University of Technology- Computer Science

Image Processing (1)

Fourth Class

2023-2024

Prof. Dr. Nidaa Flaih Hassan

- **Connectivity**
- **Distance Measures**

<u>Chapter Five : Histogram of Image</u>

- **Histogram Modification**
    - **Stretching mapping function**
    - **Shrinking mapping function**
    - **Sliding Technique.**
- **Histogram Equalization**
- **Histogram Features.**

<u>Chapter Six: Image Segmentation</u>

- **Region-based segmentation**
- **Clustering method.**
- **Boundary detection and Combined approaches.**

<u>Chapter Seven : Image Compression</u>

- **Compression System Model**
    - **Compression Ratio and Entropy**
- **Lossless Compression Method**
    - **Run Length Coding**
    - **Huffman Coding**

<u>Chapter Four: Discrete transform</u>

- **Fourier Transform.**
- **Cosine Transform.**
- **Wavelet Transform**

<u>Chapter Five: Image Fidelity Criteria</u>

- **Objective fidelity criteria**
- **Subjective fidelity criteria**
-

<u>References</u>

[1] Scotte E Umbaugh, “Digital Image Processing And Analysis Human And Computer Vision Applications With CVIPTOOLS, Second edition, CRC Press, Taylor & Francis Group. 2017

[2] Rafael C. Gonzalez , “ Digital Image Processing.”, Third Edition, Prentice-Hall, Inc. 2008

Chapter One
Introduction to Computer Vision and Image Processing

1.1 Computer Imaging

Can be defined a acquisition and processing of visual information by computer. Computer representation of an image requires the equivalent of many thousands of words of data, so the massive amount of data required for image is a primary reason for the development of many sub areas with field of computer imaging, such as image compression and segmentation .Another important aspect of computer imaging involves the ultimate “receiver” of visual information in some case the human visual system and in some cases the human visual system and in others the computer itself.

Computer imaging can be separate into two primary categories:

1. Computer Vision.
2. Image Processing.

(In computer vision application the processed images output for use by a computer, whereas in image processing applications the output images are for human consumption).

These two categories are not totally separate and distinct. The boundaries that separate the two are fuzzy, but this definition allows us to explore the differences between the two and to explore the difference between the two and to understand how they fit together (Figure 1.1).

Computer imaging can be separated into two different but overlapping areas.

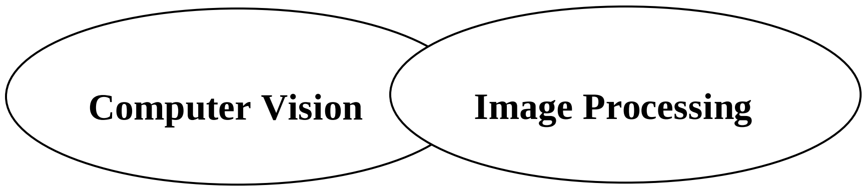


Figure (1.1): Computer Imaging [1].

Historically, the field of image processing grew from electrical engineering as an extension of the signal processing branch, whereas are the computer science discipline was largely responsible for developments in computer vision.

1.2 Computer Vision

Computer vision is a computer imaging where the application doses not involve a human being in visual loop. One of the major topics within this field of computer vision is image analysis.

1. Image Analysis: involves the examination of the image data to facilitate solving vision problem.

The image analysis process involves two other topics:

- Feature Extraction: is the process of acquiring higher level image information, such as shape or color information.
- Pattern Classification: is the act of taking this higher –level information and identifying objects within the image.

Computer vision systems are used in many and various types of environments, such as:

1. Manufacturing Systems: computer vision is often used for quality control, where the computer vision system will scan manufactured

items for defects, and provide control signals to a robotics manipulator to remove detective part automatically.

2. Medical Community: current example of medical systems to aid neurosurgeons (جراحة الاعصاب) during brain surgery, systems to diagnose skin tumors (سرطان الجلد) automatically.
3. The field of Law Enforcement and security in an active area for computer vision system development, with application ranging from automatic identification of fingerprints to DNA analysis.
4. Infrared Imaging (صور تحت الحمراء)
5. Satellites Orbiting (مدار الفضائيات).

1.3 Image Processing

Image processing is computer imaging where application involves a human being in the visual loop. In other words the image are to be examined and a acted upon by people.

Image processing systems are used in many and various types of environments, such as:

1. Medical community has many important applications for image processing involving various type diagnostics imaging , as an example MRI(Magnetics Resonance Imaging ( ) scanning, that allows the medical professional to look into the human body without the need to cut it open .
2. Computer – Aided Design , which uses tools from image processing and computer graphics , allows the user to design a new building or spacecraft and explore it from the inside out .
3. Virtual Reality is one application that exemplifies (يمثل) future possibilities

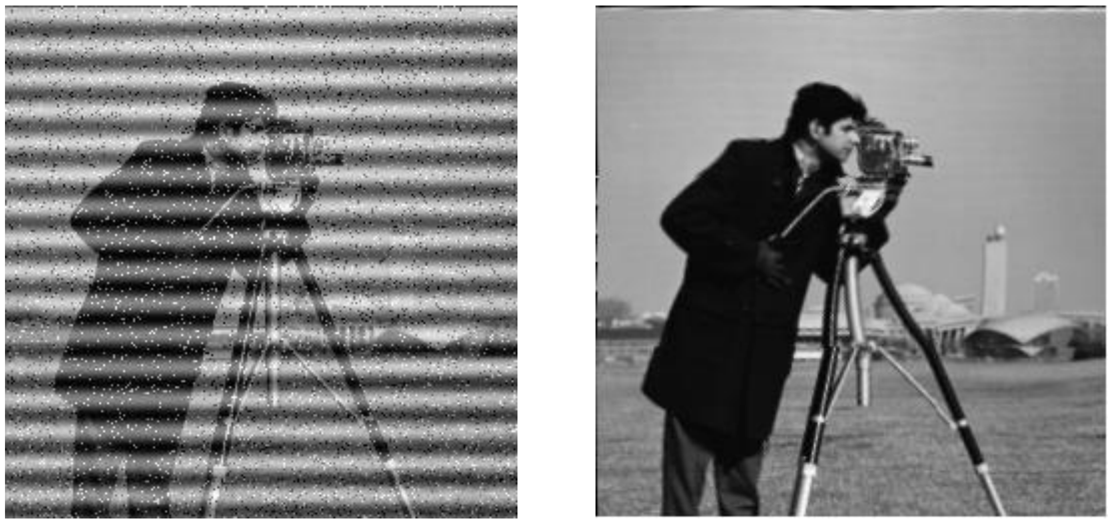
4. Image Processing is being used enable people to see what they look like with new haircut, a new pair of eyeglasses, or even anew noise.

The major topics within the field of image processing include:

1. Image restoration.
2. Image enhancement.
3. Image compression.

1.3.1 Image Restoration

Is the process of taking an image with some known, or estimated degradation, and restoring it to its original appearance. Image restoration is often used in the field of photography or publishing where an image was somehow degraded but needs to be improved before it can be printed(Figure 1.2).



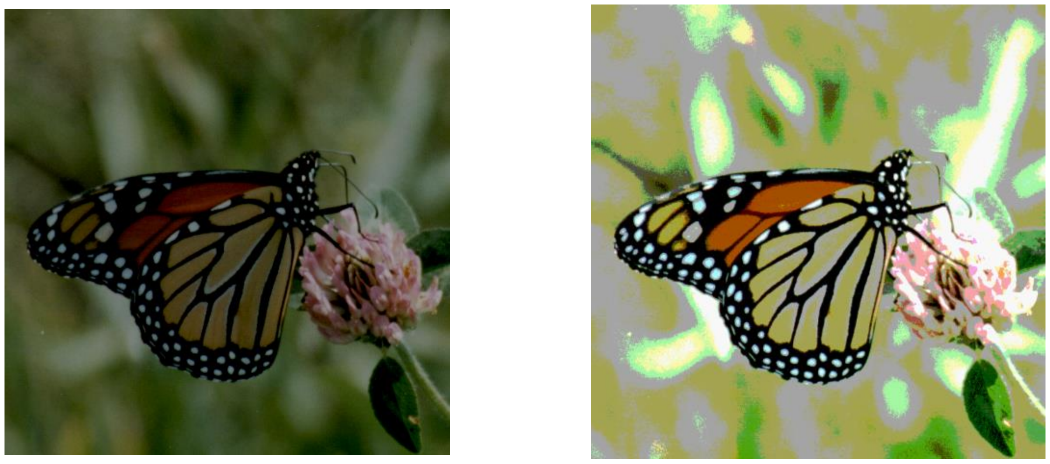
a. Image with distortion **b. Restored image**

Figure (1.2) Image Restoration

1.3.2 Image Enhancement

Involves taking an image and improving it visually, typically by taking advantages of Human Visual Systems responses. One of the simplest enhancement techniques is to simply stretch the contrast of an image.

Enhancement methods tend to be problem specific. For example, a method that is used to enhance satellite images may not suitable for enhancing medical images.



a. image with poor contrast **b. Image enhancement by contrast stretching**

Figure (1.3) Image Enhancement

1.3.1 Image Compression

Involves reducing the typically massive amount of data needed to represent an image. This done by eliminating data that are visually unnecessary and by taking advantage of the redundancy that is inherent in most images. Image data can be reduced 10 to 50 times, and motion image data (video) can be reduced by factors of 100 or even 200.

a. Image before compression (92) KB

b. Image after compression (6.59)KB

Figure (1.4): Image Compression.

1.4 <u>Computer Imaging Systems</u>

Computer imaging systems are comprised of two primary components types, hardware and software. The hard ware components can be divided into image acquiring sub system (computer, scanner, and camera) and display devices (monitor, printer).The software allows us to manipulate the image and perform any desired processing on the image data.

1.5 Digitization

The process of transforming a standard video signal into digital image. This transformation is necessary because the standard video signal in analog (continuous) form and the computer requires a digitized or sampled version of that continuous signal. The analog video signal is turned into a digital image by sampling the continuous signal at affixed rate. In the figure below we see one line of a video signal being sampled (digitized) by instantaneously measuring the voltage of the signal at fixed intervals in time.

The value of the voltage at each instant is converted into a number that is stored, corresponding to the brightness of the image at that point.

Note that **the image brightness of the image at that point** depends on both the intrinsic properties of the object and the lighting conditions in the scene.

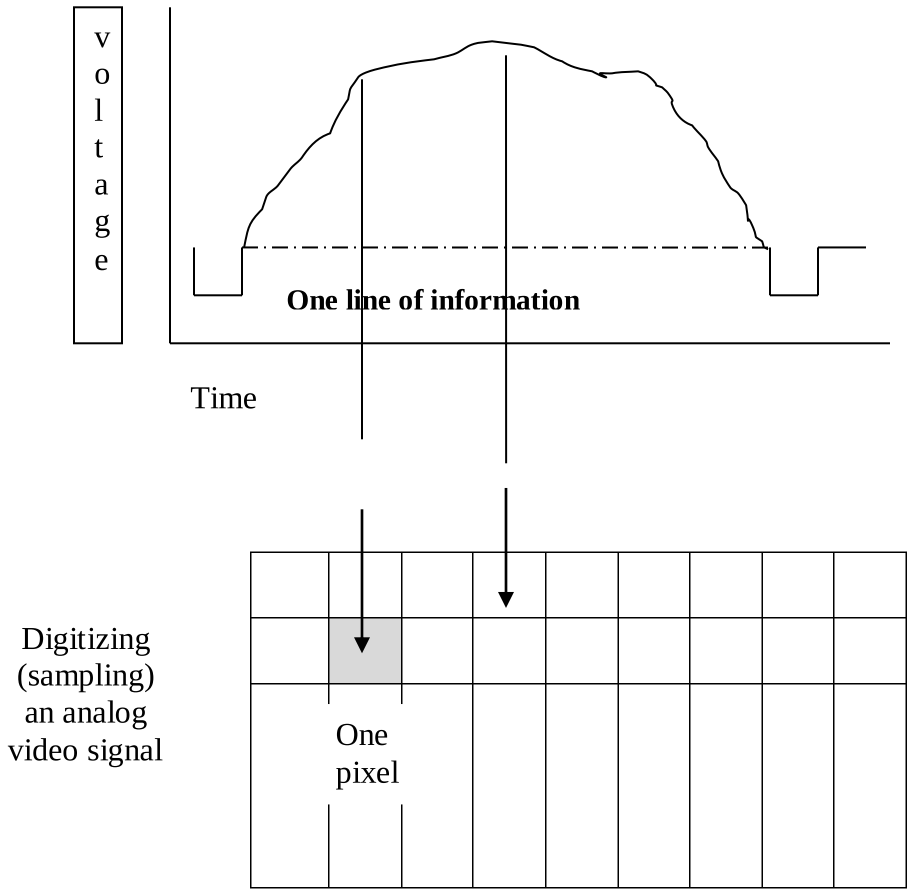


Figure (1.5) Digitizing (Sampling) an Analog Video Signal [1].

The image can now be accessed as a two-dimension array of data , where each data point is referred to a pixel (picture element).for digital images we will use the following notation :

I(r,c) = The brightness of image at the point (r,c)

Where r= row and c= column.

“When we have the data in digital form, we can use the software to process the data”.

The digital image is 2D- array as:

$$\begin{pmatrix} I(0,0) & I(0,1) \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots I(0,N-1) \\ I(1,0) & I(1,1) \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots I(1,N-1) \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ I(N-1,0) & I(N-1,1) \ldots\ldots\ldots\ldots\ldots\ldots\ldots I(N-1,N-1) \end{pmatrix}$$

In above image matrix, the image size is (N×N) [matrix dimension] then:

$Ng= 2^{m}$ ……….(1)

Where Ng denotes the number of gray levels m, where m is the no. of bits contains in digital image matrix.

<u>Example</u> : If we have (6 bit) in 128 × 128 image .Find the no. of gray levels to represent it ,then find the no. of bit in this image?

<u>Solution</u>:

$N_g= 2^{6}=64$ Gray Level

$N_b= 128 \times 128\times 6= 9.8304 \times 10^{4}$ bit

1.6 <u>The Human Visual System</u>

The Human Visual System (HVS) has two primary components:

- Eye.
- Brian.

* The structure that we know the most about is the image receiving sensors (the human eye).

* The brain can be thought as being an information processing unit analogous to the computer in our computer imaging system.

These two are connected by the optic nerve, which is really a bundle of nerves that contains the path ways for visual information to travel from the receiving sensor (the eye) to the processor (the brain).

1.7 <u>Image Resolution</u>

Pixels are the building blocks of every digital image. Clearly defined squares of light and color data are stacked up (مكدسة) next to one another both horizontally and vertically. Each **picture element (pixel for short)** has a dark to light value from 0 (solid black) to 255 (pure white). That is, there are 256 defined values. **A gradient** (ميل, نسبة الانحدار) is the gradual transition from one value to another in sequence. At the heart of any convincing photographic rendering is a smooth, seamless ((سلس and beautiful gradation.

In computers, resolution is the number of pixels (individual points of color) contained on a display monitor, expressed in terms of the number of pixels on the horizontal axis and the number on the vertical axis. The sharpness of the image on a display depends on the resolution and the size of the monitor. The same pixel resolution will be sharper on a smaller monitor and gradually lose sharpness on larger monitors because the same numbers of pixels are being spread out over a larger number of **inches**.

Display resolution is not measured in dots per inch as it usually is with printers ***(We measure resolution in pixels per inch or more commonly, dots per inch (dpi)).***

- A display with 240 pixel columns and 320 pixel rows would generally be said to have a resolution of **240×320**.

- Resolution can also be used to refer to the total number of pixels in a digital camera image. For example, a camera that can create images of 1600x1200 pixels will sometimes be referred to as a 2 megapixel resolution camera since 1600 x 1200 = 1,920,000 pixels, or roughly 2 million pixels.
- Below is an illustration of how the same image might appear at different pixel resolutions, if the pixels were poorly rendered as sharp squares (normally, a smooth image reconstruction from pixels would be preferred, but for illustration of pixels, the sharp squares make the point better).

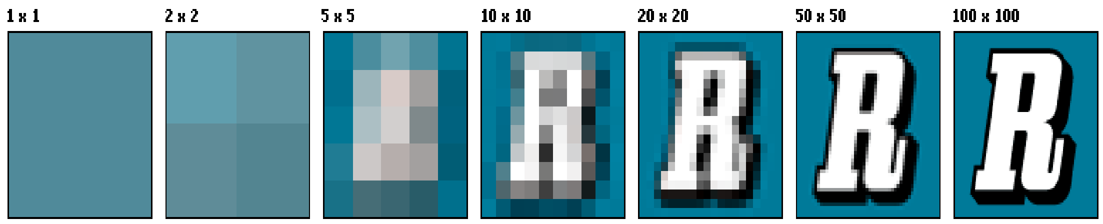


Figure (1.6) : Image Resolution .

- An image that is 2048 pixels in width and 1536 pixels in height has a total of 2048×1536 = 3,145,728 pixels or 3.1 megapixels. One could refer to it as 2048 by 1536 or a 3.1-megapixel image.

1.8 Image brightness Adaption

Brightness is intensity of light in simple words. Adaptation basically is "getting used to" and be comfortable with it. So conceptually brightness adaption is basically "getting used to changes in brightness/ changes intensity of light". A simple example is when you go out into light from

darkness you take some time to "get used" to the brightness outside and feel comfortable. This is what exactly brightness adaption is.
In image we observe many brightness levels and the vision system can adapt to a wide range. If the **mean value** of the pixels inside the image is around Zero gray level then the brightness is low and the images dark, but for mean value near the 255 then the image is light. If fewer gray levels are used, we observe false contours (منحني) bogus lines resulting from gradually changing light intensity not being accurately represented.

