values with the corresponding RG B means.

For the identification of variegated coloring in the skin tumor application, it was determined that segmenting the image into four colors was optimal. An example of this segmentation method is shown in Figure below.



a. Original image. b. SCT/Center segmentation of skin tumor using four colors.

## PCT/Median color segmentation algorithm:

This algorithm is based around the principal components transform (PCT). The median split part of the algorithm is based on an algorithm developed for color compression to map 24 bits/pixel color images into images requiring an average of 2 bits/pixel.

The PCT is based on statistical properties of the image and can be applied to any K-dimensional mathematical space. In this case, the PCT is applied to the three-dimensional color space. It was believed that the PCT used in conjunction with the median split algorithm would provide satisfactory color image segmentation because the PCT aligns the main axis along the maximum variance path in the data set.

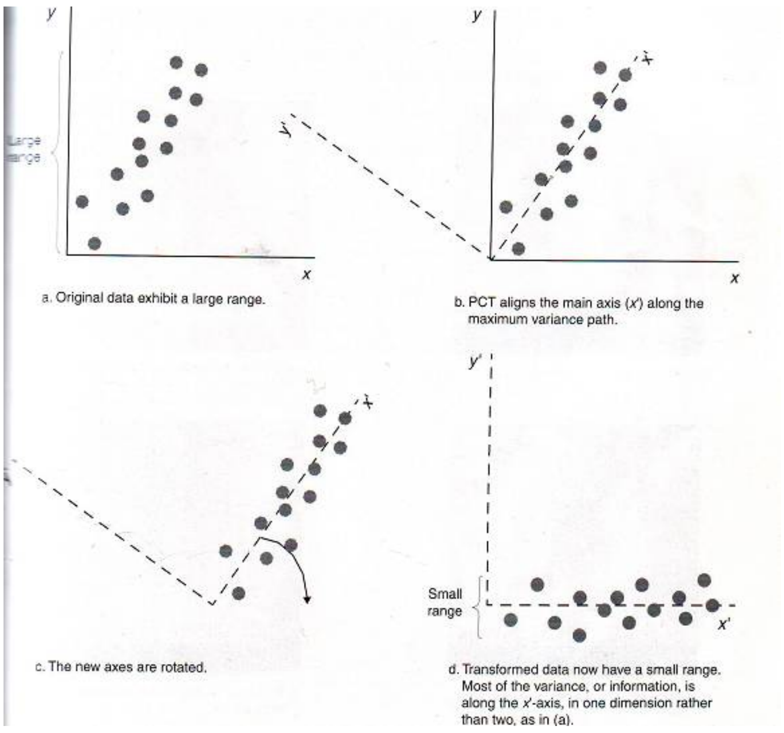
The PCT/Median segmentation algorithm proceeds as follows:

1. Find the PCT for the RGB image. Transform the RGB data using the PCT

2. Perform the median split algorithm: find the axis that has the maximal range (initially it will be the PCT axis). Divide the data along this axis so that there are equal numbers of points on either side of the split the median point. Continue until the desired number of colors is reached.

3. Calculate averages for all the pixels falling within a single box.

4. Map each pixel to the closest average color values, based on a Euclidean distance measure.

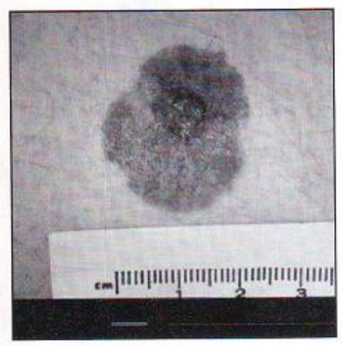


y
Large range
x
a. Original data exhibit a large range.
y
x
b. PCT aligns the main axis (x') along the maximum variance path.
c. The new axes are rotated.
y'
Small range
x'
d. Transformed data now have a small range. Most of the variance, or information, is along the x'-axis, in one dimension rather than two, as in (a).

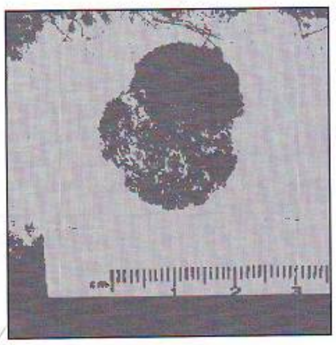
## Principal Components Transform

Results of this segmentation algorithm are shown in Figure below.

It is interesting to note that the PCT is also used in image compression (coding) since this transform is optimal in the least-square-error sense.



a. Original image.



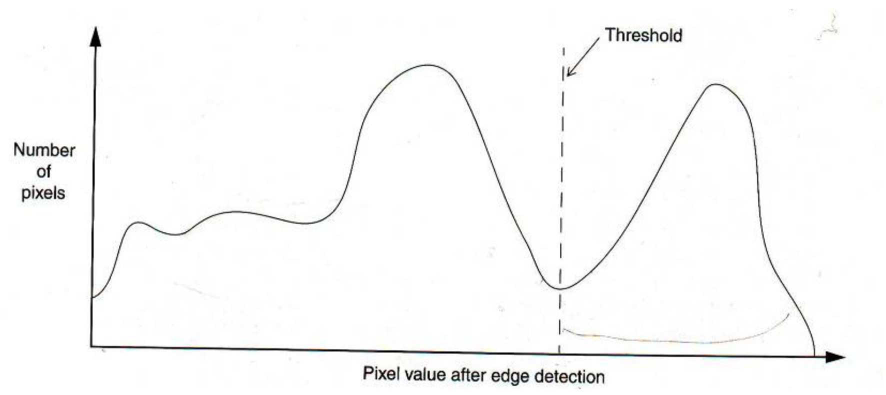
b. PCT/Median segmented image with two colors.

# 4. Segmentation

## 3.Boundary Detection

Boundary detection, as a method of image segmentation, is usually begun by marking points that may be a part of an edge. These points are then merged into line segments, and the line segments are then merged into object boundaries. The edge detectors are used to mark points, thus indicating the possibility of an edge. After the edge detection operation has been performed, the next step is to threshold the results. One method to do this is to consider the histogram of the edge detection results, looking for the best valley.

**Edge Detection Threshold**



Often, the histogram of an image that has been operated on by an edge detector is unimodal (one peak), so it may be difficult to find a good valley. This method works best with a bimodal histogram. After we have determined a threshold for the edge detection, we need to merge the existing edge segments into boundaries. This is done by edge linking.

## Morphological Filtering

Morphology relates to the structure or form of objects. Morphological filtering simplifies a segmented image to facilitate the search for objects of interest. This is done by smoothing out object outlines, filling small holes, eliminating small projections, and using other similar techniques. Even though this section will focus on applications to binary images, the extension of the concepts to gray-level images will also be discussed. We will look at the different types of operations available and at some examples of their use.

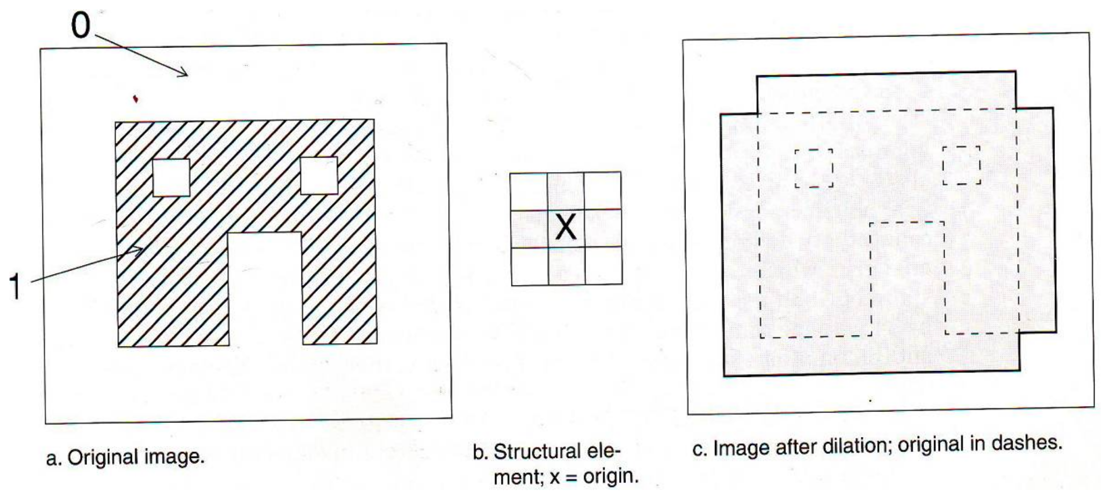
The two principal morphological operations are dilation and erosion. Dilation allows objects to expand, thus potentially filling in small holes and connecting disjoint objects. Erosion shrinks objects by etching away (eroding) their boundaries. These operations can be customized for an application by the proper selection of the structuring element, which determines exactly how the objects will be dilated or eroded. The dilation process is performed by laying the structuring element on the image and sliding it across the image in a manner similar to convolution. The difference is in the operation performed. It is best described in a sequence of steps:

1. If the origin of the structuring element coincides with a '0' in the image, there is no change; move to the next pixel.

