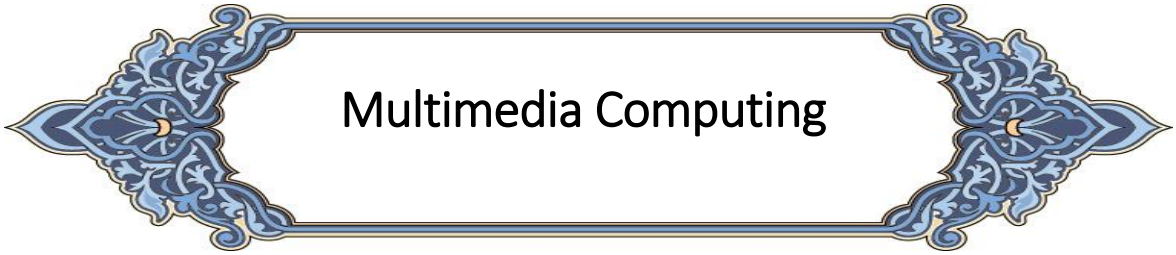
كلية التقنية الهندسية الكهربائية

قسم هندسة تقنيات الحاسوب



اعداد : د.عبدالرزاق طارق رحيم

المادة : حوسبة وسائط متعددة

المرحلة: الرابعة

السنة الدراسية 2017-2018

المحور الأول

1. Introduction to Multimedia

1.1 Introduction

Multimedia has become an inevitable part of any presentation. It has found a variety of applications right from entertainment to education. The evolution of internet has also increased the demand for multimedia content.

Definition

Multimedia is the media that uses multiple forms of information content and information processing (e.g. text, audio, graphics, animation, video, interactivity) to inform or entertain the user. Multimedia also refers to the use of electronic media to store and experience multimedia content. Multimedia is similar to traditional mixed media in fine art, but with a broader scope. The term "rich media" is synonymous for interactive multimedia.

1.2 Elements of Multimedia System

Multimedia means that computer information can be represented through audio, graphics, image, video and animation in addition to traditional media (text and graphics). Hypermedia can be considered as one type of particular multimedia application.

A Multimedia System is a system capable of processing multimedia data and applications.

A Multimedia System is characterized by the processing, storage, generation, manipulation and rendition of Multimedia information.

A Multimedia system has four basic characteristics:

- Multimedia systems must be computer controlled.
- Multimedia systems are integrated.
- The information they handle must be represented digitally.
- The interface to the final presentation of media is usually interactive.

1.3 Categories of Multimedia

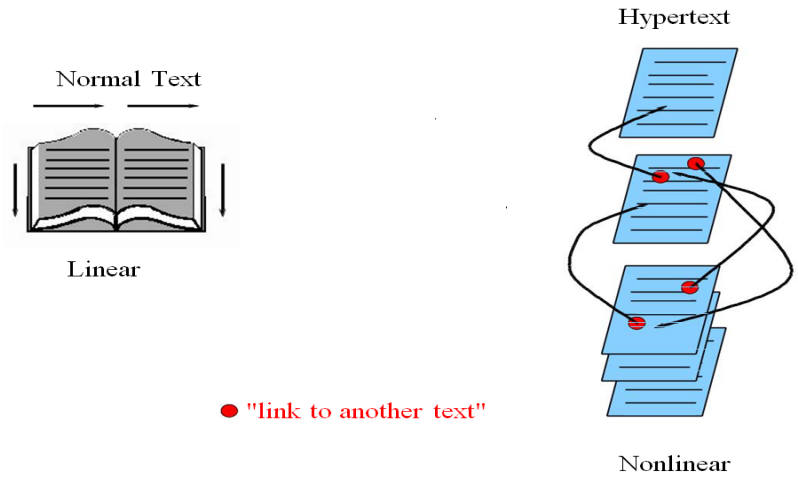
Multimedia may be broadly divided into linear and non-linear categories. Linear active content progresses without any navigation control for the viewer such as a cinema presentation. Non-linear content offers user interactivity to control progress as used with a computer game or used in self-paced computer based training. Non-linear content is also known as hypermedia content.

Multimedia presentations can be live or recorded. A recorded presentation may allow interactivity via a navigation system. A live multimedia presentation may allow interactivity via interaction with the presenter or performer.