1.9 Image Representation

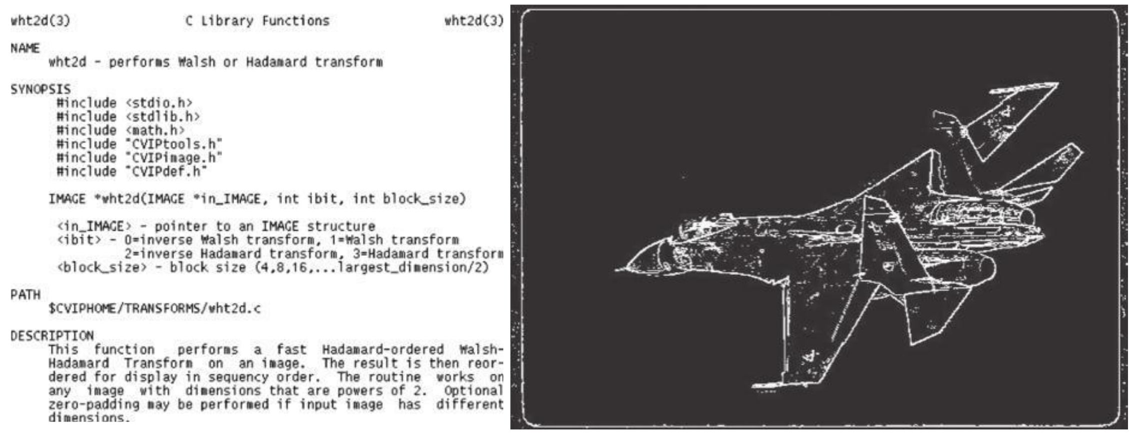
We have seen that the human visual system (HVS) receives an input image as a collection of spatially distributed light energy; this is form is called an optical image. Optical images are the type we deal with every day –cameras captures them, monitors display them, and we see them [we know that these optical images are represented as video information in the form of analog electrical signals and have seen how these are sampled to generate the digital image I(r , c).
The digital image I (r, c) is represented as a two- dimensional array of data, where each pixel value corresponds to the brightness of the image at the point (r, c). in linear algebra terms , a two-dimensional array like our image model I(r, c) is referred to as a matrix , and one row (or column) is called a vector. The image types we will consider are:

1. Binary Image

Binary images are the simplest type of images and can take one of two values, typically black and white, or '0' and '1'. A binary image is referred to as a 1 bit/pixel image because it takes only 1 binary digit to represent each pixel.

These types of images are most frequently used in computer vision application where the only information required for the task is general shapes, or outlines information. For example, to position a robotics gripper to grasp (يمسك) an object or in optical character recognition (OCR).

Binary images are often created from gray-scale images via a threshold value, those values above it are turned white (‘1’), and those below it are turned black (‘0’).



```
wht2d(3)              C Library Functions              wht2d(3)

NAME
     wht2d - performs Walsh or Hadamard transform

SYNOPSIS
      #include <stdio.h>
      #include <stdlib.h>
      #include <math.h>
      #include "CVIPtools.h"
      #include "CVIPimage.h"
      #include "CVIPdef.h"

     IMAGE *wht2d(IMAGE *in_IMAGE, int ibit, int block_size)

      <in_IMAGE> - pointer to an IMAGE structure
      <ibit> - 0=inverse Walsh transform, 1=Walsh transform
               2=inverse Hadamard transform, 3=Hadamard transform
      <block_size> - block size (4,8,16,...largest_dimension/2)

PATH
     $CVIPHOME/TRANSFORMS/wht2d.c

DESCRIPTION
     This  function   performs  a  fast  Hadamard-ordered  Walsh-
     Hadamard  Transform  on  an image.  The result is then reor-
     dered for display in sequency order.  The routine  works  on
     any  image  with  dimensions that are powers of 2.  Optional
     zero-padding may be performed if input image  has  different
     dimensions.
```

Figure (1.7) Binary Images.

2. <u>Gray Scale Image</u>

Gray _scale images are referred to as monochrome, or one-color image. They contain brightness information only, no color information. The number of different brightness level available. The typical image contains 8 bit/ pixel (data, which allows us to have (0-255) different brightness (gray) levels. The 8 bit representation is typically due to the fact that the byte, which corresponds to 8-bit of data, is the standard small unit in the world of digital computer.

Figure (1.8): Gray Scale Images.

3. <u>Color Image</u>

Color image can be modeled as three band monochrome image data, where each band of the data corresponds to a different color.

Figure (1.8) : Color Images.

The actual information stored in the digital image data is brightness information in each spectral band. When the image is displayed, the corresponding brightness information is displayed on the screen by picture elements that emit light energy corresponding to that particular color.

Typical color images are represented as red, green, and blue or RGB images .using the 8-bit monochrome standard as a model, the corresponding color image would have 24 bit/pixel – 8 bit for each color bands (red, green and blue). The following figure represent typical RGB color image.

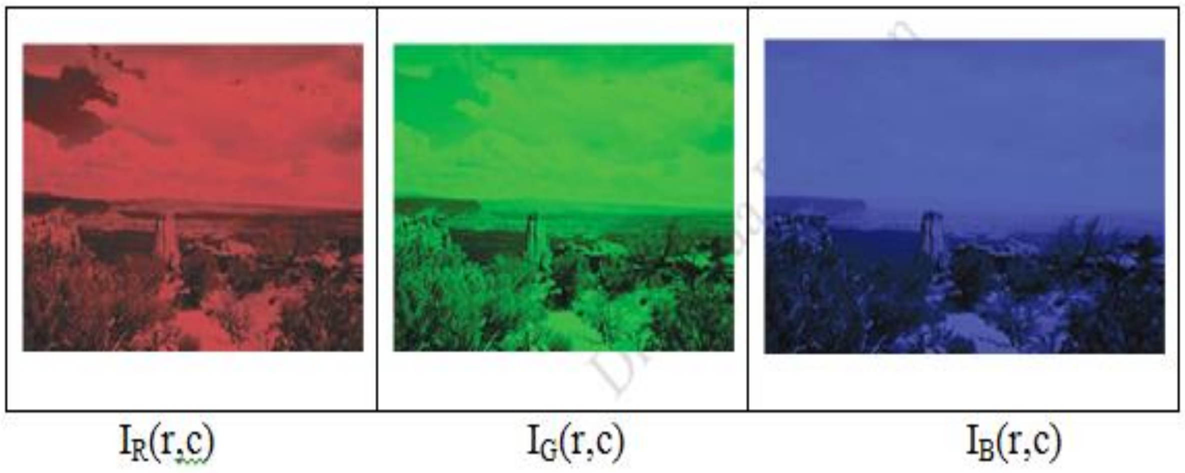


Figure (1.9) : Typical RGB color image can be thought as three separate images $I_R(r,c)$,$I_G(r,c)$,$I_B(r,c)$ [1]

In Figure (1.10 a) we see a representation of a typical RGB color image, and Figure (1.10 b) illustrate that in addition to referring to arrow or column as a vector, we can refer to a single pixel red ,green, and blue values as a color pixel vector –(R,G,B).

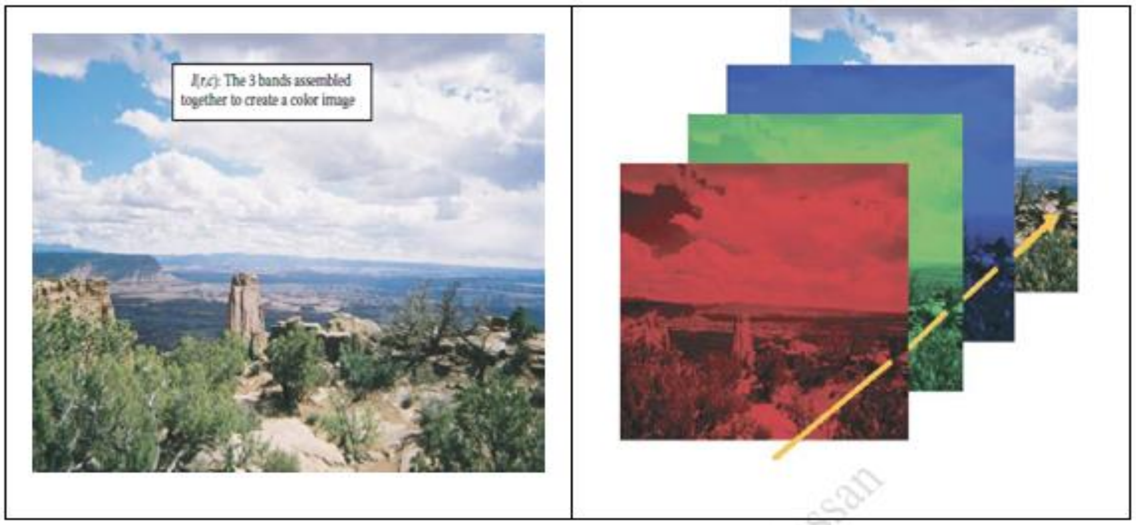

I(r,c): The 3 bands assembled together to create a color image

Figure (1.10) :A color pixel vector consists of the red, green and blue pixel values (R, G, B) at one given row/column pixel coordinate(r , c) [1].

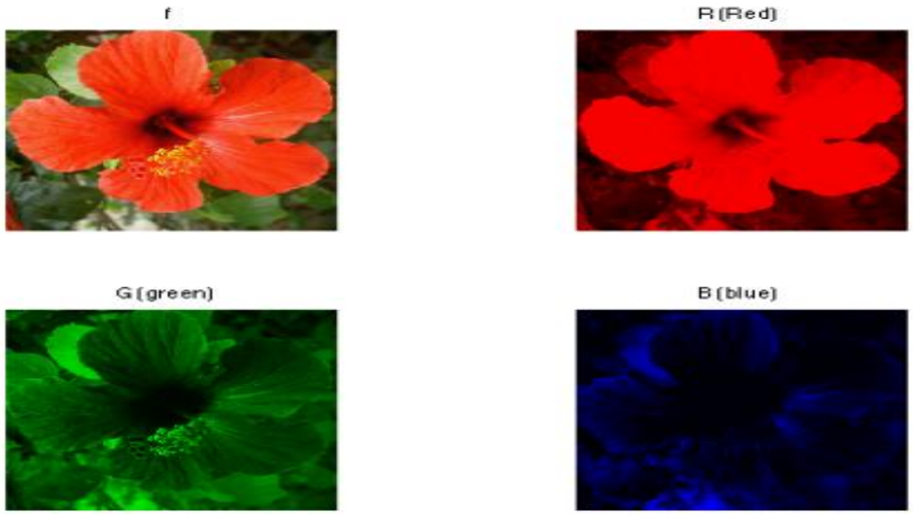


Figure (1.11) :A color pixel vector consists of the red, green and blue .

For many applications, RGB color information is transformed into mathematical space that that decouples the brightness information from the color information.

The hue/saturation /lightness (HSL) color transform allows us to describe colors in terms that we can more readily understand.

The lightness is the brightness of the color, and the hue is what we normally think of as “color” and the hue (ex: green, blue, red, and orange). The saturation is a measure of how much white is in the color (ex: Pink is red with more white, so it is less saturated than a pure red).
[Most people relate to this method for describing color].

<u>Example:</u> “a deep, bright orange” would have a large intensity (“bright”), a hue of “orange”, and a high value of saturation (“deep”).we can picture this color in our minds, but if we defined this color in terms of its RGB components, R=245, G=110 and B=20.

Modeling the color information creates a more people oriented way of describing the colors.

4. <u>Multispectral Images</u>

A multispectral image is one that captures image data at specific frequencies across the electromagnetic spectrum. Multispectral images typically contain information outside the normal human perceptual range. This may include infrared (تحت الحمراء), ultraviolet (فوق البنفسجيه), X-ray, acoustic (سمعي) or radar data. Source of these types of image include satellite systems, underwater sonar systems and medical diagnostics imaging systems.

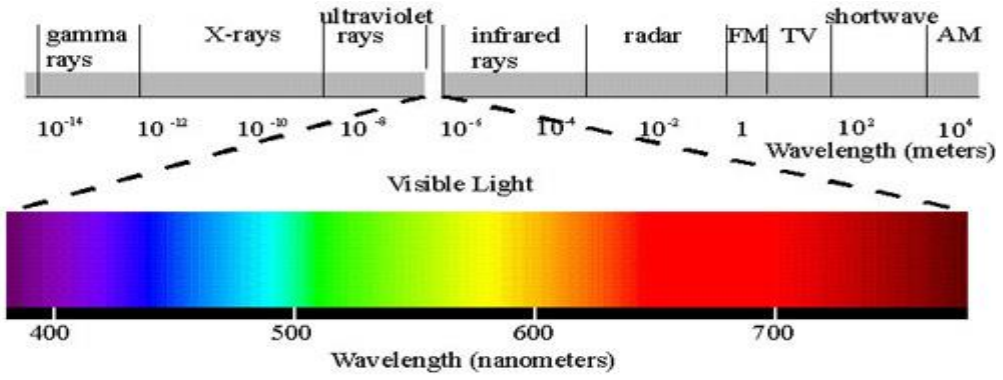


Figure (1.12): Electromagnetic spectrum.

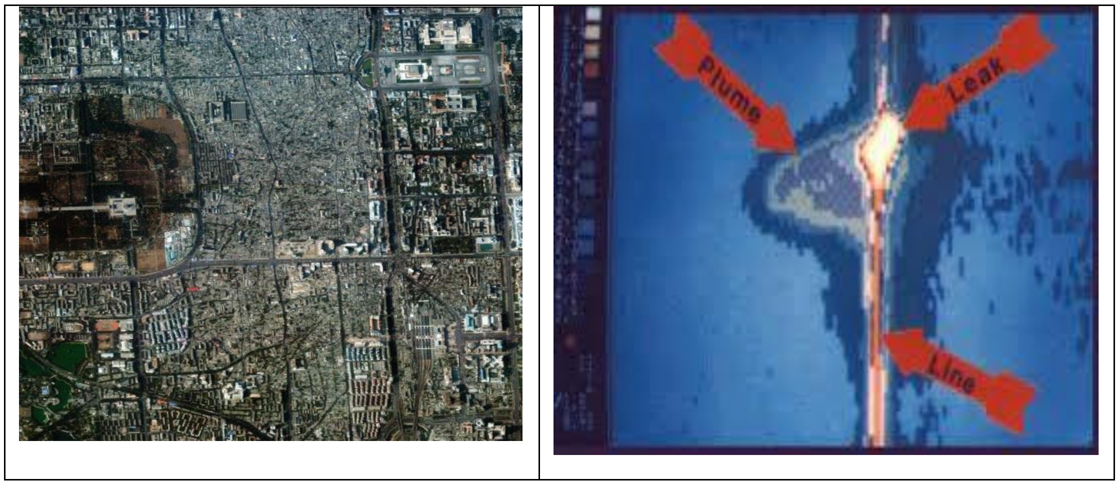


Figure (1.13): Multispectral images

1.10 <u>Computer Graphics :</u>

Computer graphics is a specialized field within that refers to the computer science realm that refers to the reproduction of visual data through the use of computer.

In computer graphics, types of image data are divided into two primarily categories:

1. **Bitmap image (or raster image):** can represented by our image model I(r, c), where we have pixel data and corresponding brightness values stored in file format.
2. **Vector images**: refer to the methods of representing lines, curves shapes by storing only the key points. These key points are sufficient to define the shapes, and the process of turing theses into an image is called rending after the image has been rendered, it can be thought of as being in bit map format where each pixel has specific values associated with it.

What is the difference between vector and raster graphics?

- Raster graphics are composed of pixels, while vector graphics are composed of paths.
- A raster graphic, such as a gif or jpeg, is an array of pixels of various colors, which together form an image. A vector graphic, such as an .eps file or Adobe illustrator file, is composed of paths, or lines, that are either straight or curved.
- Because vector graphics are not made of pixels, the images can be scaled to be very large without losing quality. Raster graphics, on the other hand, become "blocky," since each pixel increases in size as the image is made larger.

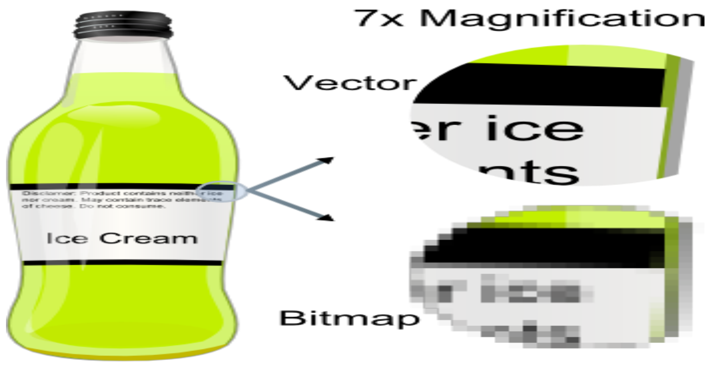


Figure (1.14): vector and bitmap image.

1.11 <u>Digital Image File Format</u>

Image file formats are standardized means of organizing and storing digital images. Image files are composed of digital data in one of these formats that can be rasterized (تنقيطه) for use on a computer display or printer. An image file format may store data in uncompressed, compressed, or vector formats. Once rasterized, an image becomes a grid of pixels, each of which has a number of bits to designate its color equal to the color depth of the device displaying it.

Why do we need so many different types of image file format?

- The short answer is that there are many different types of images and application with varying requirements.
- A more complete answer, also considers market share proprietary information, and a lack of coordination within the imaging industry.

Many image types can be converted to one of other type by easily available image conversion software. Field related to computer imaging is that computer graphics.

Most the type of file format fall into category of bitmap images. In general, these types of images contain both header information and the raw pixel data. The header information contain information regarding

1. The number of rows(height)
2. The number of columns(Width)
3. The number of bands.
4. The number of bit per pixel.
5. the file type
6. Additionally, with some of the more complex file formats, the header may contain information about the type of compression used and other necessary parameters to create the image, I(r,c).