2. If the origin of the structuring element coincides with a '1' in the image, perform the OR logic operation on all pixels within the structuring element.

An example is shown in Figure below.

**Dilation**



a. Original image. b. Structural element; x = origin. c. Image after dilation; original in dashes.

Note that with a dilation operation, all the ‘1’ pixels in the original image will be retained, any boundaries will be expanded, and small holes will be filled.

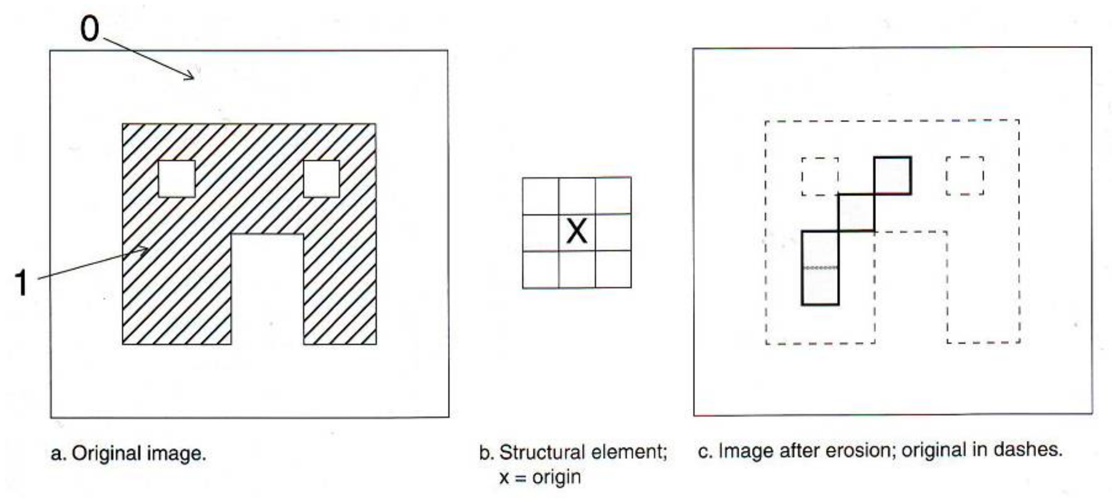
The erosion process is similar to dilation, but we turn pixels to ‘O’, not ‘1’. As before, slide the structuring element across the image and then follow these steps:

1. If the origin of the structuring element coincides with a ‘0’ in the image, there is no change; move to the next pixel.

2. If the origin of the structuring element coincides with a ‘1’ in the image, and any of the ‘1’ pixels in the structuring element extend beyond the object (‘1’ pixels) in the image, then change the ‘1’ pixel in the image to a ‘0’.

In Figure below, the only remaining pixels are those that coincide to the origin of the structuring element where the entire structuring element was contained in the existing object.

## Erosion

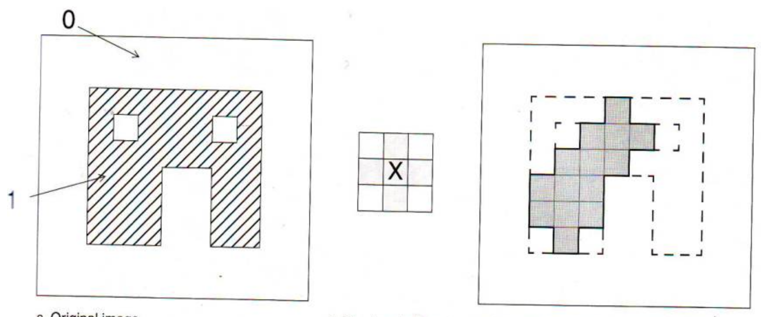


a. Original image. b. Structural element; x = origin c. Image after erosion; original in dashes.

Because the structuring element is 3 pixels wide, the 2-pixel-wide right leg of the image object was eroded away, but the 3-pixel-wide left leg retained some of its center pixels.

These two basic operations, dilation and erosion, can be combined into more complex sequences. The most useful of these for morphological filtering are called opening and closing. Opening consists of an erosion followed by a dilation and can be used to eliminate all pixels in regions that are too small to contain the structuring element. See Figure below for an example of opening.

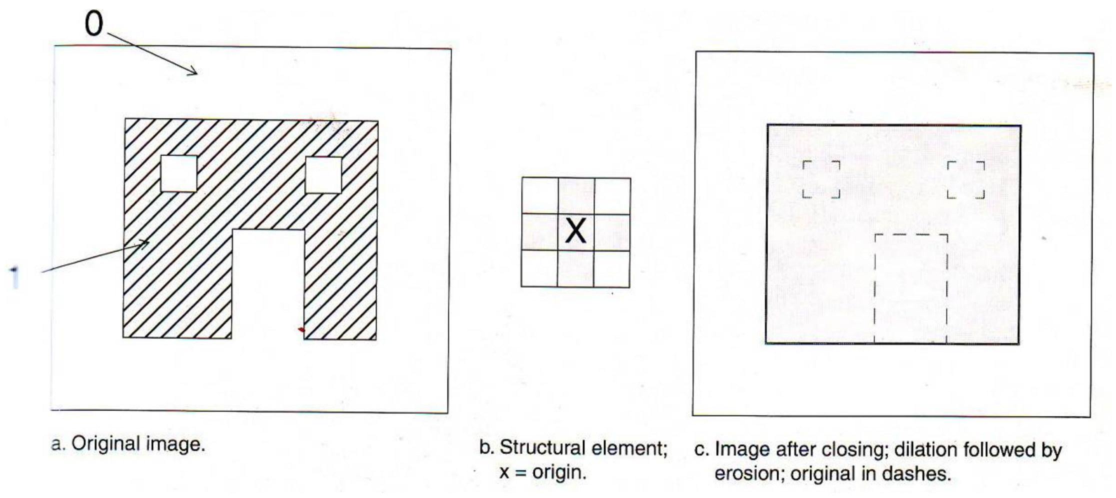
## Opening



a. Original image. b. Structural element; x = origin. c. Image after opening; erosion followed by dilation.

Closing consists of a dilation followed by erosion and can be used to fill in holes and small gaps. In Figure below we see that the closing operation has the effect of filling in holes and closing gaps.

## Closing

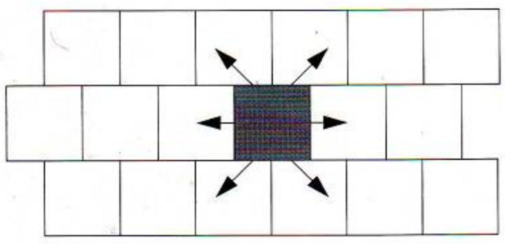


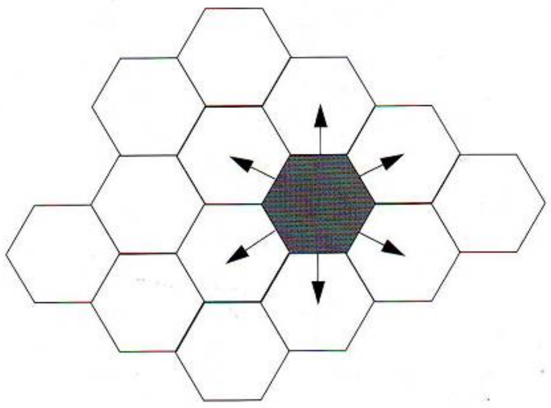
a. Original image. b. Structural element; x = origin. c. Image after closing; dilation followed by erosion; original in dashes.

Comparing Figure closing to Figure opening, we see that the order of operation is important. Closing and opening will have different results even though both consist of an erosion and a dilation.

Another approach to binary morphological filtering is based on an iterative approach. The usefulness of this approach lies in its flexibility. It is based on a definition of six-connectivity; in which each pixel is considered connected to its horizontal and vertical neighbors but to only two diagonal neighbors (the two on the same diagonal). This connectivity definition is equivalent to assuming that the pixels are laid out on a hexagonal grid, which can be simulated on a rectangular grid by assuming that each row is shifted by half a pixel (see Figure below).