2. "HyperText" and "HyperMedia"

Hypertext is a text which contains links to other texts

Hypertext System: meant to be read nonlinearly, by following links that point to other parts of the document, or to other documents



The World Wide Web (WWW) is the best example of a hypermedia application.

Multimedia means that computer information can be represented through audio, graphics, images, video, and animation in addition to traditional media.

HyperMedia: not constrained to be text-based, can include other media, e.g., graphics, images, and especially the continuous media — sound and video.

Example Hypermedia Applications:

- The World Wide Web (WWW) is a clear example of the hypermedia application.
- PowerPoint.
- Adobe Acrobat (or other PDF software).
- Adobe Flash

3. Components of Multimedia

Now let us consider the Components (Hardware and Software) required for a multimedia system:

1) ***Capture devices***: Video Camera, Video Recorder, Audio Microphone, Keyboards, mice, graphics tablets, 3D input devices, tactile sensors, VR devices. Digitizing Hardware.
2) ***Storage Devices***: Hard disks, CD-ROMs, DVD-ROM, etc.
3) ***Communication Networks***: Local Networks, Intranets, Internet, Multimedia or other special high speed networks.
4) ***Computer Systems***: Multimedia Desktop machines, Workstations, MPEG/VIDEO/DSP Hardware.
5) ***Display Devices***: CD-quality speakers, HDTV, SVGA, Hi-Res monitors, Color printers etc

Multimedia involves multiple modalities of text, audio, images, drawings, animation, and video. Examples of how these modalities are put to use:

1. Video teleconferencing.
2. Distributed lectures for higher education.
3. Tele-medicine.
4. Co-operative work environments.
5. Searching in (very) large video and image databases for target visual objects.
6. “Augmented” reality: placing real-appearing computer graphics and video objects into scenes.
7. Using voice-recognition to build an interactive environment, say a web browser

4. Multimedia Research Topics and Projects

To the computer science researcher, multimedia consists of a wide variety of **topics**:

1. ***Multimedia processing and coding:*** This includes multimedia content analysis, content-based multimedia retrieval, multimedia security, audio/image/video processing, compression, etc.
2. ***Multimedia system support and networking:*** network protocols, Internet, operating systems, servers and clients, quality of service (QoS), and databases.
3. ***Multimedia tools, end-systems and applications:*** These include hypermedia systems, user interfaces, authoring systems, multimodal interaction, and integration, web-everywhere devices, multimedia education, including computer supported collaborative learning and design, and applications of virtual environments.

The concerns of multimedia researchers also impact researchers in almost every other branch. For example, data mining is an important current research area, and a large database of multimedia data objects is a good example of just what we may be interested in mining.

4. ***Multi-modal interaction and integration:*** "ubiquity" web-everywhere devices, multimedia education including Computer Supported Collaborative Learning, and de-sign and applications of virtual environments.

Multimedia Projects

Many exciting research projects are currently underway in multimedia, and we'd like to introduce a few of them here:

- For example, researchers are interested in camera-based object tracking technology. One aim is to develop control systems for industrial control, gaming, and so on that rely on moving scale models (toys) around a real environment (a board game, say). Tracking the control objects (toys) provides user control of the process.
- 3D motion capture can also be used for multiple actor capture, so that multiple real actors in a virtual studio can be used to automatically produce realistic animated models with natural movement.
- Multiple views from several cameras or from a single camera under differing lighting can accurately acquire data that gives both the shape and surface properties of materials, thus automatically generating synthetic graphics models. This allows photo-realistic synthesis of virtual actors.
- 3D capture technology is next to fast enough now to allow acquiring dynamic characteristics of human facial expression during speech, to synthesize highly realistic facial animation from speech.
- Multimedia applications aimed at handicapped persons, particularly those with poor vision and the elderly, are a rich field of endeavor in current research.
- Digital fashion aims to develop smart clothing that can communicate with other such enhanced clothing using wireless communication, so as to artificially enhance human interaction in a social setting. The vision here is to use technology to allow individuals to allow certain thoughts and feelings to be broadcast automatically, for exchange with others equipped with similar technology.
- Georgia Tech's Electronic Housecall system, an initiative for providing interactive health monitoring services to patients in their homes, relies on networks for delivery, challenging current capabilities.