Image File Format :

1. **BMP format (Bitmap image File Format)**

The **BMP file format**, also known as **bitmap image file** or **device independent bitmap (DIB) file format** or simply a **bitmap**, is a raster

graphics image file format used to store bitmap digital images, independently of the display device (such as a graphics adapter), especially on Microsoft Windows and OS/2 operating systems.
The BMP file format is capable of storing 2D digital images of arbitrary width, height, and resolution, both monochrome and color, in various color depths, and optionally with data compression, alpha channels, and color profiles.

2. TIFF (Tagged Image File Format)

Tagged Image File Format is one of the most popular and flexible of the current public domain raster file formats. They are used on World Wide Web (WWW). GIF files are limited to a maximum of 8 bits/pixel and allows for a type of compression called LZW. The GIF image header is 13 byte long & contains basic information.

3. JPEG (Joint Photo Graphic Experts Group)

This is the right format for those photo images which must be very small files, for example, for web sites or for email. JPG is often used on digital camera memory cards. The JPG file is wonderfully small, often compressed to perhaps only 1/10 of the size of the original data, which is a good thing when modems are involved. However, this fantastic compression efficiency comes with a high price. JPG uses lossy compression (lossy meaning "with losses to quality"). Lossy means that some image quality is lost when the JPG data is compressed and saved, and this quality can never be recovered. JPEG images compression is being used extensively on the WWW. It’s, flexible, so it can create large files with excellent image equality.

4. VIP(visualization in image processing)formats:

It is developed for the CVIP tools software, when performing temporary images are created that use floating point representation which is beyond the standard 8 bit/pixel. To represent this type of data the remapping is used, which is the process of taking original image and adding an equation to translate it to the rang (0-225).

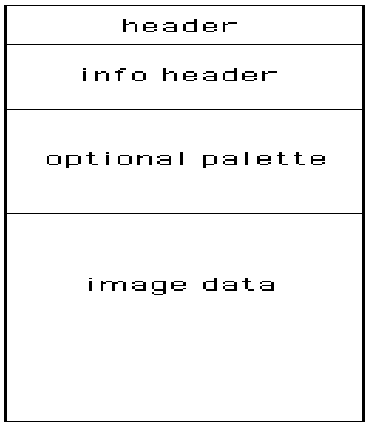
1. 12 BMP image format

Introduction

BMP files are an historic (but still commonly used) file format for the historic (but still commonly used) operating system called "Windows". BMP images can range from black and white (1 bit per pixel) up to 24 bit colour (16.7 million colours). While the images can be compressed, this is rarely used in practice and won't be discussed in detail here.

Structure

A BMP file consists of either 3 or 4 parts as shown in the following diagram



The first part is a header, this is followed by an information section, if the image is indexed colour then the palette follows, and last of all is the pixel data. Information such as the image width and height, the type of compression, the number of colours is contained in the information header.

Header

The header consists of the following fields. Note that we are assuming short int of 2 bytes, int of 4 bytes, and long int of 8 bytes.

1. Magic identifier (1+1= 2 Byte).
2. File Size in byte(4 Byte)
3. Resereved1+ Reserved2 (2+2 Byte)

14 Byte

4. Offset to image data (4 Byte)

The useful fields in this structure are the type field (should be 'BM') which is a simple check that this is likely to be a legitimate المشروعة BMP file, and the offset field which gives the number of bytes before the actual pixel data (this is relative to the start of the file).

Information

The image info data that follows is 40 bytes in length, it is described in the struct given below. The fields of most interest below are the image width and height, the number of bits per pixel (should be 1, 4, 8 or 24), the number of planes (assumed to be 1 here), and the compression type (assumed to be 0 here).

1. Header size in bytes (4 Byte)
2. Width and height of image (4 + 4 Byte)
3. Number of colour planes (2 Byte)
4. Bits per pixel (2 Byte)
5. Compression type (4 Byte)
6. Image size in bytes (4 Byte)
7. Pixels per meter (x_ resolution , y_ resolution) (4 + 4 Byte)
8. Number of colours (4 Byte)
9. Important colours (4 Byte)

40 Byte

The compression types supported by BMP are listed below:

0 - no compression

1 - 8 bit run length encoding

2 - 4 bit run length encoding

3 - RGB bitmap with mask

Chapter Two
Image Analysis

2.1 <u>Image Analysis</u>

Image analysis involves manipulating the image data to determine exactly the information necessary to help solve a computer imaging problem. This analysis is typically part of a larger process, is iterative in nature and allows us to answer application specific questions: Do we need color information? Do we need to transform the image data into the frequency domain? Do we need to segment the image to find object information? What are the important features of the image?

Image analysis is primarily **data reduction** process. As we have seen, images contain enormous amount of data, typically on the order hundreds of kilobytes or even megabytes. Often much of this information is not necessary to solve a specific computer imaging problem, so primary part of the image analysis task is to determine exactly what information is necessary. Image analysis is used both computer vision and image processing.

For computer vision, the end product is typically the extraction of high-level information for computer analysis or manipulation. This high-level information may include shape parameter to control a robotics manipulator or color and texture features to help in diagnosis of a skin tumor.

In image processing application, image analysis methods may be used to help determine the type of processing required and the specific parameters needed for that processing. For example, determine the degradation function for an image restoration procedure, developing an enhancement algorithm and determining exactly what information is visually important for image compression methods.

2.2 System Model

The image analysis process can be broken down into three primary stages:

1. Preprocessing.
2. Data Reduction.
3. Features Analysis.

1. Preprocessing:

The preprocessing algorithm, techniques and operators are used to perform initial processing that makes the primary data reduction and analysis task easier. Preprocessing is used to remove noise and eliminate irrelevant, visually unnecessary information. Noise is unwanted information that can result from the image acquisition process; other preprocessing steps might include operations related to:

- Gray –level or spatial quantization (reducing the number of bits per pixel or the image size).
- Extracting regions of interest ROI for further processing.
- Performing basic algebraic operation on image.
- Enhancing specific image features.
- Reducing data in resolution and brightness.

Preprocessing is a stage where the requirements are typically obvious and simple, such as removal of artifacts from images or eliminating of image information that is not required for the application. For example, in one application we needed to eliminate borders from the images that have been digitized from film. Another example of preprocessing step involves a robotics gripper that needs to pick and place an object ; for this we reduce a gray-level image to binary (two-valued) image that contains all the information necessary to discern the object ‘s outlines.

2. Data Reduction:

Involves either reducing the data in the spatial domain or transforming it into another domain called the frequency domain, and then extracting features for the analysis process.

3. Features Analysis:

The features extracted by the data reduction process are examine and evaluated for their use in the application.

After preprocessing we can perform segmentation on the image in the spatial domain or convert it into the frequency domain via a mathematical transform. After these processes we may choose to filter the image. This filtering process further reduces the data and allows us to extract the feature that we may require for analysis.

Figure (2.1) illustrates the system model of image analysis

Image Analysis
System Model

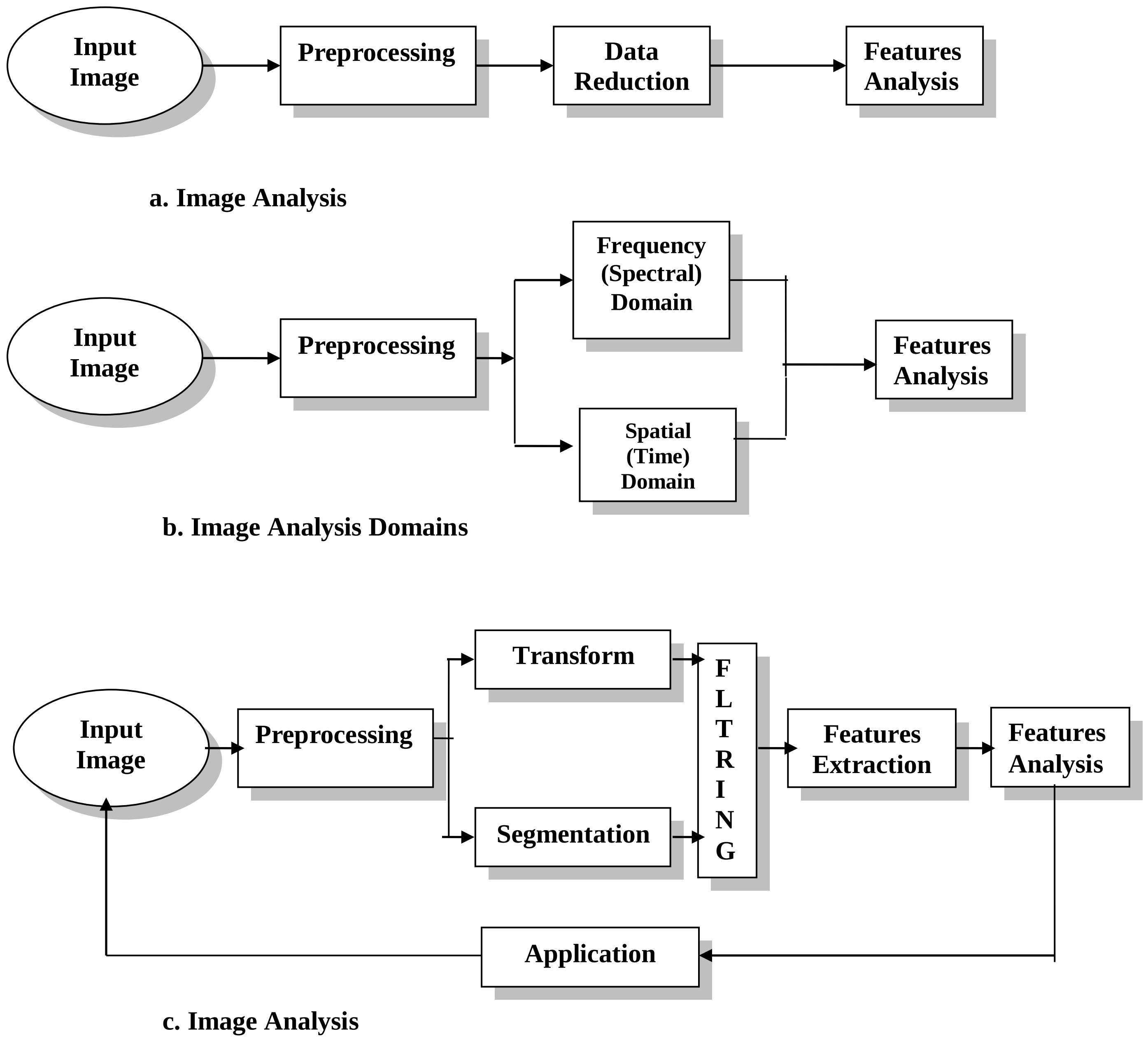


Figure (2.1): System model of image analysis

2.3 Region –of-Interest Image Geometry

Often, for image analysis we want to investigate more closely a specific area within the image, called region of interest (ROI). To do this we need operation that modifies the spatial coordinates of the image, and these are categorized as image geometry operations. The image geometry operations discussed here include:

Crop (بروز) , Zoom, enlarge , shrink, translate and rotate.

The image crop process is the process of selecting a small portion of the image, a sub image and cutting it away from the rest of the image. After we have cropped a sub image from the original image we can zoom in on it by enlarge it. The zoom process can be done in numerous ways:

1. **Zero-Order Hold.**
2. **First _Order Hold.**
3. **Convolution.**

2.3.1 Zoom Operation

1. Zero-Order hold: is performed by repeating previous pixel values, thus creating a blocky effect as in the following figure:

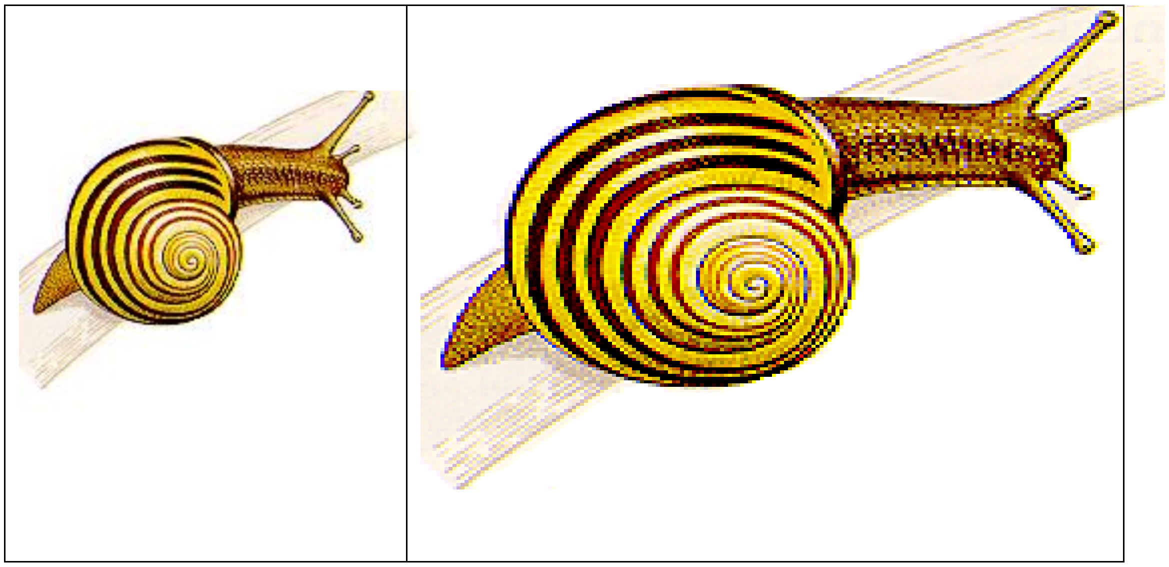


Figure (2.2): Zero _Order Hold Method

1. **First _Order Hold:** is performed by finding linear interpolation between a adjacent pixels, i.e., finding the average value between two pixels and use that as the pixel value between those two, we can do this for the rows first as follows:

Original Image Array **Image with Rows Expanded**

$$\begin{pmatrix} 8 & 4 & 8 \\ 4 & 8 & 4 \\ 8 & 2 & 8 \end{pmatrix} \qquad \begin{pmatrix} 8 & 6 & 4 & 6 & 8 \\ 4 & 6 & 8 & 6 & 4 \\ 8 & 5 & 2 & 5 & 8 \end{pmatrix}$$

The first two pixels in the first row are averaged (8+4)/2=6, and this number is inserted between those two pixels. This is done for every pixel pair in each row.

Next, take result and expanded the columns in the same way as follows:

Image with rows and columns expanded

$$\begin{pmatrix} 8 & 6 & 4 & 6 & 8 \\ 6 & 6 & 6 & 6 & 6 \\ 4 & 6 & 8 & 6 & 4 \\ 6 & 5.5 & 5 & 5.5 & 6 \\ 8 & 5 & 2 & 5 & 8 \end{pmatrix}$$

This method allows us to enlarge an N×N sized image to a size of (2N-1) × (2N-1) and be repeated as desired.

3- Convolution: this process requires a mathematical process to enlarge an image. This method required two steps:

1. Extend the image by adding rows and columns of zeros between the existing rows and columns.

2. Perform the convolution.

The image is extended as follows**:**

Original Image Array

$$\begin{pmatrix} 3 & 5 & 7 \\ 2 & 7 & 6 \\ 3 & 4 & 9 \end{pmatrix}$$

Image extended with zeros

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 5 & 0 & 7 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 7 & 0 & 6 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 4 & 0 & 9 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Next, we use convolution mask, which is slide a cross the extended image, and perform simple arithmetic operation at each pixel location

Convolution mask for first –order hold

$$\begin{pmatrix} \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \\ \frac{1}{2} & 1 & \frac{1}{2} \\ \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \end{pmatrix}$$

The convolution process requires us to overlay the mask on the image, multiply the coincident (متقابله) values and sum all these results. This is equivalent to finding the vector inner product of the mask with underlying sub image. The vector inner product is found by overlaying mask on sub image. Multiplying coincident terms, and summing the resulting products.

For example, if we put the mask over the upper-left corner of the image, we obtain (from right to left, and top to bottom):

$1/4(0) + 1/2(0) + 1/4(0) + 1/2(0) + 1(3) + 1/2(0) + 1/4(0) + 1/2(0) + 1/4(0) = 3$

Note that the existing image values do not change. The next step is to slide the mask over by on pixel and repeat the process, as follows:

$1/4(0) + 1/2(0) + 1/4(0) + 1/2(3) + 1(0) + 1/2(5) + 1/4(0) + 1/2(0) + 1/4(0) = 4$

Note this is the average of the two existing neighbors. This process continues until we get to the end of the row, each time placing the result of the operation in the location corresponding to center of the mask.

When the end of the row is reached, the mask is moved down one row, and the process is repeated row by row. This procedure has been performed on the entire image, the process of **sliding, multiplying and summing** is called convolution.