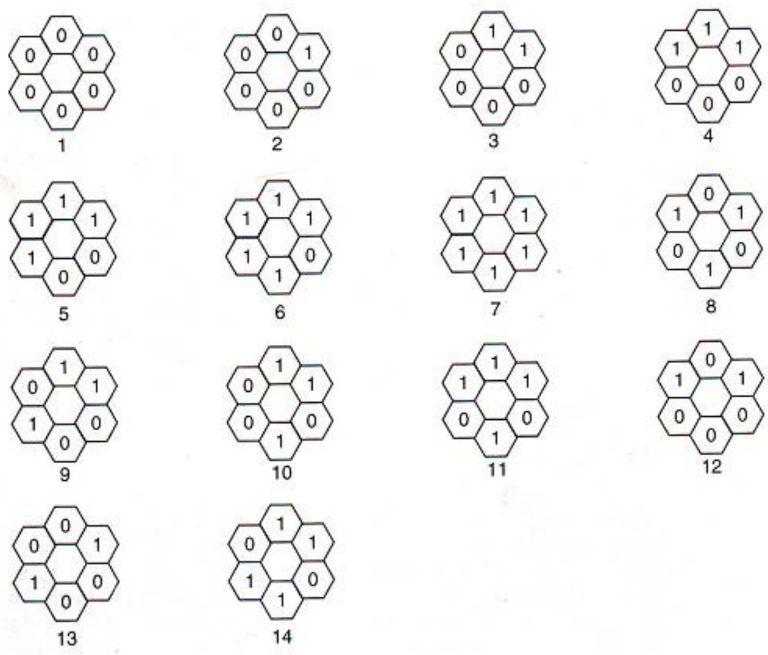
## Hexagonal Grid


a. Rectangular image grid with every other row shifted by one-half pixel.


b. Hexagonal grid.

With this definition a pixel can be surrounded by 14 possible combinations of 1’s and 0’s, as seen in Figure below;

## Surrounds for iterative Morphological Filtering



we call these different combinations surrounds. For this approach to morphological filtering, we define:

1. The set of surrounds S, where a = 1.

2. A logic function, L(a, b), where b is the current pixel value, specifies the output of the morphological function.

The function L( ) and the values of a and b are all functions of the row and column, (r, c), but for concise notation this is implied. Set S can contain any or all of the14 surrounds defined in Figure up. L(a, b) can be any logic function, but it turns out that the most useful are the AND and OR functions. The AND function tends to etch away at object boundaries (erosion), and the OR function tends to grow objects (dilation).

**EXAMPLE :**

Let S = {2, 3, 4, 5, 6} and L = a + b (+ = OR). Because L(a, b) is an OR operation, all pixels that are 1 in the original will remain 1. The only pixels that will change are those that are 0 in the original image and have a surround that is S (this means that a = 1). If we examine the set S, we see that this set contains all pixels that are surrounded by a connected set of 1's. This operation will expand the object, but because the surrounds of disconnected 1 pixels are not included in S, disjoint objects will not connect.

We can see from this example that this method is more flexible than the methods described earlier. We can use this technique to define methods for dilation, erosion, opening, and closing, as well as others. For this technique the selection of the set S is comparable to defining the structuring element in the previously described approaches, and the operation L(a, b) defines the type of filtering that occurs.

# LECTURE 5

## Image compression

**Image compression** involves reducing the size of image data files, while retaining necessary information**.**The reduced file is called the compressed file and is used to reconstruct the image ,resulting in the decompressed image. The original image, before any compression is performed, is called the uncompressed image file. The ratio of original uncompressed image file and the compressed file is referred to as the compression ratio.It is often written as $SIZE_U$:$SIZE_C$. The compression ratio is denoted by:

$$\text{Compression Ratio} = \frac{\text{Uncompressed File Size}}{\text{Compressed File Size}} = \frac{SIZE_U}{\text{SIZE}_C}$$

**EXAMPLE :**The original image is 256x256 pixels , 8 bits per pixel . this file is 65,536 bytes. After compression the image file is 6,854 bytes. The compression ratio is:

$SIZEL_U/SIZE_C$= 65536/6554 = 9.999 = 10. This can also be written as 10:1.

The reduction in file size is necessary to meet the bandwidth requirements for many transmission systems, the storage requirements in computer database.

## 1. Compression System Model:

The compression system model consists of two parts: the compressor anddecompressor.

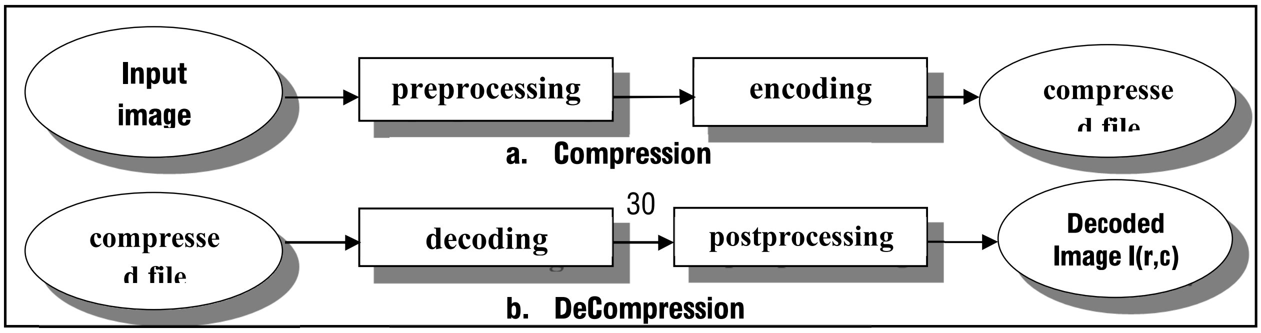
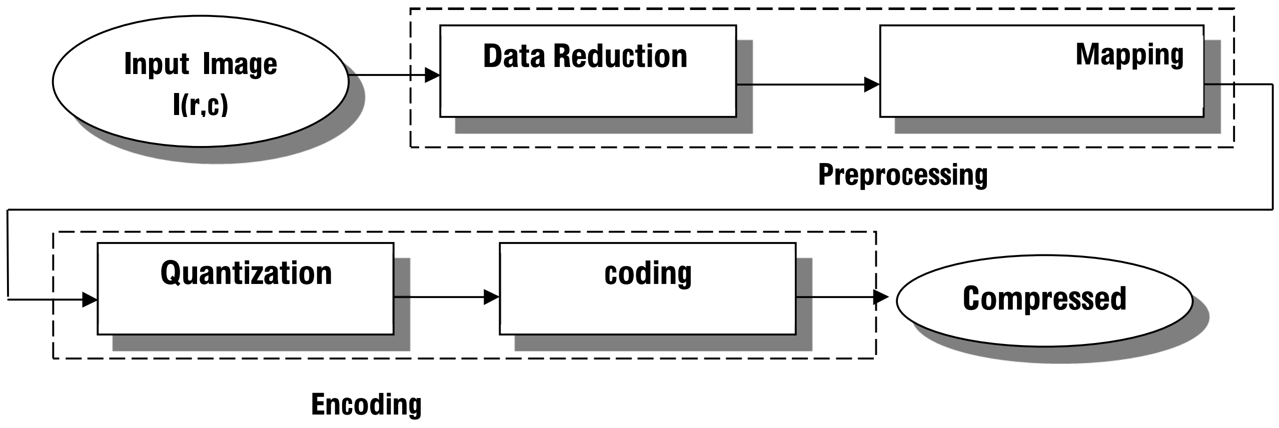


**Figure (1)Compression System Model.**

**The compressor consists of :-**

- a**preprocessing stage :**preprocessing is performed to prepare the image for the encoding process, and consists of any number of operations that are application specific.
    - **Data reduction**,Here, the image data can be reduced by gray-leve1 and/or spatial quantization, or they can undergo any desire image enhancement (for example, noise removal) process.
    - **The mapping process**, which maps the original image data into another mathematical space where it is easier to compress the data.
- **encoding** stage .

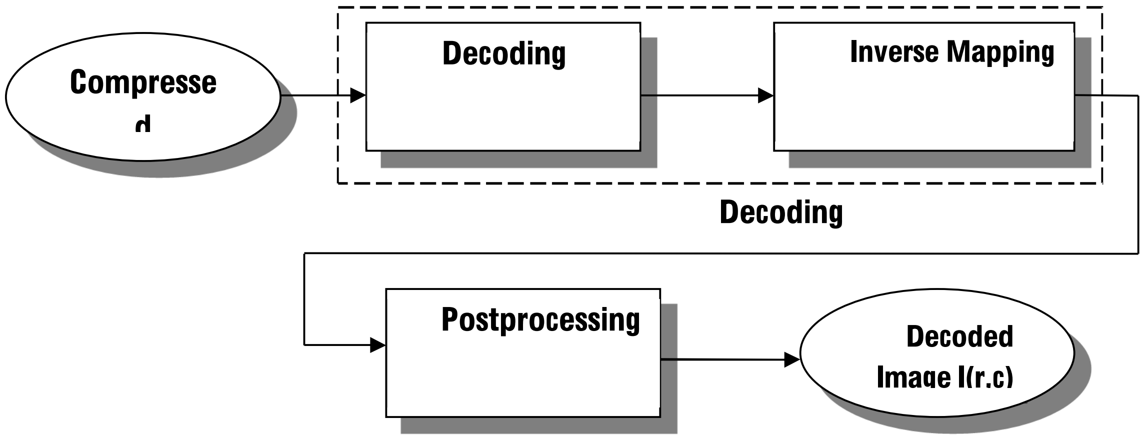
    - The quantization stage (as part of the encoding process), which takes the potentially continuous data from the mapping stage and puts it in discrete form.
    - Final stage of encoding involves coding the resulting data, which maps the discrete data from the quantizer onto a code in an optimal manner. A compression algorithm may consist of all the stages, or it may consist of only one or two of the stage.