5. Applications of Multimedia

Multimedia finds its application in various areas including, but not limited to, advertisements, art, education, entertainment, engineering, medicine, mathematics, business, scientific research and spatial, temporal applications. multimedia such as virtual reality, or VR. Goggles, helmets, special gloves.

A few application areas of multimedia are listed below:

Creative industries: Creative industries use multimedia for a variety of purposes ranging from fine arts, to entertainment, to commercial art, to journalism, to media and software services provided for any of the industries listed below. An individual multimedia designer may cover the spectrum throughout their career. Request for their skills, range from technical to analytical and to creative.

Commercial: Much of the electronic old and new media utilized by commercial artists is multimedia. Exciting presentations are used to grab and keep attention in advertising. Industrial, business to business, and interoffice communications are often developed by creative services firms for advanced multimedia presentations beyond simple slide shows to sell ideas or liven-up training. Commercial multimedia developers may be hired to design for governmental services and non-profit services applications as well.

Entertainment and Fine Arts: In addition, multimedia is heavily used in the entertainment industry, especially to develop special effects in movies and animations. Multimedia games are a popular pastime and are software programs available either as CD-ROMs or online. Some video games also use multimedia features. Multimedia applications that allow users to actively participate instead of just sitting by as passive recipients of information are called Interactive Multimedia.

Education: In Education, multimedia is used to produce computer-based training courses (popularly called CBTs) and reference books like encyclopaedia and

almanacs. A CBT lets the user go through a series of presentations, text about a particular topic, and associated illustrations in various information formats. Edutainment is an informal term used to describe combining education with entertainment, especially multimedia entertainment.

Engineering: Software engineers may use multimedia in Computer Simulations for anything from entertainment to training such as military or industrial training. Multimedia for software interfaces are often done as collaboration between creative professionals and software engineers.

Industry: In the Industrial sector, multimedia is used as a way to help present information to shareholders, superiors and coworkers. Multimedia is also helpful for providing employee training, advertising and selling products all over the world via virtually unlimited web-based technologies.

Medicine: In Medicine, doctors can get trained by looking at a virtual surgery or they can simulate how the human body is affected by diseases spread by viruses and bacteria and then develop techniques to prevent it.

Multimedia in Public Places: In hotels, railway stations, shopping malls, museums, and grocery stores, multimedia will become available at stand-alone terminals or kiosks to provide information and help. Such installation reduce demand on traditional information booths and personnel, add value, and they can work around the clock, even in the middle of the night, when live help is off duty. A menu screen from a supermarket kiosk that provide services ranging from meal planning to coupons. Hotel kiosk list nearby restaurant, maps of the city, airline schedules, and provide guest services such as automated checkout. Printers are often attached so users can walk away with a printed copy of the information. Museum kiosk are not only used to guide patrons through the exhibits, but when installed at each exhibit, provide great added depth, allowing visitors to browser though richly detailed information specific to that display.

Example Multimedia Applications

- ✓ World Wide Web
- ✓ Multimedia Authoring, e.g. Adobe/Macromedia Director
- ✓ Hypermedia courseware
- ✓ Video-on-demand
- ✓ Interactive TV
- ✓ Computer Games
- ✓ Virtual reality
- ✓ Digital video editing and production systems
- ✓ Multimedia Database systems

6. Multimedia on the Web

The World Wide Web is the largest and most commonly used hypermedia application. Its popularity is due to:

- the amount of information available from web servers,
- the capacity to post such information,
- and the ease of navigating such information with a web browser.

WWW technology is maintained and developed by the World Wide Web Consortium (W3C). The W3C has listed the following three goals for the WWW: universal access of web resources (by everyone everywhere), effectiveness of navigating available information, and responsible of posted material.

HyperText Transfer Protocol (HTTP)

HTTP is a protocol that was originally designed for transmitting hypermedia, but it also supports transmission of any file type. HTTP is a "stateless" request/response protocol, in the sense that a client typically opens a connection to the HTTP server, requests information, the server responds, and the connection is terminated - no information is carried over for the next request.The Uniform Resource Identifier (URI) identifies the resource accessed, such as the host name,

always preceded by the token "http://". A URI could be a Uniform Resource Locator (URL).