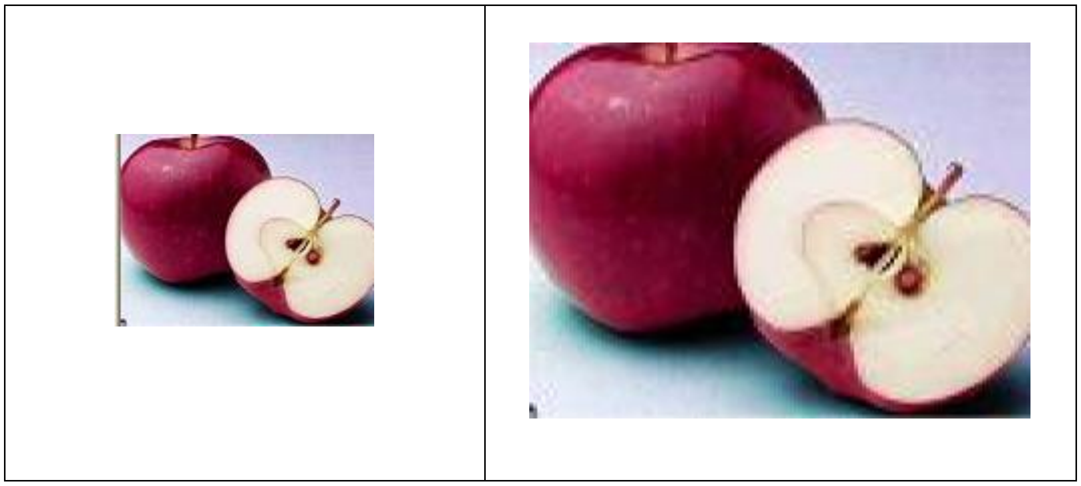
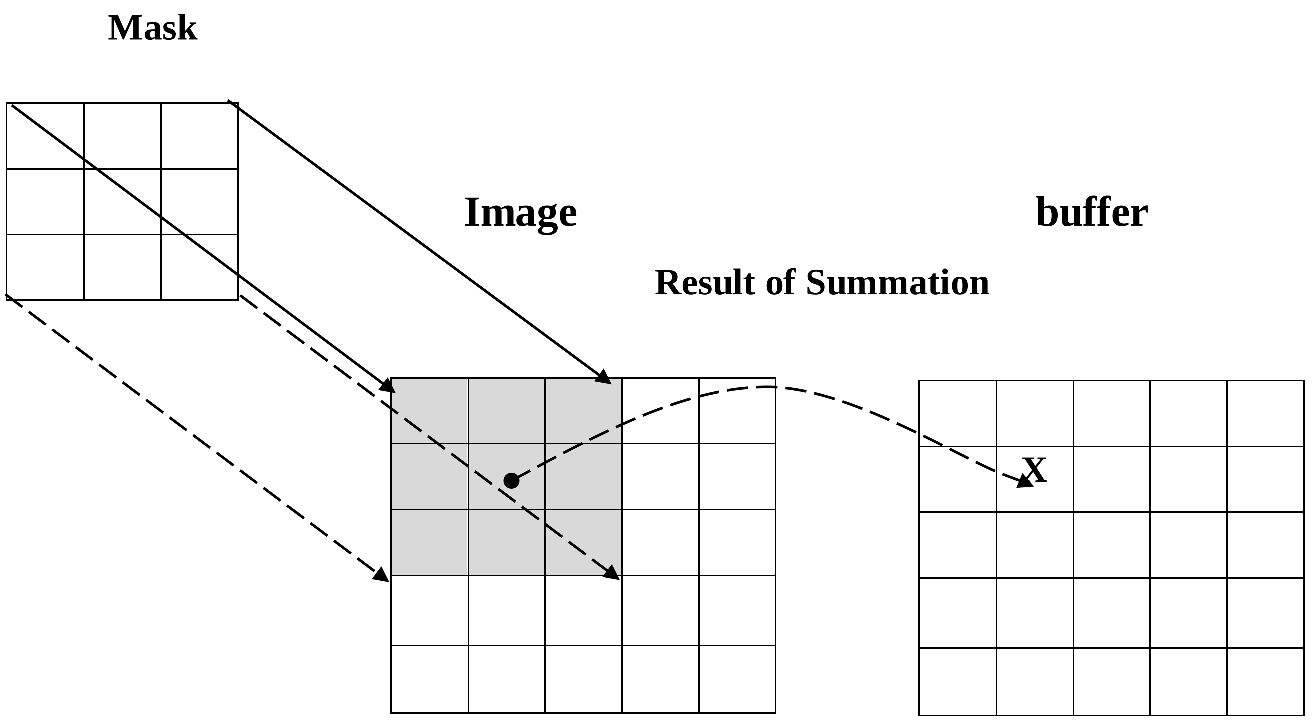


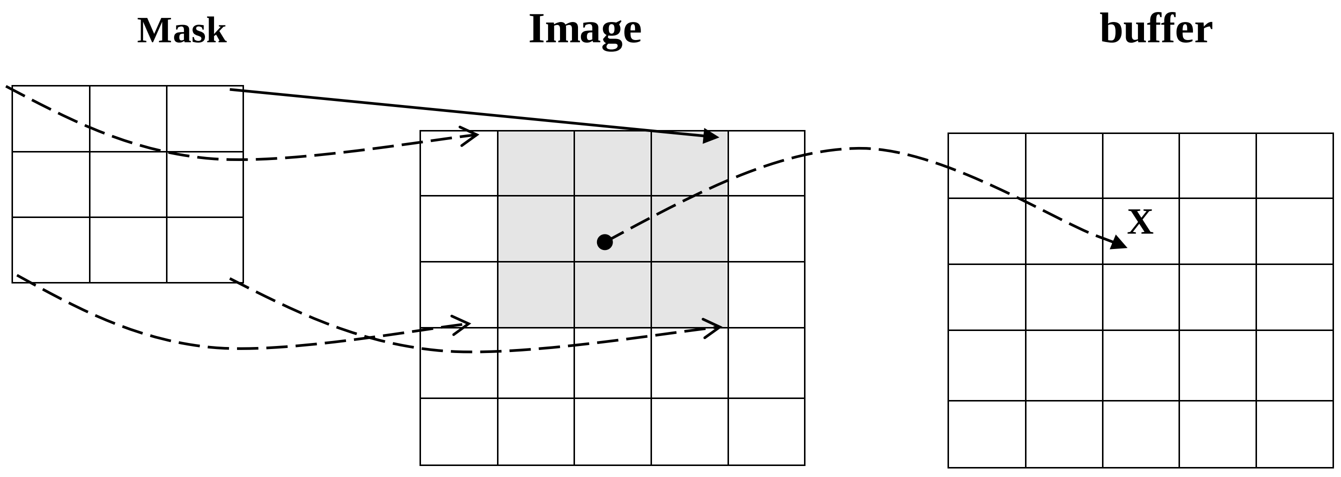
Figure (2.3): First _Order Hold Method

Note that the output image must be put in a separate image array called a buffer, so that the existing values are not overwritten during the convolution process.

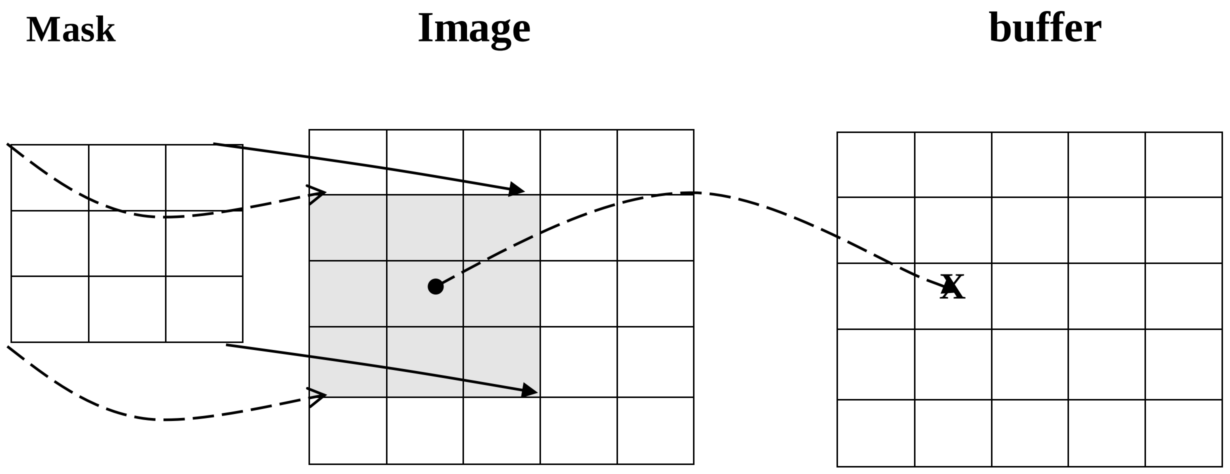
The convolution process



a. Overlay the convolution mask in the upper-left corner of the image. Multiply coincident terms, sum, and put the result into the image buffer at the location that corresponds to the masks current center, which is (r,c)=(1,1).



b. Move the mask one pixel to the right , multiply coincident terms sum , and place the new results into the buffer at the location that corresponds to the new center location of the convolution mask which is now at (r,c)=(1,2), continue to the end of the row.



c. Move the mask down one row and repeat the process until the mask is convolved with the entire image. Note that we lose the outer row(s) and columns(s).

Why we use this convolution method when it require, so many more calculation than the basic averaging of the neighbors method?

The answer is that many computer boards can perform convolution in hardware, which is generally very fast, typically much faster than applying a faster algorithm in software. Not, only first-order hold be performed via convolution, but zero-order hold can also achieved by extending the image with zeros and using the following convolution mask.

Zero-order hold convolution mask

$$\begin{pmatrix} \mathbf{1} & \mathbf{1} \\ \mathbf{1} & \mathbf{1} \end{pmatrix}$$

Note that for this mask we will need to put the result in the pixel location corresponding to the lower-right corner because there is no center pixel.

These methods will only allows us to enlarge an image by a factor of (2N-1), but what if we want to enlarge an image by something other than a factor of (2N-1)?

To do this we need to apply a more general method. We take two adjacent values and linearly interpolate more than one value between them. This is done by define an enlargement number k and then following this process:

1. Subtract the result by k.
2. Divide the result by k.
3. Add the result to the smaller value, and keep adding the result from the second step in a running total until all (k-1) intermediate pixel locations are filled.

<u>Example:</u> we want to enlarge an image to three times its original size, and we have two adjacent pixel values 125 and 140.

1. Find the difference between the two values,   140-125 =15.

2. The desired enlargement is k=3, so we get 15/3=5.
3. next determine how many intermediate pixel values .we need :
   K-1=3-1=2. The two pixel values between the 125 and 140 are
   125+5=130 and 125+2*5 = 135.

- We do this for every pair of adjacent pixels .first along the rows and then along the columns. This will allows us to enlarge the image by any factor of K (N-1) +1 where K is an integer and N×N is the image size.
- To process opposite to enlarging an image is shrinking. This process is done by reducing the amount of data that need to be processed.
- Two other operations of interest image geometry are: Translation and Rotation. These processes may be performed for many application specific reasons, for example to align an image with a known template in pattern matching process or make certain image details easer to see.

2.3.2 Image Shrinking

The process opposite to enlarge an image is shrinking it. This is not typically done to examine a ROI more closely but to reduce the amount of data that needs to be processed.

Image shrinking is accomplished by taking groups of pixels that are spatially adjacent and mapping them to one pixel. This can be done in one of three ways:

1. Averaging
2. Median
3. Decimation.

For averaging method, we take all the pixels in each group (for example 2× 2 block of pixels) and find the average gray level by summing the values and dividing by the number of pixels in the group. Within the second method, median, we sort all the pixels values from lowest to highest and then select the middle value. The third approach decimation, also known as sub sampling, entails simply eliminating some of the data. For example to reduce the image by a factor of two, we simply take every other row and column and delete them.

<u>EXAMPLE</u>

For the below 2 × 2 block of pixels, shrink this block using the three methods of shrinking.

$$\begin{bmatrix} 6 & 8 \\ 1 & 9 \end{bmatrix}$$

<u>Averaging</u>	**<u>median</u>**	**<u>decimation</u>**
6+8+1+9=24/4=6	$1,\underline{6,8},9 = (6+8)/2=7$	eliminate the first row and the first column and the result is 9

2.3.3 Translation and Rotation

Two other operation of interest for the ROI image geometry are translation and rotation. These processes may be performed for many application-specific reasons, for example to align an image with a known template in a pattern matching process or to make certain image details easier to see. The translation process can be done with the following equations:

$$r' = r + r_o$$

$$c' = c + c_o$$

where **r'** and **c'** are the new coordinates, **r** and **c** are the original coordinates, and **r_0** and **c_0** are the **distance to move, or translate, the image.**

The rotation process requires the use of these equations:

$$r^{\wedge} = r(\cos\Theta) + c(\sin\Theta)$$

$$c^{\wedge} = -r(\sin\Theta) + c(\cos\Theta)$$

Where, $r^{\wedge}$ and $c^{\wedge}$ are the new coordinate, r and c are the original coordinates, and Θ is the angle to rotate the image. **Θ is defined in a clockwise direction from the horizontal axis at the image origin in the upper- left corner.**

The **rotation and translation** process can be combined into one set of equations:

$$r^{\wedge'} = (r + r_0)(\cos\Theta) + (c + c_0)(\sin\Theta)$$

$$c^{\wedge'} = -(r + r_0)(\sin\Theta) + (c + c_0)(\cos\Theta)$$

where $r^{\wedge'}$, $c^{\wedge'}$ are the new coordinates and r, c, r_0, c_0 and Θ are previously defined.

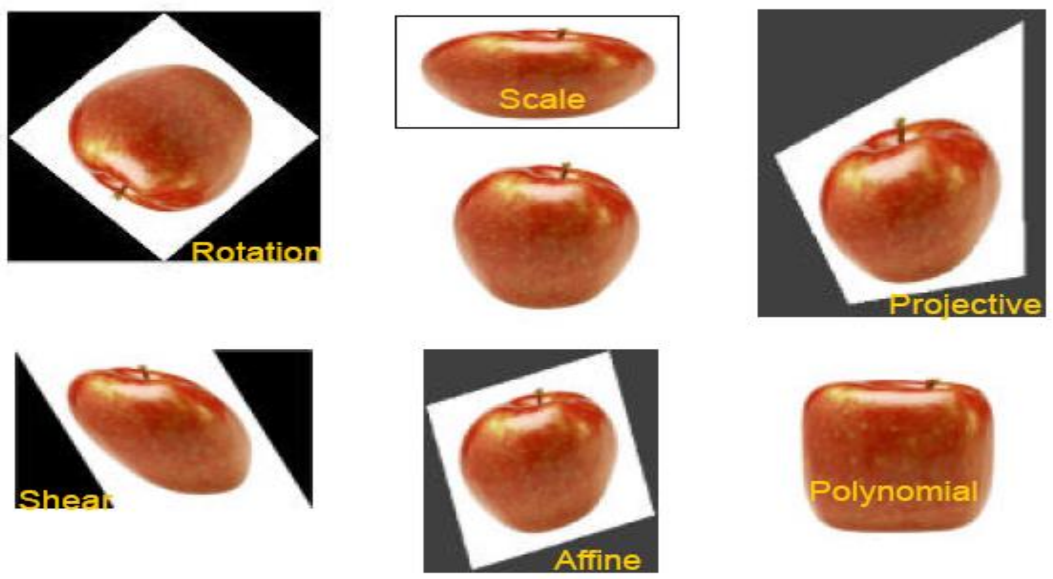


Figure (2.4): Image with Geometry Transformation

2.4 <u>Image Algebra</u>

There are two primary categories of algebraic operations applied to image:

1. Arithmetic operations.
2. Logic operations.

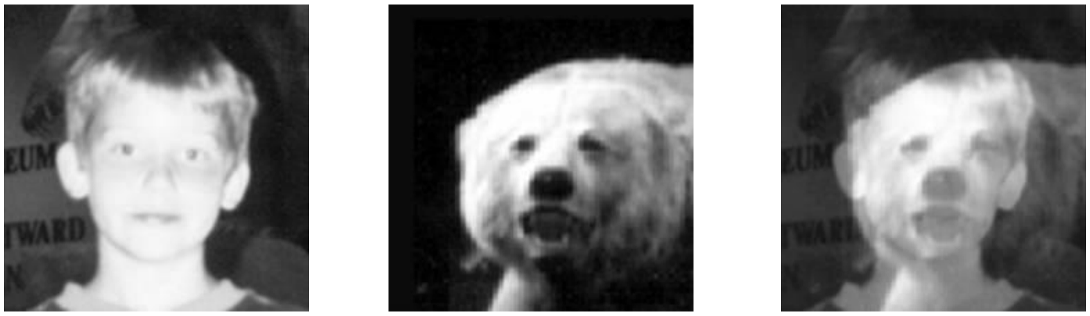
Addition, subtraction, division and multiplications comprise the arithmetic operations, while AND, OR and NOT makeup the logic operations. These operations which require only one image, and are done on a pixel –by-pixel basis.