**Figure(2) The Compressor**

**Decompressor consists of :-**

- **decoding process** is divided into two stages.
  - the decoding stage, takes the compressed file and reverses the original coding by mapping the codes to the original, quantized values.
  - Next, these values are processed by a stage that performs an inverse mapping to reverse the original mapping process. Finally.
- A **postprocessingstage:** After the compressed file has been decoded, postprocessing can be performed to eliminate some of the potentially undesirable artifacts brought about by the compression process. T**he Image postprocessed** to enhance the look of the final image. Often, many practical compression algorithms are a combination of a number of different individual compression techniques.



**Figure(3) the decompressor**

There are two types of image compressionmethods : Lossless Compression Methods and Lossy Compression Methods.

## <u>2.Lossless CompressionMethods:</u>

Lossless compression methods are necessary in some imaging applications. For example, with medical images, the law required that any archived medical images arestored without any data loss. In general, thelossless techniques alone provide marginalcompression of complex image data, oftenin the range of only a 10% reduction in file size.However, lossless compression techniques may for both preprocessing and postprocessing in image compressionalgorithms.Additionally, for simple images lossless techniques can provide substantialcompression.Entropy measure provide us with metric to evaluate coder performance .we can measure the average number of bits per pixel (length) in a coder by the following :

$$\boxed{L_{ave} = \sum_{i=0}^{L-1} I_i \; P_i}$$

Where $I_i$ = length in bits of the code for ith gray level

$P_{\text{i}}$ = histogram (probabilityof ithgray level)

This can then compared to the entropy ,which provides the theoretical minimum for average number of bits per pixel that could be used to encode the image

## • Huffman Coding

The Huffman code developed by D. Huffman in1952, is a minimum length code.
This means that given the statistical distribution of the gray levels (the histogram), theHuffman algorithm will generate a code that is as close as possible to the minimum bound' the entropy'.For example, Huffman coding alone will typically reduce the file by 10 to 50%, but this ratio can be improved to 2:1 or 3:1 by preprocessing for irrelevant information removal.

**The Huffman algorithm can be described in five steps:**

**l.** Find the gray-level probabilities for the image by finding the histogram.

**2**.order the input probabilities (histogram magnitudes) from smallest to largest.

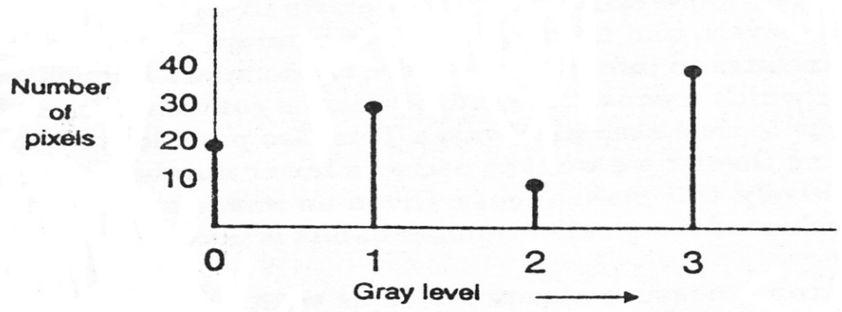
**3**.Combine the smallest two by addition.

**4**. GoTo step 2, until only two probabilities are left.

**5.**By working backward along the tree, generate code by alternating assignment of 0 and 1.

**Example 1**(Huffman Coding Example)

we have an image with 2 bit/pixel ,giving four possiblegray levels.the image is 10 rows by 10 columns.



$$g_0 = \frac{20}{100} = 0.2$$

$$g_1 = \frac{30}{100} = 0.3$$

$$g_2 = \frac{10}{100} = 0.1$$

$$g_3 = \frac{40}{100} = 0.4$$

a. Step 1: Histogram.

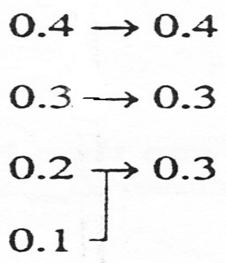
$g_3 \rightarrow 0.4$

$g_1 \rightarrow 0.3$

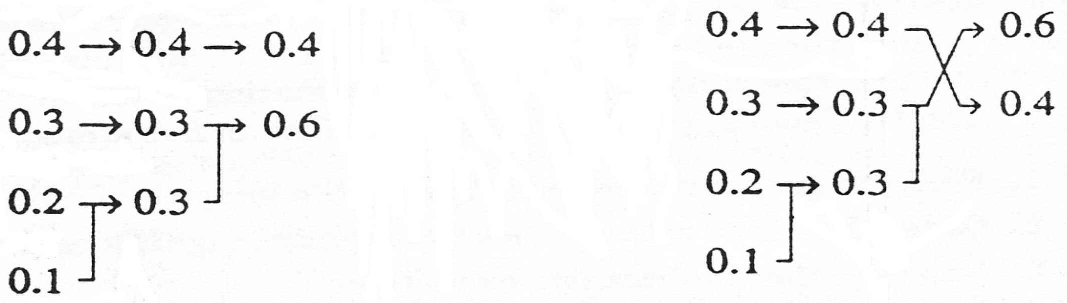
$g_0 \rightarrow 0.2$

$g_2 \rightarrow 0.1$

b. Step 2: Order.

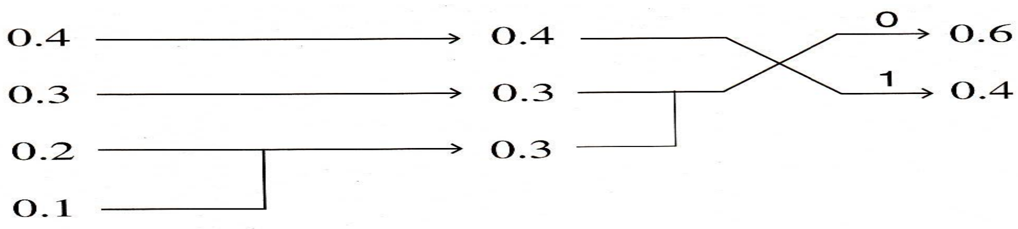


c. Step 3: Add.

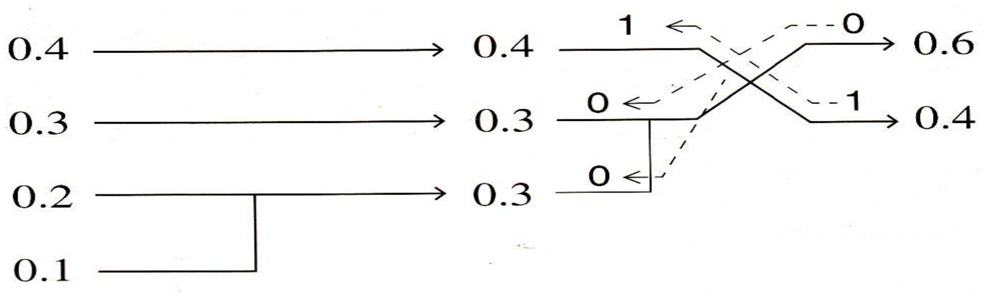


d. Step 4: Reorder and add until only two values remain.