HyperText Markup Language (HTML)

HTML is a language for publishing hypermedia on the World Wide Web. It is defined using SGML and derives elements that describe generic document structure and formatting. Since it uses ASCII, it is portable to all different (even binary-incompatible) computer hardware, which allows for global exchange of information.

Extensible Markup Language (XML)

There is a need for a markup language for the WWW that has modularity of data, structure, and view. That is, we would like a user or an application to be able to define the tags (structure) allowed in a document and their relationship to each other, in one place, then define data using these tags in another place (the XML file) and, finally, define in yet another document how to render the tags.

7. Multimedia Data Basics

Multimedia systems/applications have to deal with the

- Generation of data
- Manipulation of data
- Storage of data
- Presentation of data
- Communication of information/data

Static or Discrete Media: Some media is time independent: Normal data, text, single images and graphics are examples.
Continuous Media: Time dependent Media: Video, animation and audio are examples.

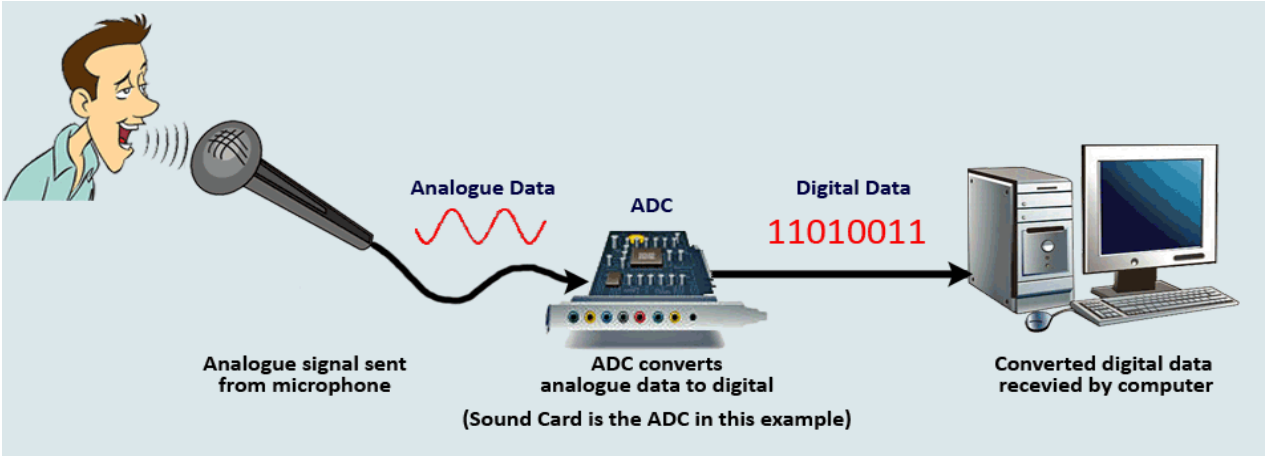
Analog and Digital Signal Conversion

The world we sense is full of analog signals: Electrical sensors convert the medium they sense into electrical signals

- E.g. transducers, thermocouples: temperature sensor, microphones: acoustic sensor
- Cameras (Video): light sensor. (usually) continuous Analog signals (e.g. Sound and Light)

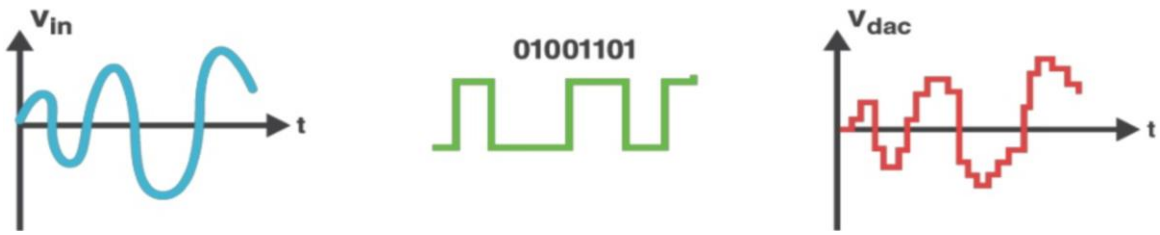
Analog: continuous signals must be converted or digitised for computer processing.