To apply the arithmetic operations to two images, we simply operate on corresponding pixel values. For example to add image I_1 and I_2 to create I_3:

$$\underset{\mathbf{I_1}}{\begin{pmatrix} 3 & 4 & 7 \\ 3 & 4 & 5 \\ 2 & 4 & 6 \end{pmatrix}} + \underset{\mathbf{I_2}}{\begin{pmatrix} 6 & 6 & 6 \\ 4 & 2 & 6 \\ 3 & 5 & 5 \end{pmatrix}} = \underset{\mathbf{I_3}}{\begin{pmatrix} 3+6 & 4+6 & 7+6 \\ 3+4 & 4+2 & 5+6 \\ 2+3 & 4+5 & 6+5 \end{pmatrix}} = \begin{pmatrix} 9 & 10 & 13 \\ 7 & 6 & 11 \\ 5 & 9 & 11 \end{pmatrix}$$

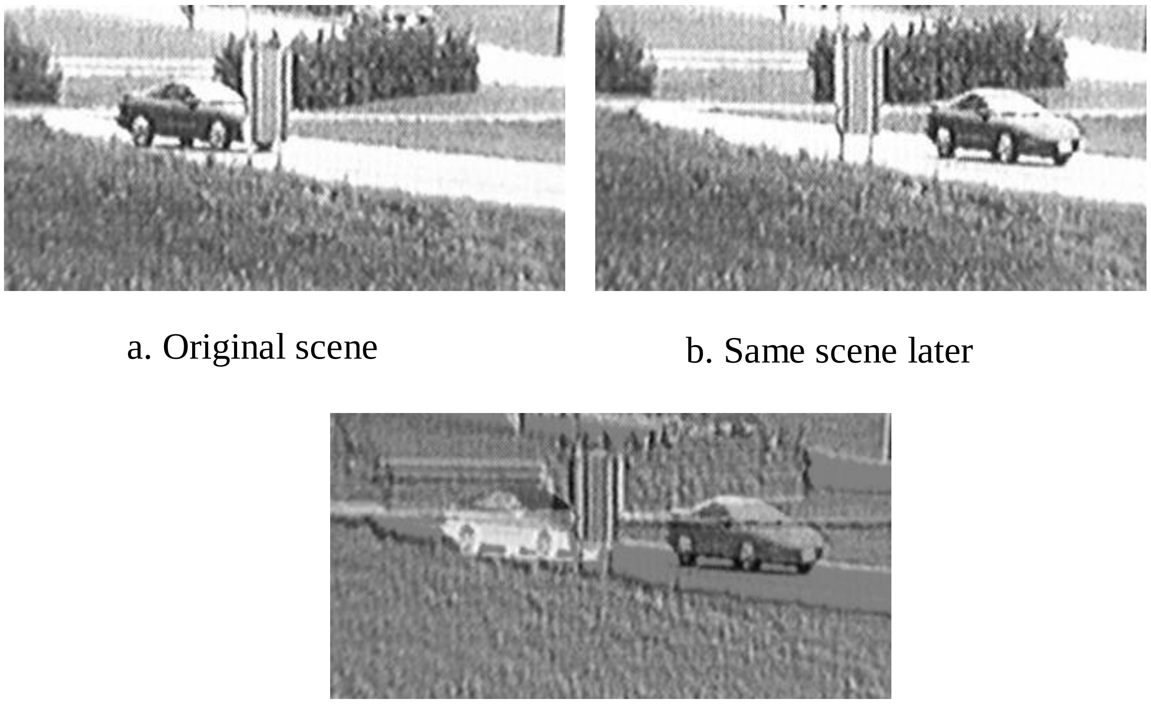
- Addition is used to combine the information in two images. Applications include development of image restoration algorithm for molding additive noise, and special effects, such as image morphing in motion pictures.
- Subtraction of two images is often used to detect motion, consider the case where nothing has changed in a sense; the image resulting from subtraction of two sequential images is filled with zero-a black image. If something has moved in the scene, subtraction produces a nonzero result at the location of movement. Applications include Object

tracking , Medical imaging, Law enforcement and Military applications



a. First Original image b. Second Original c. Addition of two images

Figure (2.5): Image Addition.



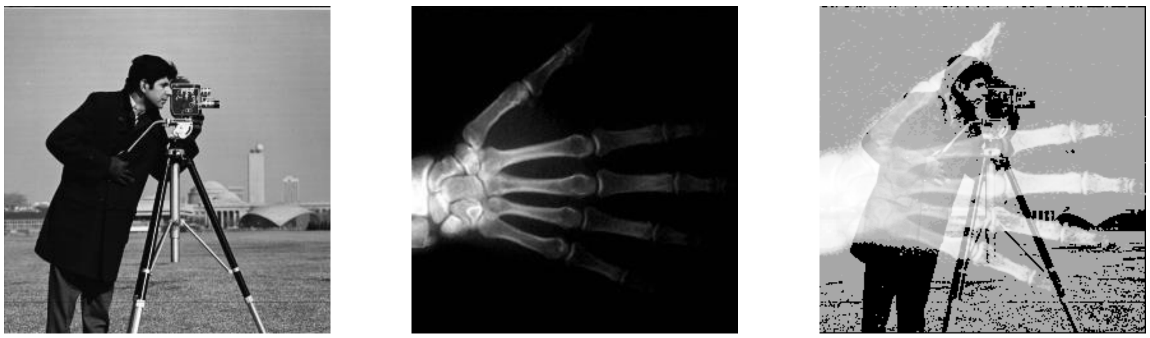
a. Original scene b. Same scene later

c. Subtraction of scene a from scene b

Figure (2.6): Image Subtraction.

- Multiplication and Division are used to adjust the brightness of an image. One image typically consists of a constant number greater than one. Multiplication of the pixel values by a number greater than one will darken the image (Brightness adjustment is often used as a processing step in image enhancement).

The logic operations AND, OR and NOT form a complete set, meaning that any other logic operation (XOR, NOR, NAND) can be created by a combination of these basic elements. They operate in a bit-wise fashion on pixel data.



a. Cameraman image **b.** X-ray image of hand c. Multiplication of two images

Figure (2.7): Image Multiplication.

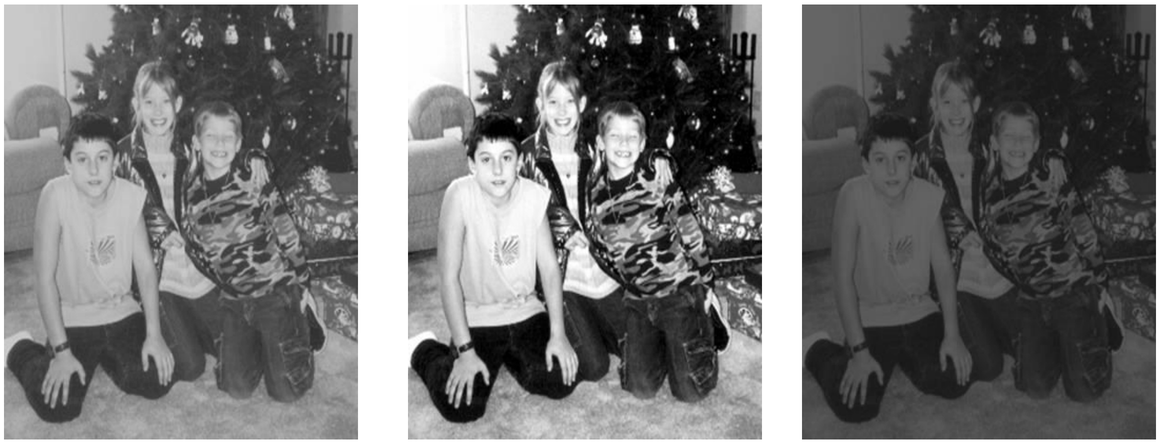

a. Original image b. Image divided by value<1 c. Image divided by value >1

Figure (2.8): Image Division

- Logical operations apply only ***to binary images***, whereas arithmetic operations apply to multi-valued pixels. Logical operations are basic tools in binary image processing, where they are used for tasks such as ***masking***, ***feature detection***, and ***shape analysis***. Logical operations on entire image are performed pixel – by – pixel. Because the AND operation of two binary variables is 1 only when both variables are 1, the result at any location in a resulting AND image is 1 only if the corresponding pixels in the two input images are 1. As logical operation involve only one pixel location at a time, they can be done in place, as in the case of arithmetic operations. The XOR (exclusive OR) operation yields a 1 when one or other pixel (but not both) is 1, and it yields a 0 otherwise. The operation is unlike the OR operation, which is 1, when one or the other pixel is 1, or both pixels are 1.

	AND				OR				XOR			
input1	1	1	0	0	1	1	0	0	1	1	0	0
Input2	1	0	1	0	1	0	1	0	1	0	1	0
output	1	0	0	0	1	1	1	0	0	1	1	0

- Logical AND & OR operations are useful for the ***masking and compositing*** of images. For example, if we compute the AND of a binary image with some other image, then pixels for which the corresponding value in the binary image is 1 will be preserved, but pixels for which the corresponding binary value is 0 will be set to 0 (erased) . Thus the binary image acts as a "***mask***" that ***removes*** information from certain parts of the image.
- On the other hand, if we compute the OR of a binary image with some other image , the pixels for which the corresponding value in the binary image is 0 will be ***preserved***, but pixels for which the corresponding binary value is 1, will be set to 1 (cleared).

So, masking is a simple method to extract a region of interest (ROI) from an image

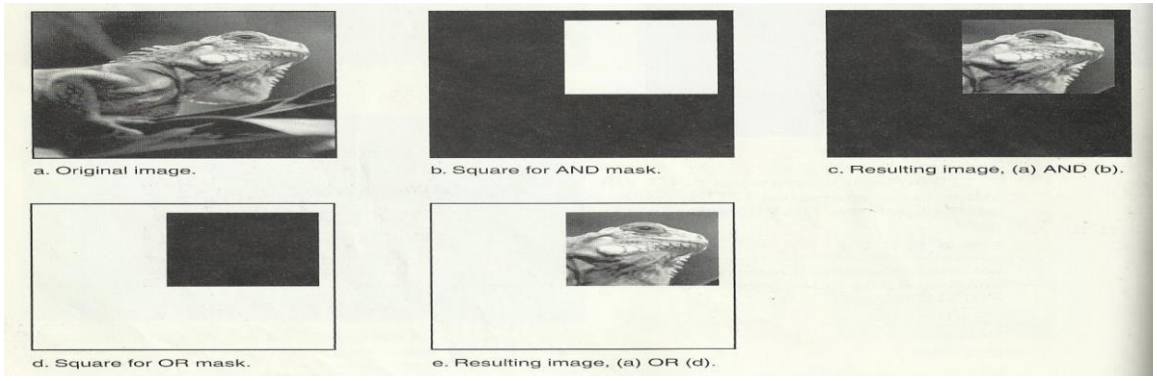

a. Original image.
b. Square for AND mask.
c. Resulting image, (a) AND (b).
d. Square for OR mask.
e. Resulting image, (a) OR (d).

Figure (2.9): Image masking.

In addition to masking, logical operation can be used in feature detection. Logical operation can be used to compare between two images, as shown below:

AND ^

This operation can be used to find the similarity white regions of two different images (it required +two images).

g (x,y) = a (x,y) ^ b (x,y)

Exclusive OR $\otimes$

This operator can be used to find the differences between white regions of two different images (it requires two images).

g (x,y) = a (x,y) $\otimes$ b (x,y)

NOT

NOT operation can be performed on grey-level images, it's applied on only one image, and the result of this operation is the ***negative*** of the original image.

g (x,y) = 255- f (x,y)

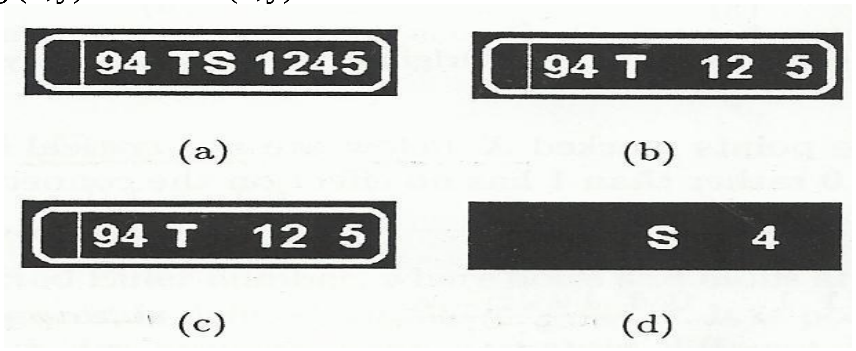


Figure 2.10: a) input image a(x,y) **b)** input image b(x,y) **c)** a(x,y) ^ b(x,y) **d)** a(x,y) $\otimes$ b(x,y)

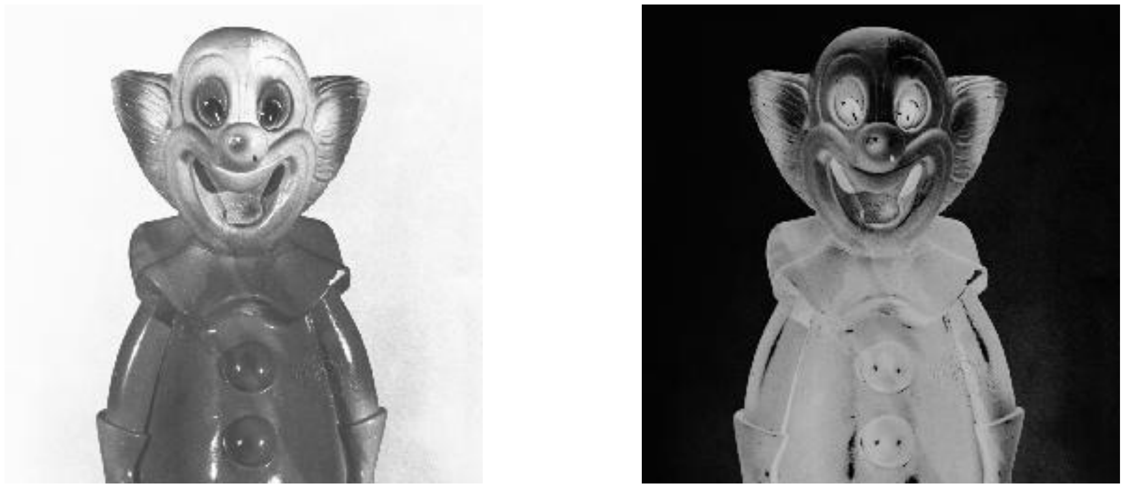

a. Original image b. Image after NOT operation.

Figure (2.11): Complement Image.

<u>Example:</u> A logic AND is performed on two images, suppose the two corresponding pixel values are **$(111)_{10}$** is one image and **$(88)_{10}$** in the second image. The corresponding bit strings are:

$$
\begin{array}{lcr}
(111)_{10} & \longrightarrow & 01101111_2 \\
 & \text{AND} & \\
(88)_{10} & \longrightarrow & \underline{01011000_2} \\
 & & 01001000
\end{array}
$$

2.5 <u>Image Restoration:</u>

Image restoration methods are used to improve the appearance of an image by application of a restoration process that use mathematical model for image degradation.

Example of the type of degradation:

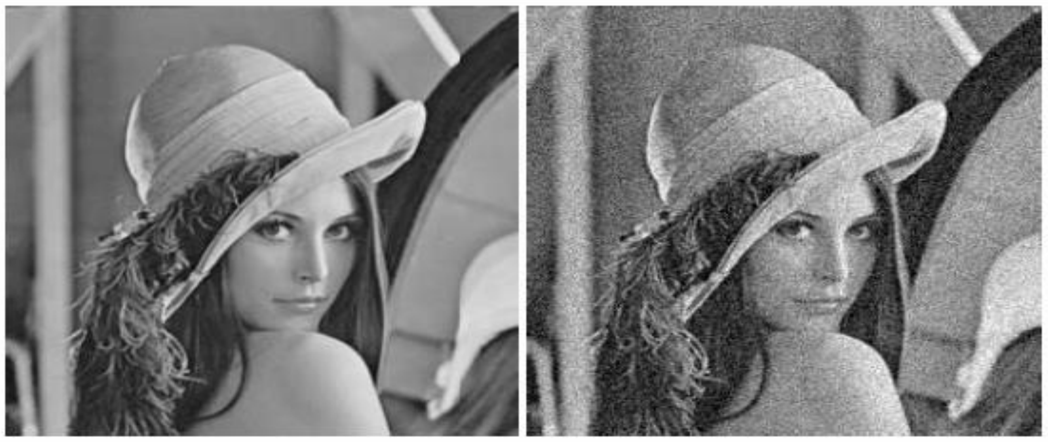
1. Blurring caused by motion or atmospheric disturbance.
2. Geometrics distortion caused by imperfect lenses.
3. Superimposed interface patterns caused by mechanical systems.
4. Noise from electronic source.

2.5.1 <u>What is noise?</u>

Noise is any undesired information that contaminates an image. Noise appears in image from a variety of source. The digital image a acquisition process, which converts an optical image into a continuous electrical signal that is then sampled is the primary process by which noise appears in digital images.

At every step in the process there are fluctuations (تذبذب) caused by natural phenomena (ظاهره) that add a random value to exact brightness value for a given pixel. In typical image the noise can be modeled with one of the following distribution:

1. Gaussian ("normal") distribution.
2. Uniform distribution.
3. Salt _and _pepper distribution.


(a) original (b) with Gaussian noise (variance=0.005)

(c) with salt and Pepper noise (p=1%)

Figure (2.12): Image Noise.

2.5.2 <u>Noise Removal using Spatial Filters:</u>

Spatial filtering is typically done for:

1. Remove various types of noise in digital images.
2. Perform some type of image enhancement.

[These filters are called spatial filter to distinguish them from frequency domain filter].

The three types of filters are:

1. Mean filters
2. Median filters (order filter)
3. Enhancement filters

Mean and median filters are used primarily to conceal or remove noise, although they may also be used for special applications. For instance, a mean filter adds "softer" look to an image. The enhancement filter high lights edges and details within the image.

Spatial filters are implemented with convolution masks. Because convolution mask operation provides a result that is weighted sum of the values of a pixel and its neighbours, it is called a linear filter.

Overall effects the convolution mask can be predicated based on the general pattern. For example:

- If the coefficients of the mask sum to one, the average brightness of the image will be retained.
- If the coefficients of the mask sum to zero, the average brightness will be lost and will return a dark image.
- If the coefficients of the mask are alternatively positive and negative, the mask is a filter that returns edge information only.
- If the coefficients of the mask are all positive, it is a filter that will blur the image.

The mean filters, are essentially averaging filter. They operate on local groups of pixel called neighbourhoods and replace the centre pixel with an average of the pixels in this neighbourhood. This replacement is done with a convolution mask such as the following 3×3 mask

Arithmetic mean filter smoothing or low-pass filter.

$$\begin{pmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{pmatrix}$$

Note that the coefficient of this mask sum to one, so the image brightness will be retained, and the coefficients are all positive, so it will tend to blur the image. This type of mean filter smooth out local variations within an image, so it essentially a low pass filter. **So a low filter can be used to attenuate image noise that is composed primarily of high frequencies components.**

a. Original image

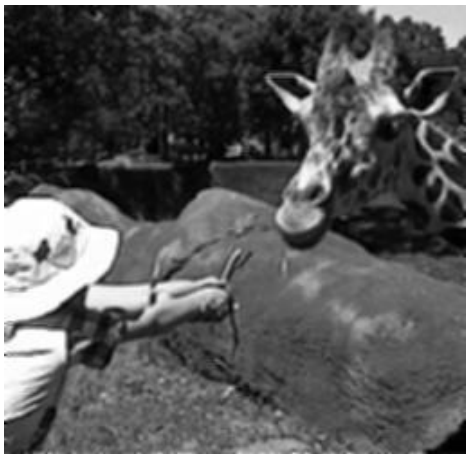

b. Mean filtered image

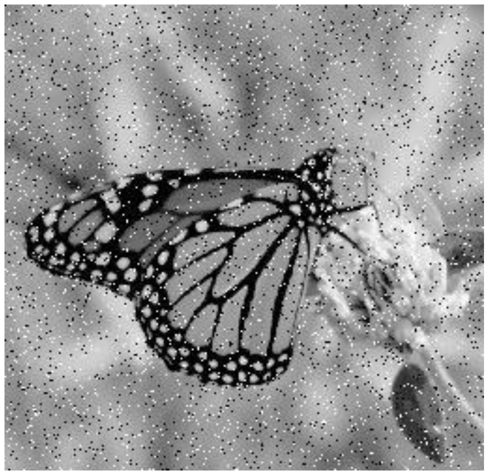
Figure (2.13): Mean Filter.