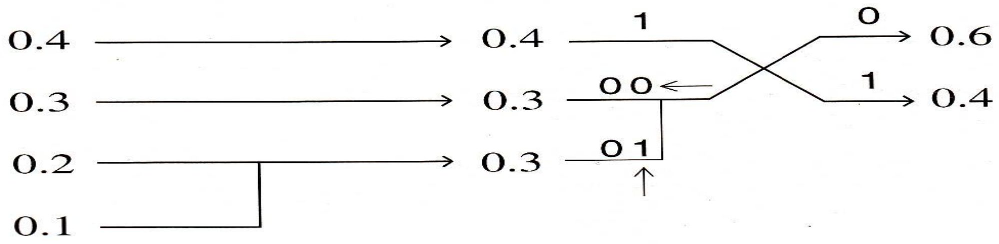
Step 5 ( haffman code):-



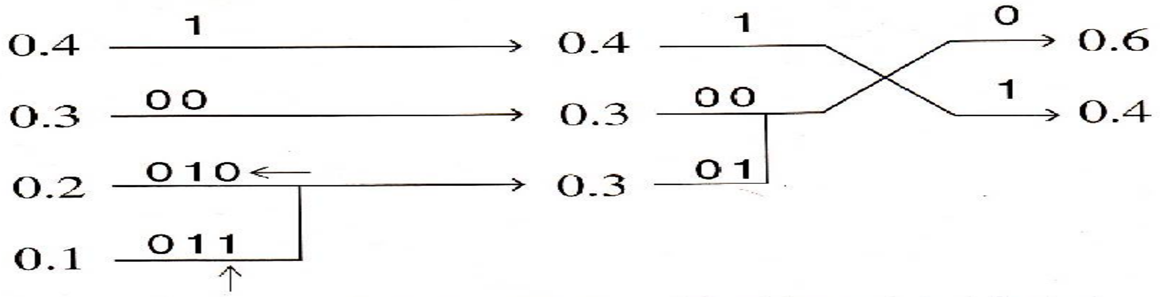
a. Assign 0 and 1 to the rightmost probabilities.



b. Bring 0 and 1 back along the tree.



c. Append 0 and 1 to previously added branches.



d. Repeat the process until the original branch is labeled.

| Original Gray Level (Natural Code) | Probability | Huffman code |
|---|---|---|
| $g_0$: $00_2$ | 0.2 | $010_2$ |
| $g_1$: $01_2$ | 0.3 | $00_2$ |
| $g_2$: $10_2$ | 0.2 | $011_2$ |
| $g_3$: $11_2$ | 0.4 | $1_2$ |

Note that two of the gray levels now have 3 bits assigned to represent them,but one gray level only has 1 bit assigned to represent it. The gray level represented by 1,g3, is the most likely to occur(40% of the time) and thus has the least information in the information theoretic sense.Note(symbols with less information require fewer bits to represent them. From the example we can see that theHuffman code is highly dependent on the histogram, so any preprocessing to the histogram may help the compression ratio. Let us examine the entropy in bits per pixel and average bit length for the Huffman coded image file .

$$\text{entropy} = -\sum_{i=0}^{3} P_i \, log_2 \; (P_i)$$

$$=[(0.2)\, log_2\, (0.2) + (0.3)\, log_2\, (0.3) +(0.1)\, log_2\, (0.1)+(0.4)\, log_2\, (0.4)$$

$$=1.846 \text{ bits/pixel}$$

(note : $log_2$ (x) can be found by taking $log_{10}$ (x) and multiplying by 3.322)

$$L_{ave} = \sum_{i=0}^{L-1} I_i \; P_i$$

$$=3(0.2)+2(0.3)+3(0.1)+1(0.4)$$

=1.9 bits/pixels (average length with Huffman code)

# LECTURE 6

## Image compression

- **Run-Length Coding**

Run-length coding (RLC) is an image compression method that works by countingthe number of adjacent pixels with the same gray-level value. This count, called the run length, is then coded and stored. Here we will explore several methods of run-lengthcoding-basic methods that are used primarily for binary (two-valued) images andextended versions for gray-scale images. Basic RLC is used primarily for binary images but can work with compleximages that have been preprocessed by thresholding to reduce the number of gray levels to two.

- **basic RLC**There are various ways to implement basic RLC, and the **first step** is todefine the required parameters. We can either use horizontal RLC, counting along therows, or vertical RLC, counting along the columns. In basic horizontal RLC the number of bits used for the coding depends on the number of pixels in a row. If the row has$2^n$ pixels, then the required number of bits is n, so that a run that is the length of theentire row can be coded.

A256 x 256 image requires 8 bits, since $2^8 = 256$.

A 512 x 512 image requires 9 bits, since $2^9 = 512$.

**The next step** is to define a convention for the first RLC number in a row-does it

represent a run of 0's or 1's? Defining the convention for the first RLC number to represent 0's, we can look at the following example.

**Example:**

The image is an 8 x 8 binary image, which requires 3 bits for each run-length coded word. In theactual image file are stored 1's and 0's, although upon display the 1's become 255 (white) and the0's are 0 (black). To apply RLC to this image, using horizontal RLC:imageCompression.

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

The RLC numbers are:First row: 8

Second row:0,4,4

Third row: 1, 2, 5

Fourth row;1,5,2

Fifth row: 1, 3,2,l,l

Sixth row 2,1,2,2,1

Seventh row: 0, 4, 1,1,2

Eighth row: 8

Note that in the second and seventh rows, the first RLC number is 0, since we are using the convention that the first number corresponds to the number of zeros in a run.

- **RLC(G,L)** : Another way to extend basic RLC to gray level images is to include gray level of particular run as part of the code. here instead of single value for a run, two parameter are used to characterize the run. the pair (G,L) correspond to the gray-level l value *G* and the run-length *L*. This technique is only effective with images containing a small number of gray levels .

Example:

Given the following 8×8 ,4-bit image:-

$$\begin{bmatrix} 10 & 10 & 10 & 10 & 10 & 10 & 10 & 10 \\ 10 & 10 & 10 & 10 & 10 & 12 & 12 & 12 \\ 10 & 10 & 10 & 10 & 10 & 12 & 12 & 12 \\ 0 & 0 & 0 & 10 & 10 & 10 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 & 0 & 0 & 0 \\ 5 & 5 & 5 & 10 & 10 & 9 & 9 & 10 \\ 5 & 5 & 5 & 4 & 4 & 4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

First row: 10,8

Second row: 10,5 12,3

Third row: 10,512,3

Fourth row: 0,3 10,3 0,3

Fifth row: 5,3 0,5

Sixth row: 5,310,2 9,2 10,1

Seventh row: 5,34,3 0,2

Eighth row: 0,8

These numbers are then stored in the RLC compressed file as:

10,8, 10,5, 12,3,L0,5,L2,3,0,3, 10,3,0,3, 5,3,0,5,5,3,70,2,9,2,10,1,5,3,4,3,0,2,0,8

The decompression process requires the number of pixels in a row, and the type

of coding used.

## 3.LOSSY COMPRESSION METHODS :

In order to achieve high compression ratios with complex images, lossy compression *methods* are *required. images can be compressed 10 to 20 times with virtually no visible information loss, and 30* to *50* with *minimal* degradation. Image enhancement and restoration techniques can be combined with lossy compression schemes to improve appearance of the decompressed image.

-Lossy compression is performed in both the spatial and transform domains. We will explore method that utilize each of these domains.

❖ In the spatial domain we will discuss :

1- gray-level run-length coding (GLRLC) .

2- block truncation coding (BTC).

❖ in the transform domain we will discuss :

1- JPEG algorithm

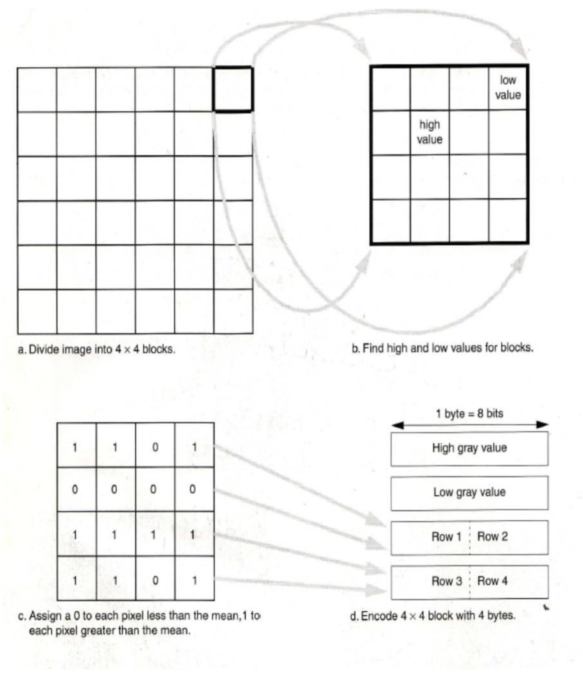
### • Gray-Level Run-Length Coding :

The RLC can also be used for lossy image compression by reducing the number of gray levelsand then applying standard RLC techniques. A more sophisticated RLC algorithm for encoding gray-level images is called the dynamic window-based RLC.This algorithm relaxes the critertion of the runs being the same value and allows for the runs to fall within a gray-level range, called the dynamic window range.

This range is dynamic because it starts out larger than theactual gray-level window range, and maximum and minimum values are narroweddown to the actual range as each pixel value is encountered. This process continuesuntil a pixel is found out of the actual range. The image isencoded with two values,one for the run length and one to approximate the gray-level value of the run, Thisapproximation can simply be the average of all the gray-level values in the run.