Digital: discrete digital signals that computer can readily deal with. Special hardware devices: Analog-to-Digital Playback {a converse operation to Analog-to-Digital converters. E.g. Audio: Take analog signals from analog sensor (e.g. microphone)



Analog-to-Digital-to-Analog Pipeline

• Begins at the conversion from the analog input and ends at the conversion from the output of the processing system to the analog output as shown:



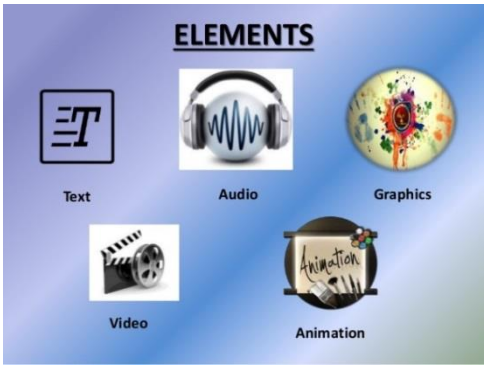
Multimedia Data: Input and Format

How to capture and store each Media format?

- Note that text and graphics (and some images) are mainly generated directly by computer/device (e.g. drawing/painting programs) and do not require digitising: They are generated directly in some (usually binary) format.
- Printed text and some handwritten text can be scanned via Optical Character Recognition.
- Handwritten text could also be digitised by electronic pen sensing.
- Printed imagery/graphics can be scanned directly to image formats.

Multimedia data may be in a **variety of formats**:

1- **Text**
2- **Graphics**
3- **Images**
4- **Audio**
5- **Video**



1) Text and Static Data

- Source: keyboard, speech input, optical character recognition, data stored on disk.
- Stored and input character by character:
- Storage: 1 byte per character (text or format character), e.g. ASCII; more bytes for Unicode. For other forms of data (e.g. Spread sheet les). May store as text (with formatting, e.g. CSV {Comma-Separated Values) or may use binary encoding.
- Formatted Text: Raw text or formatted text e.g HTML, Rich Text Format (RTF), Word or a program language source (Java, Python, MATLAB etc.)
- Data Not temporal | BUT may have natural implied sequence e.g. HTML format sequence, Sequence of Java program statements.

- Compression: convenient to bundle les for archiving and transmission of larger les. E.g. Zip, RAR, 7-zip.

2) Graphics

- Format: constructed by the composition of primitive objects such as lines, polygons, circles, curves and arcs.
- Input: Graphics are usually generated by a graphics editor program (e.g. illustrator, Freehand) or automatically by a program (e.g. Postscript).
- Graphics input devices: keyboard (for text and cursor control), mouse, trackball or graphics tablet.
- Graphics are usually selectable and editable or revisable (unlike images).
- Graphics les usually store the primitive assembly.
- Do not take up a very high storage overhead.
- Graphics standards: Open Graphics Library, a standard specification defining a cross-language, cross-platform API for writing applications that produce 2D/3D graphics.
- Animation: can be generated via a sequence of slightly changed graphics.
- 2D animation: e.g. Flash | Key frame interpolation: tweening: motion & shape.

  3D animation: e.g. Maya.
- Change of shape/texture/position, lighting, camera Graphics animation is compact
- Suitable for network transmission (e.g. Flash).

3) Images

- Still pictures which (uncompressed) are represented as a bitmap (a grid of pixels).
- Input: scanned for photographs or pictures using a digital scanner or from a digital camera.
- Input: May also be generated by programs similar to graphics or animation programs.
- Analog sources will require digitising.
- Compression is commonly applied.

- Can usually only edit individual or groups of pixels in an image editing application, e.g. photoshop.

4) Audio

- Audio signals are continuous analog signals.
- Input: microphones and then digitised and stored.
- CD Quality Audio requires 16-bit sampling at 44.1 KHz.
- Usually compressed (E.g. MP3, AAC, Flac, Ogg Vorbis).