The median filter is a non-linear filter (order filter). These filters are based on as specific type of image statistics called order statistics. Typically, these filters operate on small sub image, “Window”, and replace the centre pixel value (similar to the convolution process).

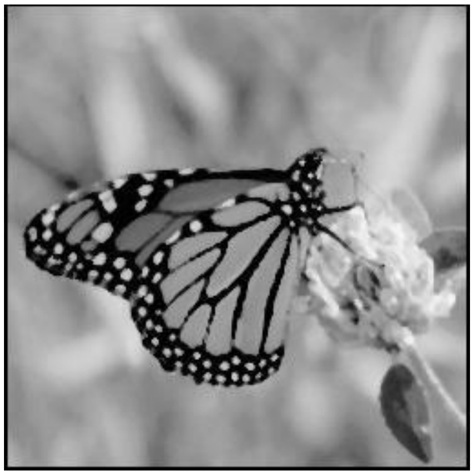
Order statistics is a technique that arranges the entire pixel in sequential order, given an N×N window (W) the pixel values can be ordered from smallest to the largest.

$$\mathbf{I_1 \leq I_2 \leq I_3 \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots < I_N}$$

Where $\mathbf{I_1, I_2, I_3 \ldots\ldots\ldots, I_N}$ are the intensity values of the subset of pixels in the image.



a. Salt and pepper noise



b. Median filtered image (3x3)

Figure (2.14): Median Filter

<u>Example:</u>

Given the following 3X3 neighborhood

$$\begin{pmatrix} 5 & 5 & 6 \\ 3 & 4 & 5 \\ 3 & 4 & 7 \end{pmatrix}$$

We first sort the value in order of size (3,3,4,4,<u>5</u>,5,5,6,7) ; then we select the middle value ,in this case it is 5. This 5 is then placed in centre location.

A median filter can use a neighbourhood of any size, but 3X3, 5X5 and 7X7 are typical. Note that the output image must be written to a separate image (a buffer); so that the results are not corrupted as this process is performed.

(The median filtering operation is performed on an image by applying the sliding window concepts, similar to what is done with convolution).

The window is overlaid on the upper left corner of the image, and the median is determined. This value is put into the output image (buffer) corresponding to the centre location of the window. The window is then slide one pixel over, and the process is repeated.

When the end of the row is reached, the window is slide back to the left side of the image and down one row, and the process is repeated. This process continues until the entire image has been processed.

Note that the outer rows and columns are not replaced. In practice this is usually not a problem due to the fact that the images are much larger than the masks. And these “wasted” rows and columns are often filled with zeros (or cropped off the image). For example, with 3X3 mask, we lose one outer row and column, a 5X5 mask we lose two rows and columns. This is not visually significant for a typical 256 X 256 or 512X512 images.

The maximum and minimum filters are two order filters that can be used for elimination of salt- and-pepper noise. The maximum filter selects the largest value within an ordered window of pixels values; whereas the minimum filter selects the smallest value.

The minimum filters works best for salt- type noise (High value), and the maximum filters work best for pepper-type noise.

In a manner similar to the median, minimum and maximum filter, order filter can be defined to select a specific pixel **rank** within the ordered set. For example we may find for certain type of pepper noise that selecting the second highest values works better than selecting the maximum value. This type of ordered selection is very sensitive to their type of images and their use it is application specific. It should note that, in general a **minimum or low rank filter** will tend to darken an image and a maximum or high rank filter will tend to brighten an image.

The **midpoint filter** is actually both order and mean filter because it rely on ordering the pixel values , but then calculated by an averaging process. This midpoint filter is the average of the maximum and minimum within the window as follows:

$$\text{Order set} = \mathbf{I_1 \le I_2 \le I_3 \ldots\ldots\ldots\ldots\ldots\ldots \le I_{N^2}}.$$

$$\mathbf{Midpoint = (I_1 + I_{N^2})/2}$$

The midpoint filter is most useful for Gaussian and uniform noise.

2.5.3 The Enhancement filter:

The enhancement filters are:

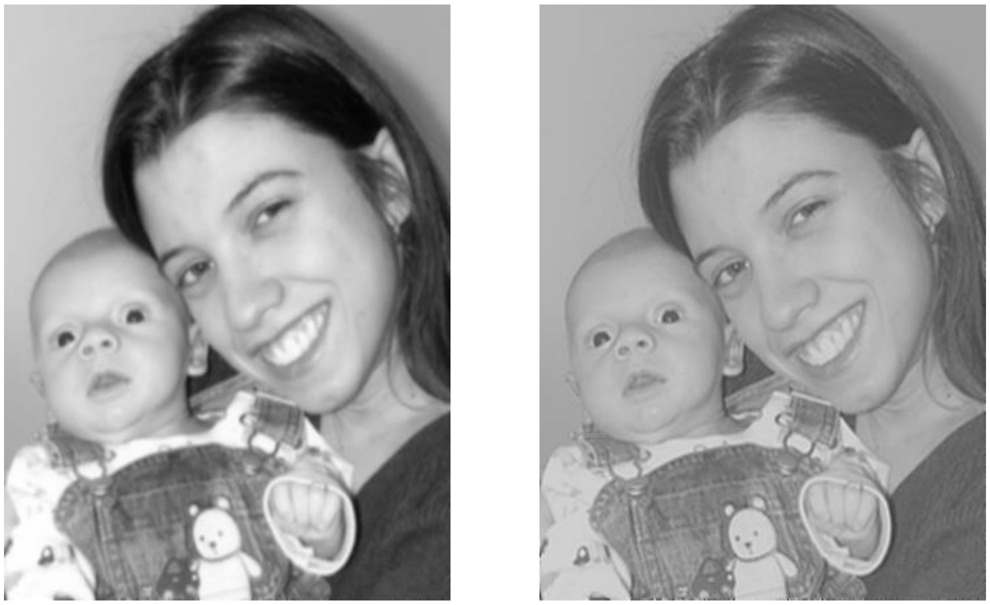
1. Laplacian type.
2. Difference filter.

These filters will tend to bring out, or enhance details in the image.

Example of convolution masks for the Laplacian-type filters are:

$$\begin{pmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{pmatrix} \quad \begin{pmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{pmatrix} \quad \begin{pmatrix} -2 & 1 & -2 \\ 1 & 5 & 1 \\ -2 & 1 & -2 \end{pmatrix}$$

The Laplacian type filters will enhance details in all directions equally.



a. Original image b. Laplacian filtered image

Figure (2.15): Laplacian Filter.

The difference filters will enhance details in the direction specific to the mask selected. There are four different filter convolution masks, corresponding to lines in the vertical, horizontal and two diagonal directions.

Vertical

$$\begin{pmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & -1 & 0 \end{pmatrix}$$

Horizontal

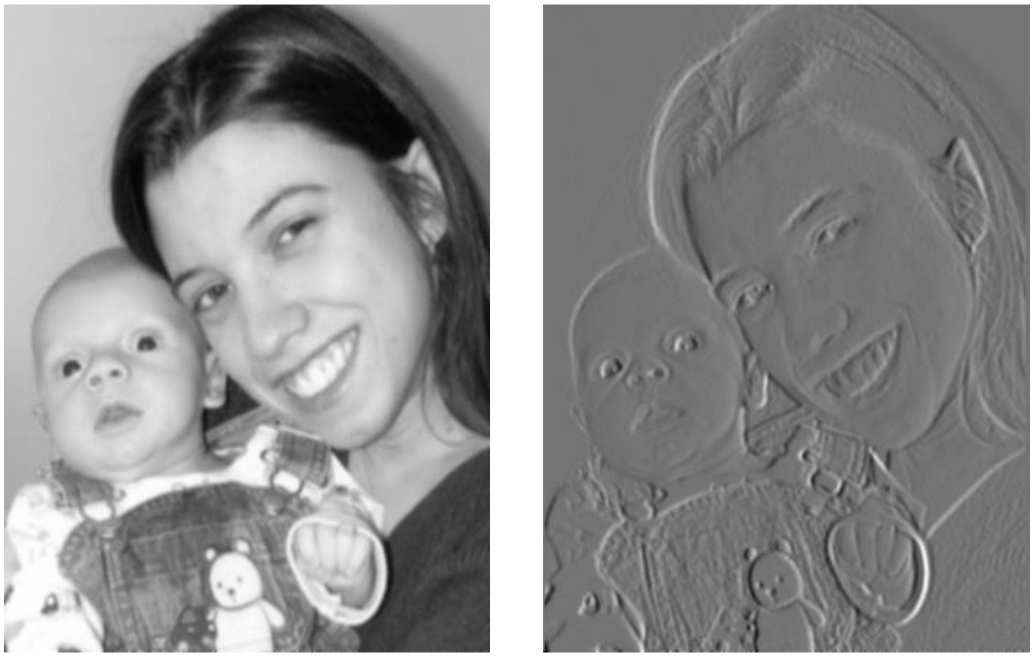
$$\begin{pmatrix} 0 & 0 & 0 \\ 1 & 1 & -1 \\ 0 & 0 & 0 \end{pmatrix}$$

Diagonal 1

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{pmatrix}$$

Diagonal 2

$$\begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{pmatrix}$$



a. Original image b. Difference filtered image

Figure (2.16): Difference Filter

2.5.4 <u>Image quantization</u>

Image quantization is the process of reducing the image data by removing some of the detail information by mapping group of data points to a single point. This can be done by:

1. Gray_Level reduction (reduce pixel values themselves I(r, c).
2. Spatial reduction (reduce the spatial coordinate (r, c).

The simplest method of gray-level reduction is **Thresholding.** We select a threshold gray _level and set everything above that value equal to "1" and everything below the threshold equal to "0". This effectively turns a gray_level image into a binary (two level) image and is often used as a preprocessing step in the extraction of object features, such as shape, area, or perimeter.

A more versatile method of gray _level reduction is the process of taking the data and reducing the number of bits per pixel. This can be done very efficiency by masking the lower bits via an AND operation. Within this method, the numbers of bits that are masked determine the number of gray levels available.

<u>Example</u>:

We want to reduce 8_bit information containing 256 possible gray_level values down to 32 possible values.

This can be done by ANDing each 8-bit value with the bit string **1111000.** this is equivalent to dividing by eight(2^3), corresponding to the lower three bits that we are masking and then shifting the result left three times. [Gray _level in the image 0-7 are mapped to 0, gray_level in the range 8-15 are mapped to 8 and so on].

We can see that by masking the lower three bits we reduce 256 gray levels to 32 gray levels:

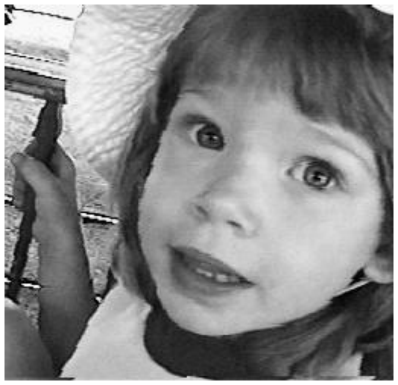
256 ÷ 8= 32

The general case requires us to mask k bits, where $2^{\mathbf{k}}$ is divided into the original gray-level range to get the quantized range desired. Using this method, we can reduce the number of gray levels to any power of 2: 2,4,6,8, 16, 32, 64 or 128.

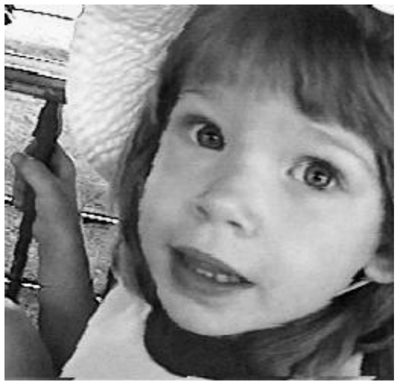
- Image quantization by masking to 128 gray level, this can be done by ANDing each 8-bit value with bit string 11111110($2^{\mathbf{1}}$).
- Image quantization by masking to 64 gray_level. This can be done by ANDing each 8-bit value with bit string 11111100(2^2).

As the number of gray levels decreases, we can see increase in a phenomenon called contouring.

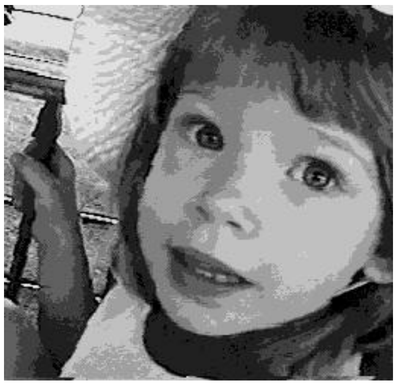
Contouring appears in the image as false edges, or lines as a result of the gray _level quantization method.



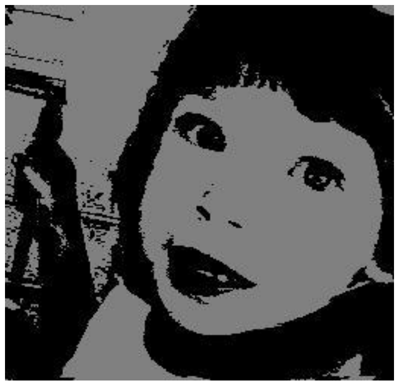
Original 8-bit image,
256 gray levels



Quantized to 6 bits,
64 gray levels



Quantized to 3 bits,
8 gray levels

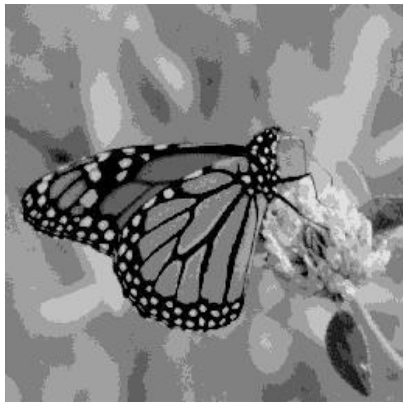


Quantized to 1 bits,
2 gray levels

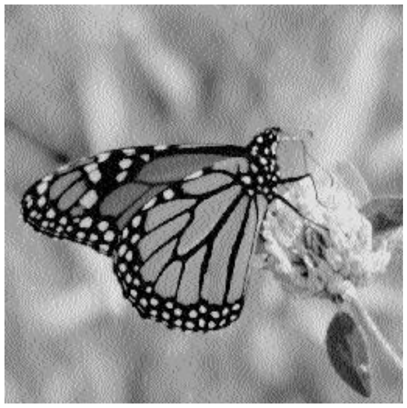
Figure (2-17): False Contouring

This false contouring effect can be visually improved upon by using an IGS (improved gray-scale) quantization method. In this method (IGS) the improvement will be by adding a small random number to each pixel before quantization, which results in a more visually pleasing appearance.

Original Image



Uniform quantization
to 8 levels (3 bits)



IGS quantization
to 8 levels (3 bits)

Figure (2-18): IGS quantization

Chapter Three
Edge/Line Detection

3.1 Edge Detection

Detecting edges is a basic operation in image processing. The edges of items in an image hold much of the information in the image.

The edges tell you where:

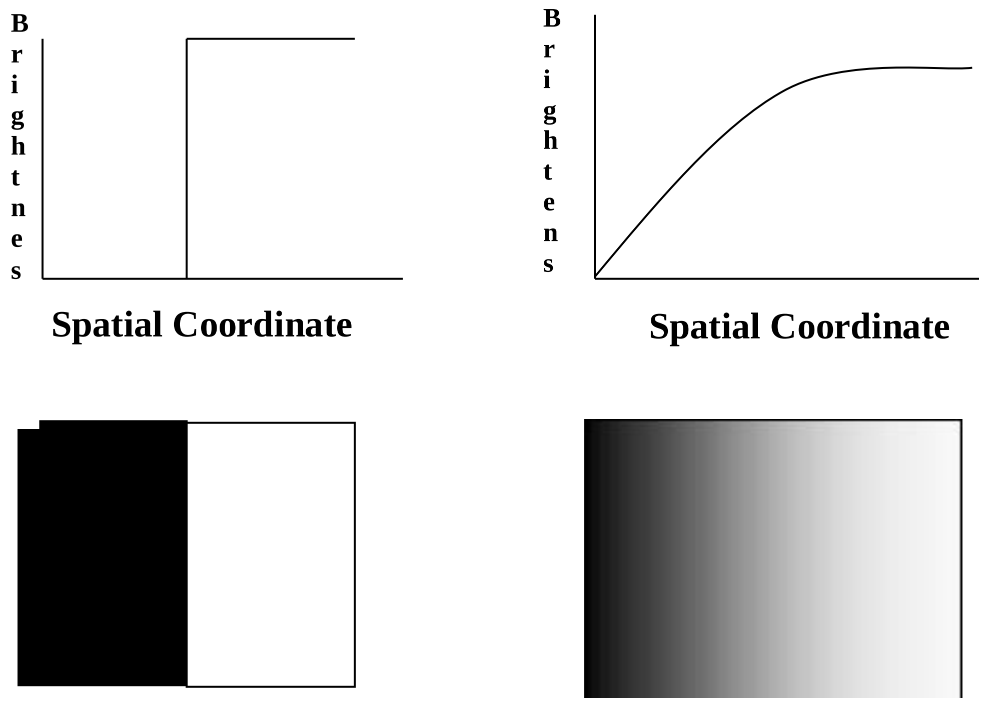
- Items are.
- Their size.
- shape
- and something about their texture.