- **Block Truncation Coding :**

The basic form of BTC divides the image into 4 x 4 blocks and codes each blockusing a two-level quantizer. The two levels are selected so that the mean and varianceof the gray levels within the block are preserved. Each pixel value within the block isthen compared with a threshold, typically the mean, and then is assigned to one of thetwo levels, If it is above the mean, it is assigned the high-level code (1); if it is belowthe mean, it is assigned the low-level code (0). After this is done, the 4 x 4 block is encoded with 4 bytes: Since the original 4 x 4subimage has 16 bytes and the resulting code has 4 bytes (two for the high and lowvalues, and two for the bit string), this provides a 16:4 or 4:1 compression.


low value
high value
a. Divide image into 4 × 4 blocks.
b. Find high and low values for blocks.
1 1 0 1
0 0 0 0
1 1 1 1
1 1 0 1
1 byte = 8 bits
High gray value
Low gray value
Row 1 Row 2
Row 3 Row 4
c. Assign a 0 to each pixel less than the mean,1 to each pixel greater than the mean.
d. Encode 4 × 4 block with 4 bytes.

Although the results of this algorithm are image dependent, it tends to produce images with blocky effects as shown in the Figure1.2. These artifacts can be smoothed by applying enhancement techniques such as median and average (lowpass) filters.

- **Transform Coding :**

Transform coding is a form of block coding done in the transform domain. The image is divided into blocks, or subimages, and the transform is `calculated for each block. Any of the previously defined transforms can be used, frequency (e.g. Fourier) or sequency (e.g. Walsh), but it has been determined that the discrete cosine transform

(DCT) is optimal for most images. After the transform has been calculated, the transform coefficients are quantized and coded.

## 4. The objective fidelity criteria

Commonly used objective measures are the **root—mean-square error** ( **eRMS**), the**root—mean—square signal—to—noise ratio (SNRRMS)**, and the **peak signal-to—noise ratio (SNRPEAK)**. We can define the error between an original, uncompressed pixel value andthe reconstructed (decompressed) pixel value as**:**

$$\begin{aligned} \text{error}(r, c) &= \hat{I}(r, c) - I(r, c) \\ \text{where } I(r, c) &= \text{the original image} \\ \hat{I}(r, c) &= \text{the decompressed image} \end{aligned}$$

Next, we can define the total error in an N *N decompressed image as:

$$\text{Total error} = \sum_{r=0}^{N-1} \sum_{c=0}^{N-1} [\hat{I}(r, c) - I(r, c)]$$

**The root-mean-square** error is found by taking the square root ("root") of the errorsquared ("square") divided by the total number of pixels in the image ("mean") as:

$$e_{\text{RMS}} = \sqrt{\frac{1}{N^2} \sum_{r=0}^{N-1} \sum_{c=0}^{N-1} [\hat{I}(r, c) - I(r, c)]^2}$$

The smaller the value of the error metrics, the better the compressed image represents the original image. Alternately with the **signal-to-noise (SNR)** metrics, a largernumber implies a better image. The SNR metrics consider the decompressed image**î(r, c)** to be the "signal" and the error to be "noise." We can define the root-mean- squaresignal- to -noise ratio as:

$$SNR_{RMS} = \sqrt{\frac{\sum_{r=0}^{N-1}\sum_{c=0}^{N-1}[\hat{I}(r,c)]^2}{\sum_{r=0}^{N-1}\sum_{c=0}^{N-1}[\hat{I}(r,c) - I(r,c)]^2}}$$

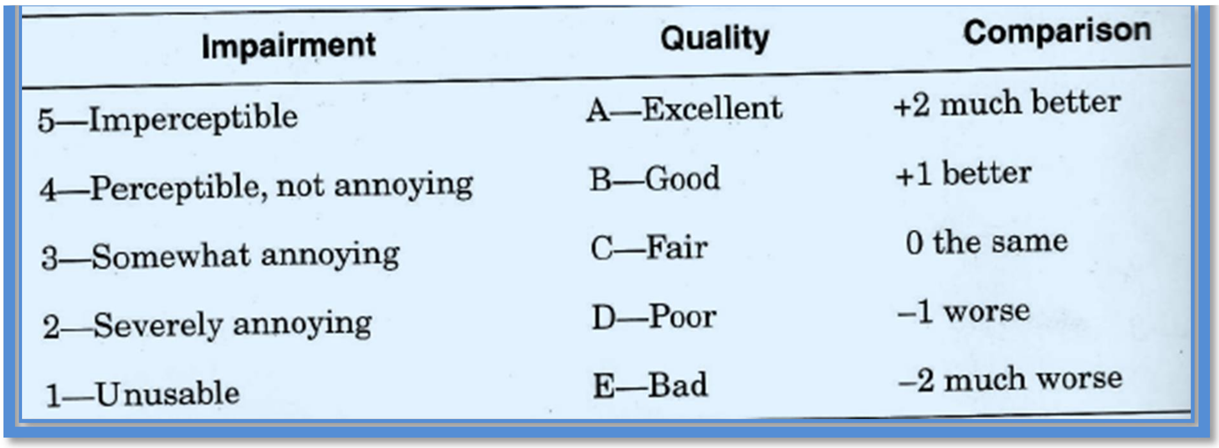
Another related metric, the **peak signal-to-noise ratio**, is defined as:

$$SNR_{PEAK} = 10\log_{10}\frac{(L-1)^2}{\frac{1}{N^2}\sum_{r=0}^{N-1}\sum_{c=0}^{N-1}[\hat{I}(r,c) - I(r,c)]^2}$$

These objective measures are often used in research because they are easy to generate and seemingly unbiased, but remember that these metrics are not necessarily correlated to our perception of an image. The subjective measures are a better method for comparison of compression algorithms, if the goal is to achieve high-quality images as defined by our visual perception.

**The subjectivefidelity criteria**

Subjective testing is performed by creating a database of images to be tested, gathering a group of people that are representative of the desired population, and then having all the test subjects evaluate the images according to a predefined scoringcriterion. The results are then analyzed statistically, typically using the averages andstandard deviations as metrics. Subjective fidelity measures can be classified intothree categories. The first type are referred to as impairment tests, where the test subjects score the images in terms of how bad they are. The second type are quality tests,where the test subjects rate the images in terms of how good they are. The third typeare called comparison tests, where the images are evaluated on a side-by-side basis.The comparison type tests are considered to provide the most useful results, as theyprovide a relative measure, which is the easiest metric for most people to determine.Impairment and quality tests require an absolute measure, which is more difficult todetermine in an unbiased fashion. In **Table (1)**are examples of internationallyaccepted scoring scales for these three types of subjective fidelity measures.



| Impairment | Quality | Comparison |
|---|---|---|
| 5—Imperceptible | A—Excellent | +2 much better |
| 4—Perceptible, not annoying | B—Good | +1 better |
| 3—Somewhat annoying | C—Fair | 0 the same |
| 2—Severely annoying | D—Poor | –1 worse |
| 1—Unusable | E—Bad | –2 much worse |

**Table(1) Subjective Fidelity Scoring Scales**

# LECTURE 7

## *DISCRETE TRANSFORMS*

A transform maps image data into a different mathematical space via a transformation equation. We may transform our image data into alternate color spaces to achieve image segmentation. However, the color transforms mapped data from one color space to another color space with a one-to-one correspondence between a pixel in the input and the output**.**

**Here, we are mapping the image data from the spatial domain to the frequency domain (also called the spectral domain), where all the pixels in the input (spatial domain) contribute to each value in the output (frequency domain).**

This is illustrated in **Figure 1**. These transforms are used as tools in many areas of engineering and science, including computer imaging.

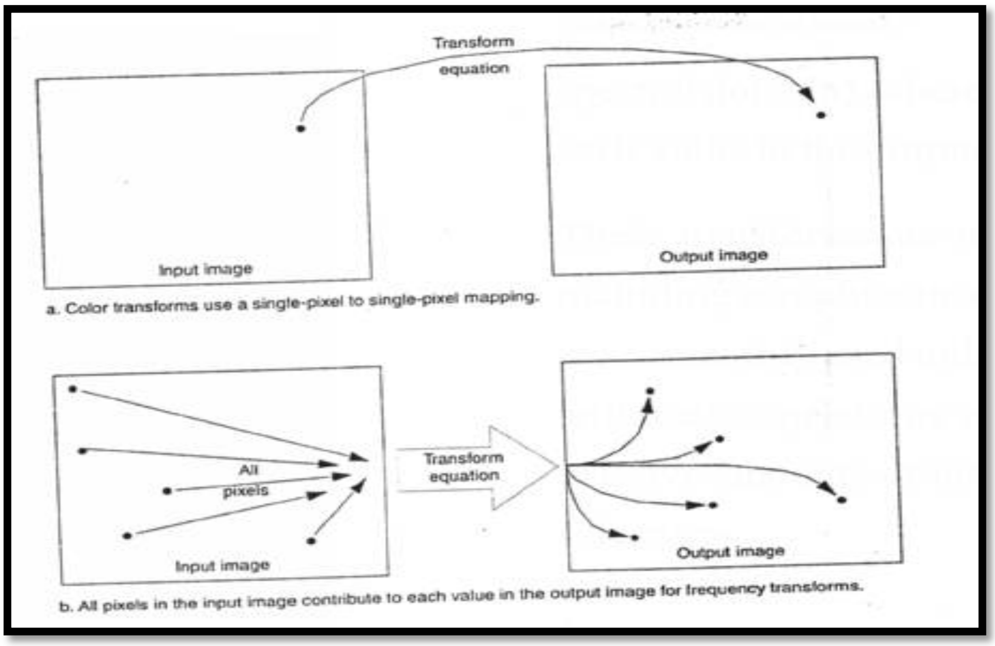
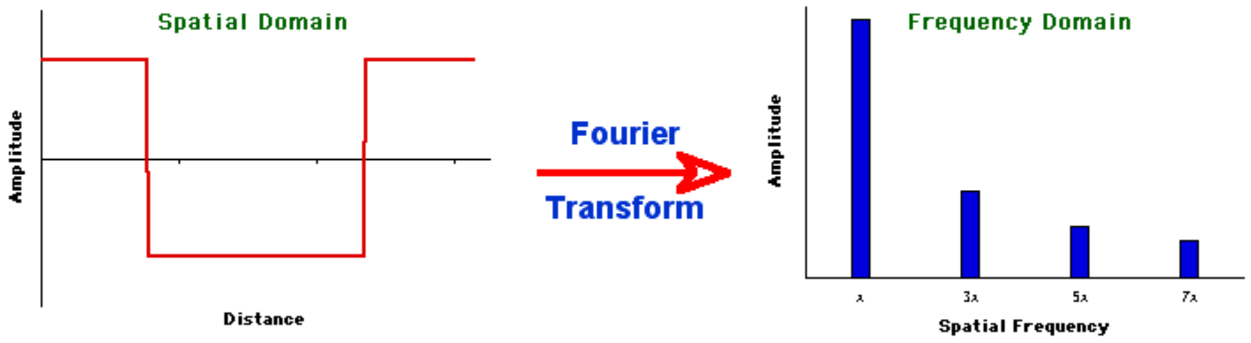

Transform
equation
Input image
Output image
a. Color transforms use a single-pixel to single-pixel mapping.
All
pixels
Input image
Transform
equation
Output image
b. All pixels in the input image contribute to each value in the output image for frequency transforms.

Figure (1) discrete transform

**For example:**



## 1-Fourier transform

The Fourier transform is the most well-known, and the most widely used, transform. This transform allows for the decomposition of an image into a weighted sum of 2-D sinusoidal terms. Assuming an N x N image, the equation for the 2-D discrete Fourier transform is:

$$[F(u,v)] = \frac{1}{N}\sum_{r=0}^{N-1}\sum_{c=0}^{N-1} I(r,c)\, e^{-j2\pi\frac{(ur+vc)}{N}}$$

The base of the natural logarithmic function e is about 2.71828; j, the imaginary coordinate for a complex number equals √-1 The basis functions are sinusoidal in nature, as can be seen by Euler's identity: $e^{jx}=\cos x+j\sin x$

So we can also write the Fourier Transform equation as

$$[F(u,v)] = \frac{1}{N}\sum_{r=0}^{N-1}\sum_{c=0}^{N-1} I(r,c)\left[\cos\left(\frac{n\pi}{N}(ur+vc)\right)+j\sin\left(\frac{2\pi}{N}(ur+vc)\right)\right]$$

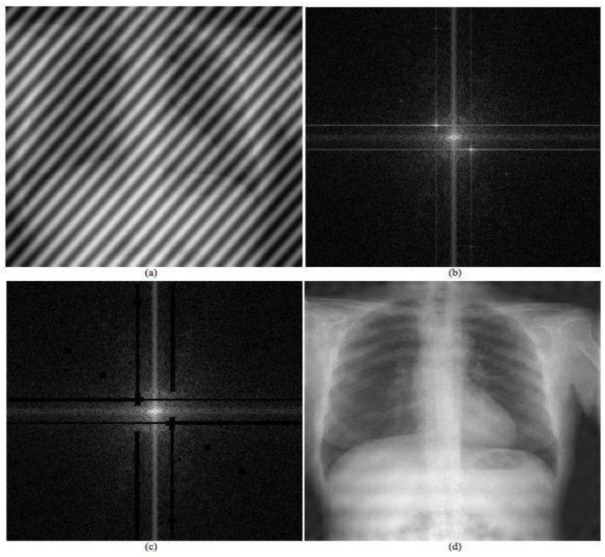
In this case, F(u, v) is also complex, with the real part corresponding to the cosine terms and the imaginary part corresponding to the sine terms. If we represent a complex spectral component F(u,v)=R(u,v)+jI(u,v) , where R(u,v) in the real part and I (u,v) is the imaginary part, then we can define the magnitude and phase of a complex spectral component us:

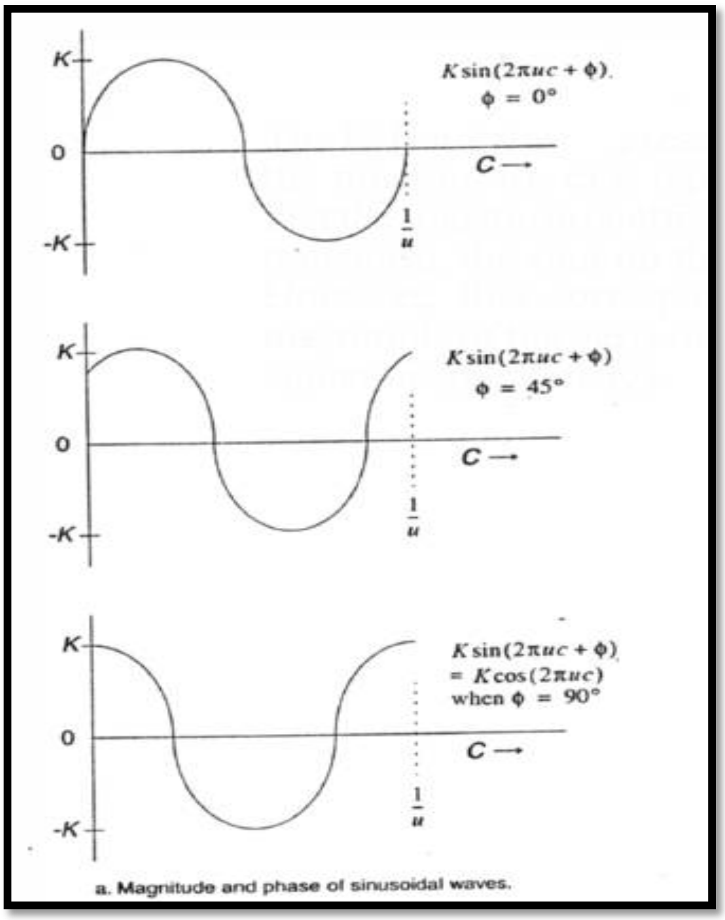
**MAGNITUDE=**$|F(u,v)| = \sqrt{[R(u,v)]^2 + [I(u,v)]^2}$

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

**The magnitude of a sinusoid is simply its peak value, the phase , data contain information about where objects arc in an image.** After we perform the transform, if we want to get our original image back, we need to apply the inverse transform. The inverse Fourier transform is given by:

$$F^{-1}[F(u,v)] = I(r,c) = \frac{1}{N}\sum_{u=0}^{N-1}\sum_{v=0}^{N-1} F(u,v)e^{j2\pi\frac{(ur+vc)}{N}}$$





Figure(2) magnitude and phase of Fourier transform

## 2 -Cosine Transform

The cosine transform, like the Fourier transform, uses sinusoidal basis functions. The difference is that the cosine transform basis functions are not complex; they use only cosine functions and not sine functions. Assuming an N x N image, the discrete cosine transform equation is given by:

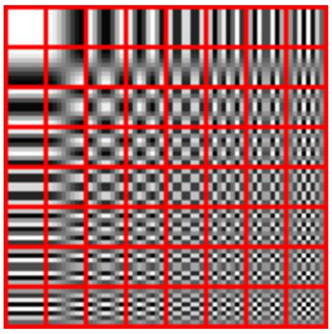
$$C(u,v) = \alpha(u)\alpha(V)\sum_{r=0}^{N-1}\sum_{c=0}^{N-1} I(r,c)\cos[\frac{(2r+1)u\pi}{2N}]\cos\frac{(2c+1)v\pi}{2N}]$$

Where:

$$\alpha(u), \alpha(v) = \begin{cases} \sqrt{\frac{1}{N}} & \text{for } u,v = 0 \\ \sqrt{\frac{2}{N}} & \text{for } u,v = 1,2,\ldots\ldots,N-1 \end{cases}$$

Because this transform uses only the cosine function, it can be calculated using only real arithmetic (not complex). The cosine transform is often used in image compression, in particular the Joint Photographic Experts Group (JPEG) image compression method, The inverse cosine transform is given by :

$$C^{-1}(u,v) = \alpha(u)\alpha(V)\sum_{r=0}^{N-1}\sum_{c=0}^{N-1} C(u,v)\cos[\frac{(2r+1)u\pi}{2N}]\cos\frac{(2c+1)v\pi}{2N}]$$



Discrete cosine transform basis images

A photo of a cat with the compression rate decreasing using (DCT), and hence quality increasing, from left to right

## 3- Walsh-Hadamard Transform

The Walsh-Hadamard transform differs from the Fourier and cosine transforms in that the basis functions are not sinusoids. The basis functions are based on square or rectangular waves with peaks of $\pm 1$ . One primary advantage of a transform with these types of basis functions is that the computations are very simple, all we need to do is multiply each pixel by $\pm 1$ , as is seen is the Walsh-Hadamard Transform equation (assuming an N xN image):

$$WH(u,v)= \frac{1}{N}\sum_{r=0}^{N-1}\sum_{c=0}^{N-1} I(r,c)(-1)^{\sum_{i=0}^{n-1}[bi(r)pi(u) + bi(c)pi(v)]}$$

where $N = 2^n$, and bi(r) is found by considering r as a binary number and finding the ***i*th** bit.