Edge detection methods are used as a first step in the line detection processes, and they are used to find object boundaries by marking potential edge points corresponding to place in an image where rapid changes in brightness occur. After these edge points have been marked, they can be merged to form lines and objects outlines.

Edge detection operations are based on the idea that edge information in an image is found by looking at the relationship a pixel has with its neighbors. If a pixel gray_level values similar to those around it, there is probably not an edge at that point. However, if a pixel has neighbors with widely varying gray levels, it may represent an edge point. In other words, an edge is defined by a discontinuity in gray-level values. Ideally, an edge separates two distinct objects. In practice, edges are caused by:

- Change in color or texture or
- Specific lighting conditions present during the image acquisition process.

The following figure illustrates the difference between an ideal edge and a real edge.



a. Ideal Edge **b. Real Edge**

Figure (3-1): Ideal vs. Real Edge

The vertical axis represents brightness, and the horizontal axis shows the spatial coordinates. The abrupt change in brightness characterizes an ideal edge. In the figure (b) we see the representation of real edge, which change gradually. This gradual change is a minor form of blurring caused by:

- Imaging devices
- The lenses
- Or the lighting and it is typical for real world (as opposed to computer _generated) images.

An edged is where the gray level of the image moves from an area of low values to high values or vice versa. The edge itself is at the centre of this transition. The detected edge gives a bright spot at edge and dark area everywhere else. This mean it is the slope or rate of change of the gray level in the edge.

How do you calculate the derivative (the slop) of an image in all direction?

Convolution of the image with masks is the most often used techniques of doing this.

The idea is to take a 3×3 array of numbers and multiply it point by point with 3×3 section of the image you sum the products and place the result in the centre point of the image.

The question in this operation is how to choose the 3×3 mask?

There are several masks that amplify the slop of the edge. Take the simple one-dimensional case and take as an example points on the ideal edge near the edge. They could have values such as [3 5 7]. The slop through these three points is (7-3)/2=2.if you convolve these three point with [-1 0 1] you have -3+7=4.

The convolution amplified the slope, and the result is a large number at the transition point in the edge.

There are two basic principles for each edge detector mask:

- **First:** the number in the mask sum to zero. If 3×3 areas of an image contains a constant value (such as all ones), then there are no edges in that area. The result of convolving that area with a mask should be zero. If the numbers in the mask sum to zero, then convolving the mask with a constant area will result in the correct answer of zeros.

- **Second:** the masks should approximate differentiation or amplify the slope of the edge. The simple example [-1 0 1] given earlier showed how to amplify the slope of the edge.

The number of masks used for edge detection is almost limitless. Research have used different techniques to derive masks, some of will be illustrated in the following section.

<u>1- Sobel Operator</u>: The Sobel edge detection masks look for edges in both the horizontal and vertical directions and then combine this information into a single metric. The masks are as follows:

$$\begin{array}{cc} \textbf{Row Mask} & \textbf{Column Mask} \\ \begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix} & \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix} \end{array}$$

These masks are each convolved with the image. At each pixel location we now have two numbers: S1, corresponding to the result form the row mask and S2, from the column mask. We use this numbers to compute two matrices, the edge magnitude and the edge direction, which are defined as follows:

Edge Magnitude= $\sqrt{S_1^2 + S_2^2}$

Edge Direction = Tan $^{-1}$ $\left[\frac{S_1}{S_2}\right]$

<u>2- Prewitt Operator:</u> The Prewitt is similar to the Sobel but with different mask coefficients. The masks are defined as follows:

$$\begin{array}{cc} \textbf{Row Mask} & \textbf{Column Mask} \\ \begin{pmatrix} -1 & -1 & -1 \\ & & \\ & & \end{pmatrix} & \begin{pmatrix} -1 & 0 & 1 \\ & & \\ & & \end{pmatrix} \end{array}$$

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

These masks are each convolved with the image. At each pixel location we find two numbers: P1 corresponding to the result from the row mask and P2 from the column mask. We use these results to determine two metrics, the edge magnitude and edge direction, which are defined as follows:

Edge Magnitude= $\sqrt{P_1^2 + P_2^2}$

Edge Direction = Tan $^{-1}$ $\left[\frac{P_1}{P_2}\right]$

3- Kirch Compass (محيط) Mask: the Kirch edge detection masks are called compass masks because they are defined by taking a single mask and rotating it to the eight major compass orientations (اتجاهات) :
north, north-east, east, south-east, south, south-west, and west and north-west edges in an image. The masks are defined as follows:

$$\begin{pmatrix} -3 & -3 & 5 \\ -3 & 0 & 5 \\ -3 & -3 & 5 \end{pmatrix} \quad \begin{pmatrix} -3 & 5 & 5 \\ -3 & 0 & 5 \\ -3 & -3 & -3 \end{pmatrix} \quad \begin{pmatrix} 5 & 5 & 5 \\ -3 & 0 & -3 \\ -3 & -3 & -3 \end{pmatrix} \quad \begin{pmatrix} 5 & 5 & -3 \\ 5 & 0 & -3 \\ -3 & -3 & -3 \end{pmatrix}$$

K_0 **K_1** **K_3** **K_4**

$$\begin{pmatrix} 5 & -3 & -3 \\ 5 & 0 & -3 \\ 5 & -3 & -3 \end{pmatrix} \quad \begin{pmatrix} -3 & -3 & -3 \\ 5 & 0 & -3 \\ 5 & 5 & -3 \end{pmatrix} \quad \begin{pmatrix} -3 & -3 & -3 \\ -3 & 0 & -3 \\ 5 & 5 & 5 \end{pmatrix} \quad \begin{pmatrix} -3 & -3 & -3 \\ -3 & 0 & 5 \\ -3 & 5 & 5 \end{pmatrix}$$

K_4 **K_5** **K_6** **K_7**

The edge magnitude is defined as the maximum value found by the convolution of each of the mask, with the image.

[Given a pixel, there are eight directions you can travel to a neighbouring pixel (above, below , left ,right ,upper left, upper right, lower left, lower right). Therefore there are eight possible directions for an edge. The directional edge detectors can detect an edge in only one of the eight directions. If you want to detect only left to right edges, you would use only one of eight masks. If; however you want to detect all of the edges, you would need to perform convolution over an image eight times using each of the eight masks].

<u>4- Robinson Compass Masks</u>: the Robinson compass masks are used in a manner similar to the Kirch masks but are easier to implement because they rely only on coefficient of 0,1 and 2, and are symmetrical about their directional axis-the axis with the zeroes, we only need to compute the result on four of the mask. From the other four can be obtained by negating the results from the first four. The masks are as follows:

$$\underset{\mathbf{R_0}}{\begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}} \quad \underset{\mathbf{R_1}}{\begin{pmatrix} 0 & 1 & 2 \\ -1 & 0 & 1 \\ -2 & -1 & 0 \end{pmatrix}} \quad \underset{\mathbf{R_3}}{\begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}} \quad \underset{\mathbf{R_4}}{\begin{pmatrix} 2 & 1 & 0 \\ 1 & 0 & -1 \\ 0 & -1 & -2 \end{pmatrix}}$$

$$\underset{\mathbf{R_4}}{\begin{pmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{pmatrix}} \quad \underset{\mathbf{R_5}}{\begin{pmatrix} 0 & -1 & -2 \\ 1 & 0 & -1 \\ 2 & 1 & 0 \end{pmatrix}} \quad \underset{\mathbf{R_6}}{\begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix}} \quad \underset{\mathbf{R_7}}{\begin{pmatrix} -2 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 2 \end{pmatrix}}$$

The edge magnitude is defined as the maximum value found by the convolution of each of the masks with the image. The edge detection is defined by the mask that produces the maximum magnitude.

It's interesting to note that masks R_0 and R_7 are the same as the Sobel masks. We can see that any of the edge detection masks can be extended by rotating them in a manner like these compass masks which allow us to extract explicit information about edge in any direction.

5- Laplacian Operators: the Laplacian operator described here are similar to the ones used for pre-processing (as described in enhancement filter). The three Laplacian masks that follow represent different approximation of the Laplacian masks are rationally symmetric, which means edges at all orientation contribute to the result. They are applied by selecting one mask and convolving it with the image

Laplacian masks

$$\begin{pmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{pmatrix} \qquad \begin{pmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{pmatrix} \qquad \begin{pmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{pmatrix}$$

These masks differ from the Laplacian type previously described in that the centre coefficients have been decreased by one. So, if we are only interested in edge information, the sum of the coefficients should be **zero**. If we want to retain most of the information the coefficient should sum to a number greater than zero. Consider an extreme example in which the centre coefficient value will depend most heavily up on the current value, with only minimal contribution from surrounding pixel values.

3.2 Other Edge Detection Methods

Two other methods using Gaussian and homogeneity/difference operators are given below:

$$\begin{pmatrix} \mathbf{0} & \mathbf{0} & \mathbf{-1} & \mathbf{-1} & \mathbf{-1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{-2} & \mathbf{-3} & \mathbf{-3} & \mathbf{-3} & \mathbf{-2} & \mathbf{0} \\ \mathbf{-1} & \mathbf{-3} & \mathbf{5} & \mathbf{5} & \mathbf{5} & \mathbf{-3} & \mathbf{-1} \\ \mathbf{-1} & \mathbf{-3} & \mathbf{5} & \mathbf{16} & \mathbf{5} & \mathbf{-3} & \mathbf{-1} \\ \mathbf{0} & \mathbf{0} & \mathbf{-1} & \mathbf{-1} & \mathbf{-1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{-2} & \mathbf{-3} & \mathbf{-3} & \mathbf{-3} & \mathbf{-2} & \mathbf{0} \\ \mathbf{-1} & \mathbf{-3} & \mathbf{5} & \mathbf{5} & \mathbf{5} & \mathbf{-3} & \mathbf{-1} \end{pmatrix}$$

7×7 Gaussian Mask to detect large edges

Gaussian edge detector has the advantage that the details in the output image can be adjusted by varying the width of the convolution mask. A wider mask eliminates small or fine edges and detects only large, significant edges.

Other than by masking, edge detection can also be performed by *subtraction.* Two methods that use subtraction to detect the edge are **Homogeneity operator** and **Difference operator**.

The **homogeneity operator** subtracts each of the pixels next to the center of the n×n area (where n is usually 3) from the center pixel. The result is the maximum of the absolute value of these subtractions. Subtraction in a homogenous region produces zero and indicates an absence of edges. A high maximum of the subtractions indicates an edge. This is a quick operator since it performs only subtraction- eight operations per pixel and no multiplication. This operator then requires thresholding. If there is no thresholding then the resulting image looks like a faded copy of the original.

Generally thresholding at 30 to 50gives good result. The thresholding can be varied depending upon the extent of edge detection desired.

The **difference operator** performs differentiation by calculating the differences between the pixels that surround the center pixel of an n×n area. This operator finds the absolute value of the difference between the opposite pixels, the upper left minus the lower right, upper right minus the lower left, left minus right, and top minus bottom. The result is the maximum absolute value. as in the homogeneity case, this operator requires thresholding. But it is quicker than the homogeneity operator since it uses four integer subtractions as against eight subtractions in homogeneity operator per pixel.

Shown below is how the two operators detect the edge:

Consider an image block with center pixel intensity 5,

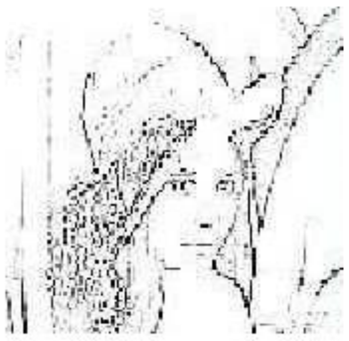
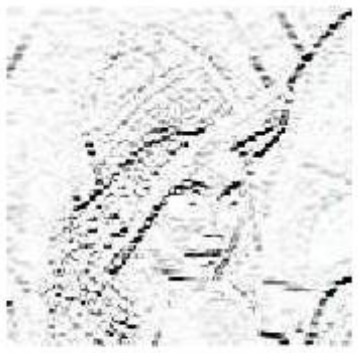
$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

Output of *homogeneity operator* is:

Max of {| 5-1 |, | 5-2 |, | 5-3 |, | 5-4 |, | 5-6 |, | 5-7 |, | 5-8 |, | 5-9 | } = 4

Output of *difference operator* is:

Max of {| 1-9 |, | 7-3 |, | 4-6 |, | 2-8 | } = 8

		
Original Image	**Edge detect by Sobel' horizontal mask**	**Edge detect by Sobel overall mask**

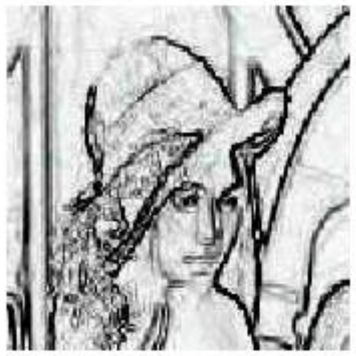
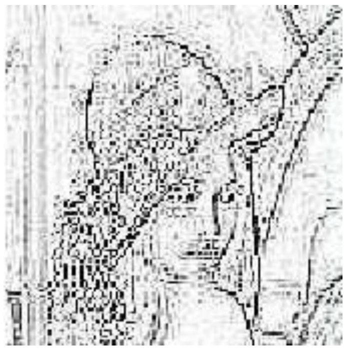
	
Edge detect by Gaussian mask	**Edge detect by Prewitt mask**

Figure (3-2) : Example of Edge Operators

Chapter Four
Color images formats (RGB, HSL and YCbCr)

4.1 RGB Color Format

Any color that can be represented on a computer monitor is specified by means of the three basic colors- Red, Green and Blue called the RGB colors. By mixing appropriate percentages of these basic colors, one can design almost any color one ever imagines.

The model of designing colors based on the intensities of their RGB components is called the ***RGB model***, and it’s a fundamental concept in computer graphics. Each color, therefore, is represented by a triplet (Red, Green, and Blue), in which red, green and blue three bytes are that represent the basic color components. The smallest value, 0, indicates the absence of color. The largest value, 255, indicates full intensity or saturation. The triplet (0, 0, 0) is black, because all colors are missing, and the triplet (255, 255, 255) is white. Other colors have various combinations:

(255,0,0) is pure red, (0,255,255) is a pure cyan (what one gets when green and blue are mixed), and (0,128,128) is a mid-cyan (a mix of mid-green and mid-blue tones).

The possible combinations of the three basic color components are 256×256×256, or 16,777,216 colors. Figure (4.1) shows Color specification of the RGB Color Cube

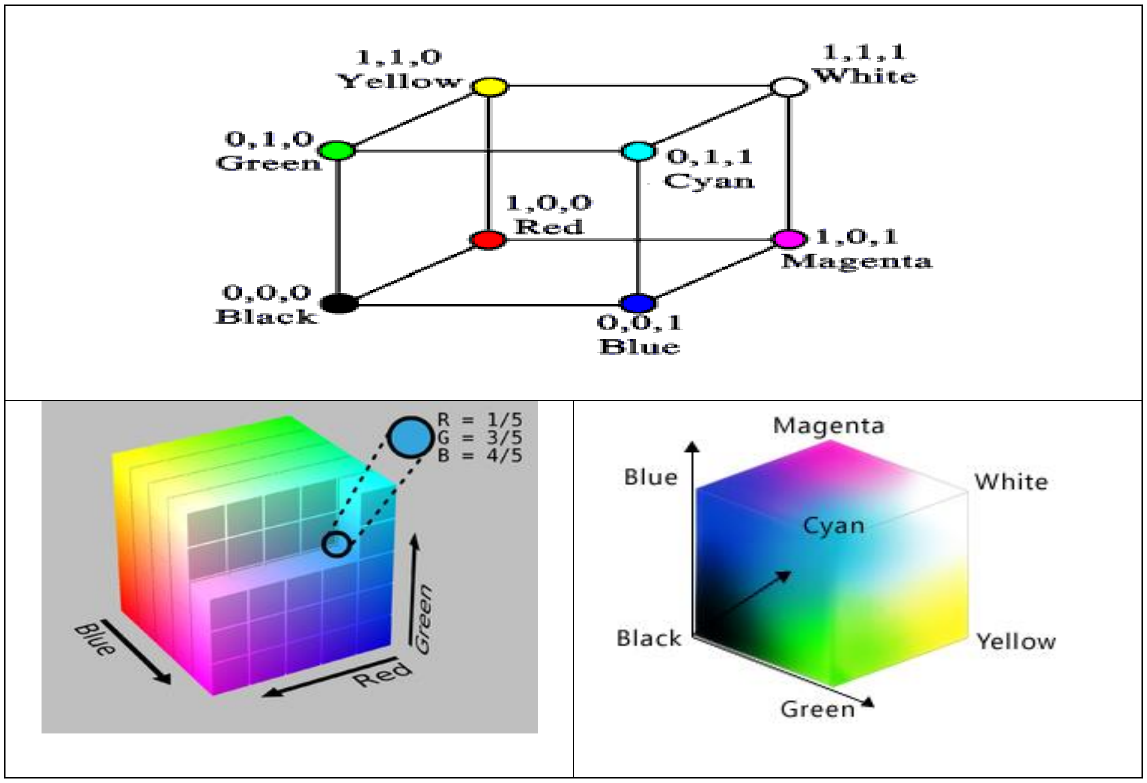


Figure (4.1): Color specification of the RGB Color Cube

The process of generating colors with three basic components is based on the RGB Color cube as shown in the above figure. The three dimensions of the color cube correspond to the three basic colors. The cube's corners are assigned each of the three primary colors, their complements, and the colors black and white. Complementary colors are easily calculated by subtracting the Color values from 255. For example, the color **(0, 0,255) is a pure blue tone. Its complementary color is (255-0,255-0,255-255), or (255, 255, 0),** which is a pure yellow tone. Blue and Yellow are complementary colors, and they are mapped to opposite corners of the cube. The same is true for red and cyan, green and magenta, and black and white. Adding a color to its complement gives white.

It is noticed that the components of the colors at the corners of the cube have either zero or full intensity. As we move from one corner to another along the same edge of the cube, only one of its components changes value. For example, as we move from the Green to the Yellow corner, the Red component changes from 0 to 255. The other two components remain the same. As we move between these two corners, we get all the available tones from Green to Yellow (256 in all). Similarly, as one moves from the Yellow to the Red corner, the only component that changes is the Green, and one gets all the available shades from Yellow to Red. This range of similar colors is called gradient.

Although we can specify more than 16 million colors, we can’t have more than 256 shades of Gray. The reason is that a gray tone, including the two extremes (black and white), is made up of equal values of all the three primary colors. This is seen from the RGB cube as well. Gray shades lie on the cube’s diagonal that goes from black to white. As we move along this path, all three basic components change value, but they are always equal. The value (128,128,128) is a mid-gray tone, but the values (129,128,128) aren’t gray tones, although they are too close for the human eye to distinguish. That's why it is wasteful to store grayscale pictures using 16-million color True Color file formats. A 256-color file format stores a grayscale just as accurately and more compactly. Once an image is known in grayscale, we needn’t store all three bytes per pixel. One value is adequate (the other two components have the same value).

4.2 HSV Color Format

HSV color space HSL or HSI is one color space, which describes colors as perceived by human beings. HSI (or HSV) stands for hue (H), (S) saturation and intensity (I) (or value V). For example, a blue car reflects blue hue. Moreover is also attributing of the human perception. The hue which is essentially the chromatic component of our perception may again be considered as weak hue or strong hue. The colorfulness of a color is described by the saturation component. For example, the color from a single monochromatic source of light, which produce colors of a single wavelength only , is highly saturated , while the colors comprising hues of different wavelengths have little Chroma and have less saturation.

The gray colors do not have any hue and hence they have less saturation or unsaturated. Saturation is thus a measure of colorfulness or whiteness in the color perceived.

The lightness (L) or intensity (I) or value (V) essentially provides a measure of the brightness of colors. This gives a measure of how much light is reflected from the object or how much light is emitted from a region.

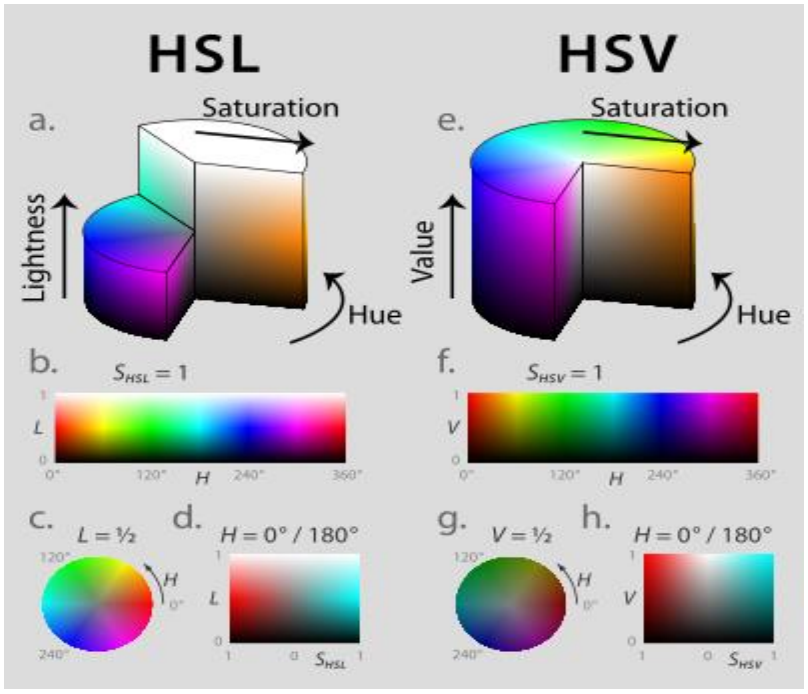


Figure (4.2) : HSL and HSV Color Space

The HSV image may be computed from RGB image using different transformation. Some of them are as follows:

A. <u>The simplest form of HSV transformation is</u> :

H= tan [3(G-B)/(R-G) +(R-B)]

S= 1-(min(R, G, B)/V)

V= (R+G+B/3)

However, the hue (H) becomes undefined when saturation S=0

B. <u>The most popular form of HSV transformation is shown next</u>, where the r, g, b values are first obtained by normalizing each pixel such that

r=(R/(R+G+B)), g= (G/(R+G+B)), b= (B/(R+G+B))

Accordingly the H, S and V value can be computed as

V=max (r, g, b),

$$S = \begin{cases} 0, & if\ V = 0 \\ V - \min(r,g,b)/V, & if\ V > 0 \end{cases}$$

$$H = \begin{cases} 0, & if\ S = 0 \\ 60 * [(g-b)/S * V] & if\ V = r \\ 60\,[2 + ((b-r)/S * V)] & if\ V = g \\ 60[4 + ((r-g)/S * V)] & if\ V = b \end{cases}$$

H=H+360 if H<0

The HSV color space is compatible with human color perception.

4.3 YC_bC_r Color Format

In this color space, Y is the luminous component while Cb and Cr, provide the color information. The color information is stored as two color difference components C_b and C_r This color space is used in digital video. The information from RGB to YC_bC_r is as follows:

$$\begin{bmatrix} Y \\ Cb \\ Cr \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.169 & -0.331 & 0.500 \\ 0.500 & -0.419 & -0.081 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

4.4 Basic Relationships between Pixels

In this section, we consider several important relationships between pixels in a digital image. As mentioned before, an image is denoted by **f(x, y)**.When referring in this section to a particular pixel, we use lowercase letters, such as **p** and **q**.

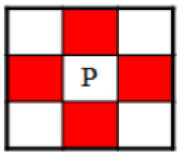
The basic relationships between pixels are:

- **Neighborhood**
- **Adjacency**
- **Connectivity**
- **Paths**
- **Regions and boundaries**

1- Neighbors of a Pixel

The 4- neighbors of pixel p are: **N_4 (p)** any pixel p(x,y) has two vertical and two horizontal neighbors, given by:

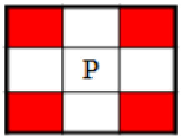
(x+1, y), (x-1, y), (x, y+1), (x, y-1)



This set of pixels, called the **4-neighbors** of **p**, is denoted by **N_4 (p)**. Each pixels a 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.

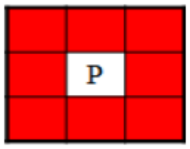
The four diagonal neighbors of **p** have coordinates **(x+1, y+1), (x+1, y-1), (x-1, y+1), (x-1, y-1)** and are denoted by **N_D(p)**.

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

$N_8(p) = N_4(p) \cup N_D(p)$



2- Connectivity

Connectivity between pixels is a fundamental concept that simplifies the definition of numerous digital image concepts, such as **regions** and **boundaries**. To establish if two pixels are connected, it must be determined if they are neighbors and if their gray levels satisfy a specified criterion of similarity (say, if their gray levels are equal). For instance, in a binary image with values **0** and **1**, two pixels may be **4-neighbors**, but they are said to be connected only if they have the same value.

In other words two pixels are said to be connected if they are adjacent in some sense.

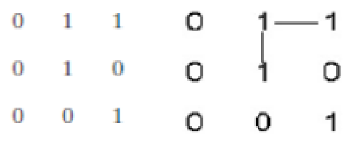
- They are neighbors (N_4, N_D, N_8) and
- Their intensity values (gray levels) are similar

3- Adjacency

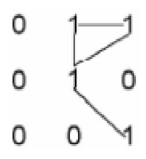
Let V be the set of gray-level values used to define adjacency. In a binary Image, V= {1} if we are referring to adjacency of pixels with value 1. In a gray scale image, the idea is the same, but set V typically contains more elements. For example, in the adjacency of pixels with a range of possible gray-level values 0 to 255, set V could be any subset of these 256 values.

We consider three types of adjacency:

(a) **4-adjacency**.Two pixels **p** and **q** with values from **V** are **4-adjacent** if **q** is in the set **N_4 (p)**.



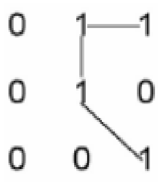
(b) **8-adjacency**. Two pixels **p** and **q** with values from **V** are **8-adjacent** if **q** is in the set **N_8(p)**.



(c) **m-adjacency** (**mixed adjacency**).Two pixels **p** and **q** with values from **V** are **m-adjacent** if

(i) **q** is in **N_4(p)**, or

(ii) **q** is in **N_D(p)** and the set N4(p) ∩ N4(q) is empty (has no pixels whose values are from V)



Two image subsets S1 and S2 are adjacent if some pixel in S1 is adjacent to some pixel in S2

<u>Example :</u>

Consider the two image subsets, S1 and S2, shown in the following figure. For V={1}, determine whether these two subsets are (a) 4-adjacent, (b) 8-adjacent, or (c) m-adjacent?

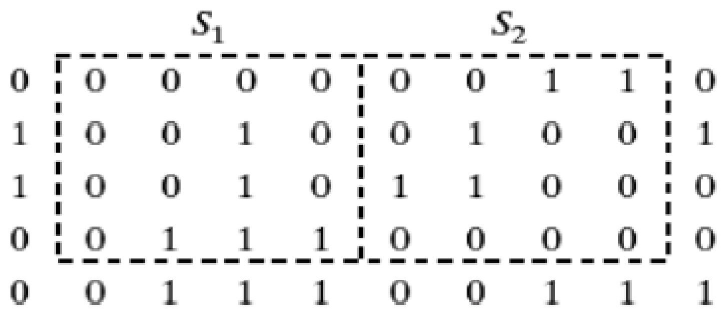


Figure (4.4): two image subsets

Sol:

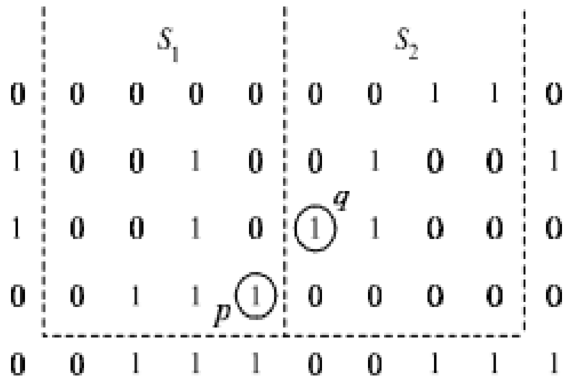
Let p and q be as shown in below figure, then,

(a) S1 and S2 are **not 4-** adjacent because q is not in the set N_4 (p);

(b) S1 and S2 are 8- adjacent because q is in the set N_8 (p);

(c) S1 and S2 are m- adjacent because:

(i) q is in $N_D(p)$, and (ii) the set $N_4(p) \cap N_4(q)$ is empty



4- Paths

A (digital) *path* (or *curve*) from pixel p with coordinates (x, y) to pixel q with coordinates (s, t) is a sequence of distinct pixels with coordinates

$$\boxed{(x_0, y_0), (x_1,y_1), \ldots\ldots, (x_n, y_n)}$$

- where $(x_0, y_0) = (x, y)$, $(x_n, y_n) = (s, t)$,
- and pixels (x_i, y_i) and (x_{i-1}, y_{i-1}) are adjacent for $1 \le i \le n$.
- In this case, n is the length of the path.
- If $(x_0, y_0) = (x_n, y_n)$ the path is a closed path.
- The path can be defined 4-,8-,m-paths depending on adjacency type.

Let S be 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**.

A (digital) *path* (or *curve*) from pixel p with coordinates (x, y) to pixel q with coordinates (s, t) is a sequence of distinct pixels with coordinates

$$\boxed{(x_0, y_0), (x_1,y_1), \ldots\ldots, (x_n, y_n)}$$

- where $(x_0, y_0) = (x, y)$, $(x_n, y_n) = (s, t)$,
- and pixels (x_i, y_i) and (x_{i-1}, y_{i-1}) are adjacent for $1 \leq i \leq n$.
- In this case, n is the length of the path.
- If $(x_0, y_0) = (x_n, y_n)$ the path is a closed path.
- The path can be defined 4-,8-,m-paths depending on adjacency type.

Let S be 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**.

Example: Consider the image segment shown.

$$\begin{array}{ccccccc} & \mathbf{3} & \mathbf{1} & \mathbf{2} & \mathbf{1} & \mathbf{(q)} \\ & \mathbf{2} & \mathbf{2} & \mathbf{0} & \mathbf{2} & \\ & \mathbf{1} & \mathbf{2} & \mathbf{1} & \mathbf{1} & \\ \mathbf{(p)} & \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{2} & \end{array}$$

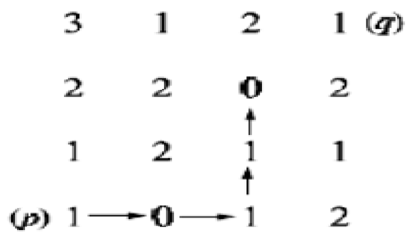
A. Let V={0, 1} and compute the length of shortest 4-, 8-, and m-path between p and q. If a particular path does not exist between these two points explain why?

B. Repeat for V={1, 2}.

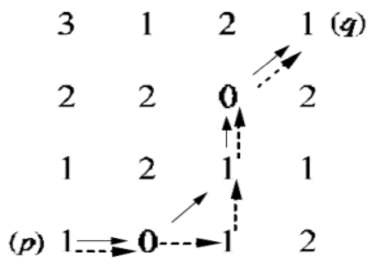
Solution: A

- When **V = {0,1}, 4-path** does not exist between **p** and **q** because it is impossible to get from **p** to **q** by traveling along points that are both **4-**

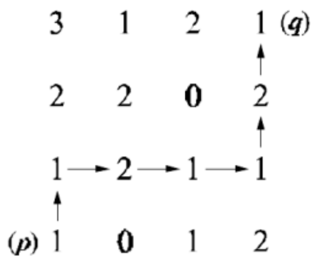
adjacent and also have values from **V**. Figure below shows this condition; <u>**it is not possible to get to q.**</u>



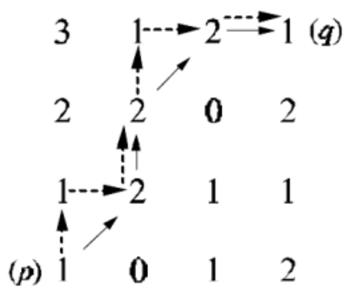
- The shortest **8-path** is shown in Figure **below,** its **length is 4**. The length of the shortest **m- path** (shown dashed) is **5**. Both of these shortest paths are **unique** in this case.



- One possibility for the shortest **4-path** when **V={1,2}** is shown in figure below ; **its length is 6**. It is easily verified that another **4-path** of the same length exists between **p** and **q**.



- One possibility for the shortest **8-path** (it is not unique) is shown in figure below; its length is **4**. The length of a shortest **m-path** (shown dashed) is **6**. This path is **not unique**.



5- Regions and boundaries

- Let R be a subset of pixels in an image. We call R a region of the image if R is a connected set.

- 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.

6- Distance Measures

- **Euclidean Distance** between two point p and q is defining as:

$$D_e(p,q) = \sqrt{(x-s)^2 + (y-t)^2}$$

- The **D_4 distance** between two point p and q is defining as:

$$\mathbf{D_4(p, q) = |x - s| + |y - t|}$$

- The **D_8 distance** between two point p and q is defining as:

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

<u>Example:</u>

What are the **Euclidean, D_4, D_8**, distances between points **p** and **q** if the coordinates of **P** = (3, 1) and **q** = (4, 4)?

Sol:

Euclidean Distance: $D_e(p,q) = \sqrt{(x-s)^2 + (y-t)^2}$

$$D_e(p,q) = \sqrt{(3-4)^2 + (1-4)^2} = \sqrt{10}$$

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

$D_4(p, q) = |3 - 4| + |1 - 4|$

$= |-1| + |-3| = 1+3 = 4$

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

$D_8(p, q) = \max(|3 - 4|, |1 - 4|)$

$= \max(|-1|, |-3|)$

$= \max(1, 3) = 3$

<u>Example</u>(1.4): Suppose you have the following sub image, calculate D_e, D_4, D_8 and D_m between (p,q) and(p,h) . Let V= {2}

$$\begin{array}{c|ccccc} & 0 & 1 & 2 & 3 & 4 \\ \hline 0 & 2 & 2 & 2 & 2 & 2_{(q)} \\ 1 & 2 & 1 & 1 & 1 & 2 \\ 2 & 2 & 1_{(p)} & 0 & 1 & 2_{(h)} \\ 3 & 2 & 1 & 1 & 1 & 2 \\ 4 & 2 & 2 & 2 & 2 & 2 \end{array}$$

$D_e(p, q) = \sqrt{(x-s)^2 + (y-t)^2}$

$= \sqrt{(2-0)^2 + (2-4)^2}$

$= \sqrt{4+4} = \sqrt{8}$

$D_e(p, h) = \sqrt{(2-2)^2 + (2-4)^2}$

$= \sqrt{0+4} = \sqrt{4} = 2$

$D_4(p,q) = (|2-0| + |2-4|)$

$= |2| + |2| = 4$

$D_4(p,h) = (|2-2| + |4-2|)$

$= |0| + |2| = 2$

$D_8(p,q) = \max(|2-0|, |4-2|)$

$= \max(2, 2) = 2$

$D_8(p,h) = \max(|2-2|, |4-2|)$

$= \max(0,2) = 2$

$\therefore$ D_8=shortest path, where the path does not depend on the value of the pixels.