**Example 1:**

n=3 (3bits, so N=8) ,and r=4

r in binary is 100, so b2(r)=1, b1(r)=0, b0(r)=0.

**Example 2:**

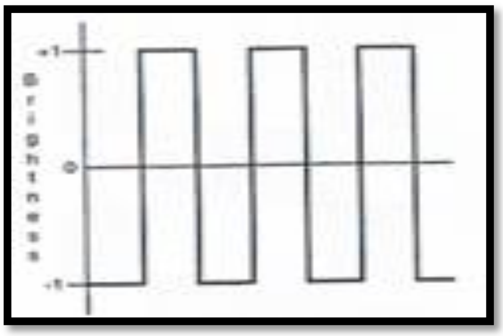
n=4, (4bits , so N=16), and r=2

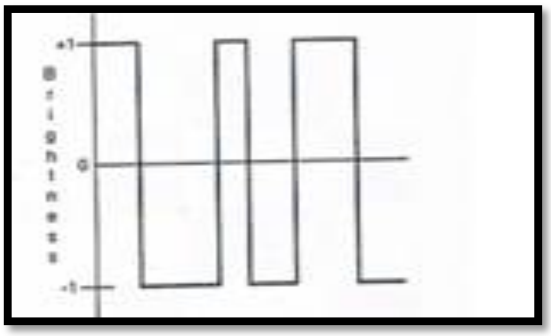
r in binary is 0010, so b3(r)=0, b2(r)=0 , b1(r)=1, b0(r)=0. and

$p_i(u)$ is found as follows:

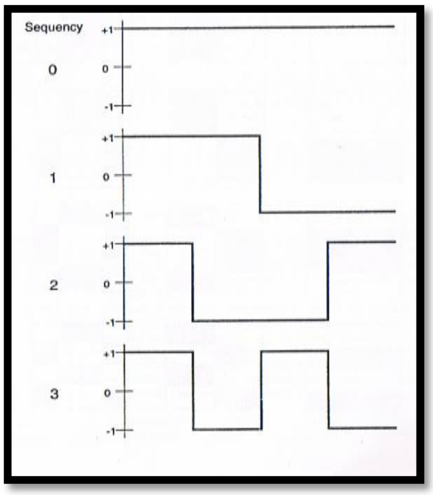
$$p_0(u) = b_{n-1}(u)$$
$$p_1(u) = b_{n-1}(u) + b_{n-2}(u)$$
$$p_2(u) = b_{n-2}(u) + b_{n-3}(u)$$
$$\vdots$$
$$p_{n-1}(u) = b_1(u) + b_0(u)$$

**Form of the walsh-Hadamard Basis Functions**





The sum are performed in modulo 2 arithmetic, and the values for bi(c) and pi(v) are found in a similar manner Strictly speaking we cannot call the Walsh-Hadamard transform a **frequency transform** because the basis functions do not exhibit the frequency concept in the manner of sinusoidal functions. we have a measure that is comparable to frequency and we call this **sequency**. In the Figure below we see the 1-D Walsh-Hadamard basis functions



Also can this process to generate the 2-D basis image for any function that has a separable
basis .Remember that the separable means that the basis function can be expressed as a product of terms that depend only on one of the variable pairs. First doing a 1-D transform on the rows and then performing the 1-D transform on the resulting columns.

1-D basis vector for v=2

|  | +1 | -1 | -1 | +1 |  |
|---|---|---|---|---|---|
| +1 | +1 | -1 | -1 | +1 |  |
| u | -1 | +1 | **+1** | **-1** | = |
| -1 | +1 | -1 | -1 | +1 |  |
| 3 | -1 | +1 | +1 | -1 | +1 |

-1

2-D basis image for (u,v)=(3,2)

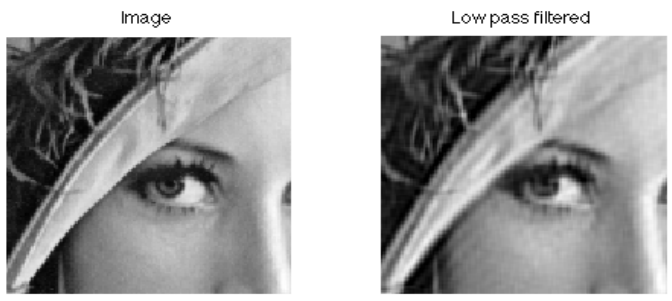
The inverse Walsh-Hadamard transform equation is:

$$WH^{-1}[WH(u,v)]= I(r,c)= \frac{1}{N}\sum_{u=0}^{N-1}\sum_{v=0}^{N-1} WH(u,v)\ (-1) \sum_{r=0}^{N-1}[b_i(r)\ p_i(u)+b_i(c)p_i(v)]$$

## *4- Filtering*

After the image has been transformed into the frequency or sequency domain, we may want to modify the resulting spectrum. There are three types of filtering:-

1. **Lowpass filters** tend to blur images. They pass low frequencies and attenuate or eliminate the high-frequency information. They are used for image compression or for hiding effects caused by noise. It imparts a softer effect to the image . **Low-pass filtering** is performed by multiplying the spectrum by a filter and then applying the inverse transform to obtain the filtered image.

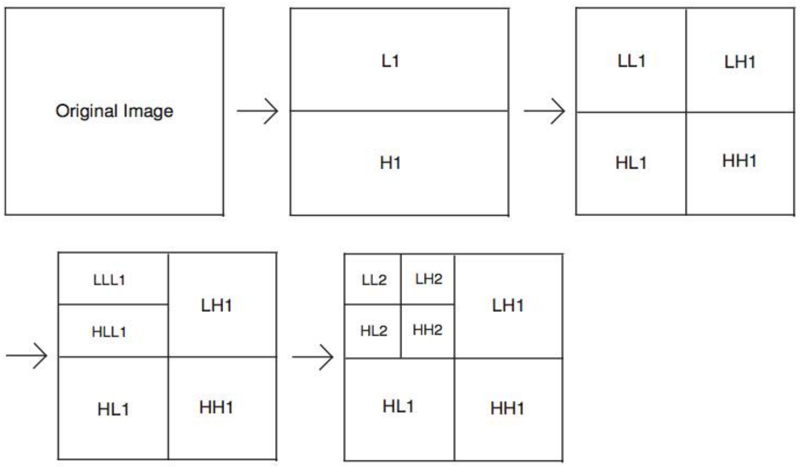


2- **A highpass filter** can be used for edge enhancement because it passes only high-frequency information, corresponding to places where gray levels are changing rapidly (edges in images are characterized by rapidly changing gray levels).

3-**The bandpass and bandreject filters** are specified by two cutoff frequencies, a low cutoff and a high cutoff shown. A special form of these filters is called a **notch** filter because it only notches out, or passes, specific frequencies . These three types of filters are typically used in image restoration, enhancement, and compression.

## *5 -Wavelet Transform*

The wavelet transform is really a family of transforms that satisfy specific conditions. the wavelet transform can be described as a transform that has basis functions that are shifted and expanded versions of themselves. the wavelet transform contains not just frequency information but spatial information as well. One of the most common models for a wavelet transform uses the Fourier transform and highpass and lowpass filters. The use of the wavelet transform is increasingly popular for **image compression**



(c)

Numerous filters can be used to implement the Wavelet Transform ,one of the commonly used is the Haar ,it is separable ,so they can be used to implement a wavelet transform by first convolving them with rows and then with columns. The Haar basis vectors are simple:

LOWPASS :1/√2[1  1]

HIGHPASS : 1/√2[1  -1]