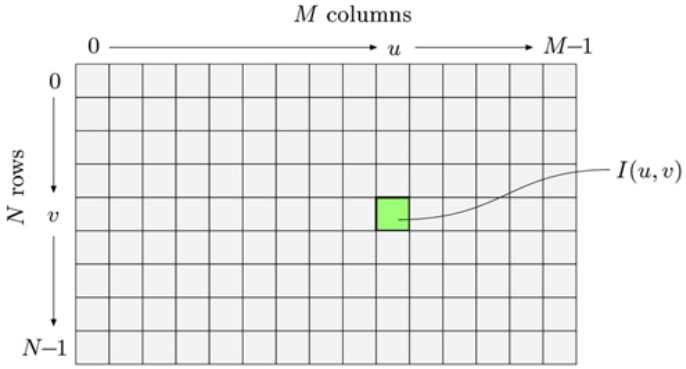
5) Video

- Input: Analog Video is usually captured by a video camera and then digitised, although digital video cameras now essentially perform both tasks.
- There are a variety of video (analog and digital) formats.
- Raw video can be regarded as being a series of single images. There are typically 25, 30 or 50 frames per second.

المحور الثاني

8 & 9 : Graphics and Image Data Representation

The image data structure is a 2D array of pixel values as shown in Figure below.



Digital Images

An image must be converted to numerical form before processing. This conversion process is called digitization, and a common form is illustrated in Figure (1). The image is divided into small regions called ***picture elements***, or ***pixel*** for short. The most common subdivision scheme is the rectangular sampling grid shown in Figure (1). The image is divided into horizontal lines made up of adjacent pixels.

At each pixel location, the image brightness are sampled and quantized. This

step generates an integer at each pixel representing the brightness or darkness of the image at that point. When this has been done for all pixels, the image is represented by a rectangular array of integer. Each pixel has a location or address (Line or row number and sample or column number) and an integer value called gray level. This array of digital data is now a candidate for computer processing.

A digital image is a numeric representation of a two-dimensional image. Depending on whether the image resolution is fixed. The term "digital image" usually refers to **raster images or bitmapped images**.

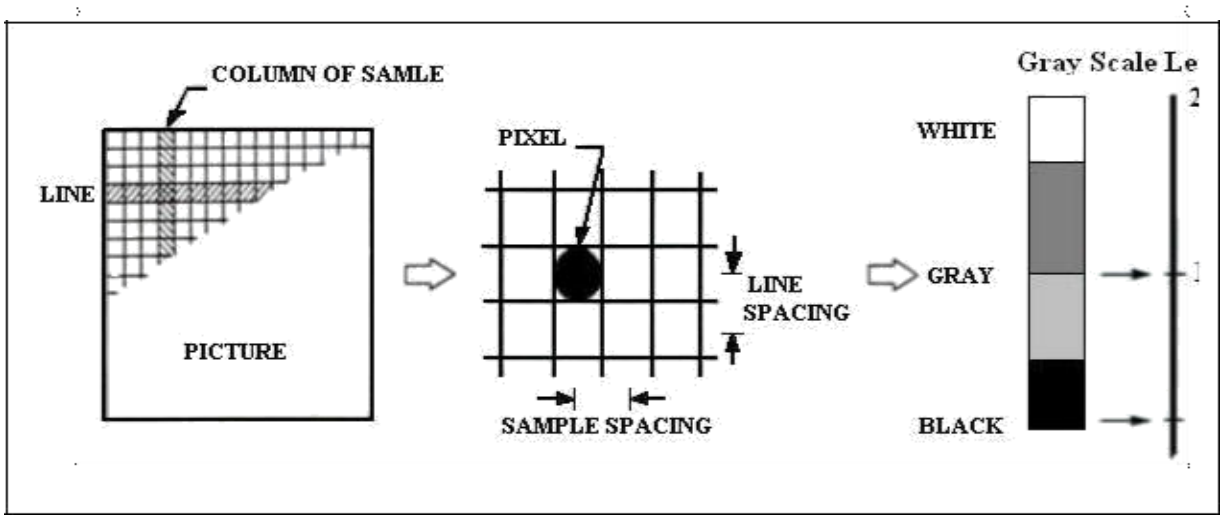


Figure Digitizing an Image

Thus a digital image is now a two-dimensional rectangular array of quantized sample value.

A digital image consists of many picture elements, termed pixels. The number of pixels determine the quality of the image (resolution).

Image Representation (Data Structures bitmap representation)

As we know, the human visual system receives an input image as a collection of spatially distributed light energy; this form is called an optical image. Optical images are the types we deal with every day - cameras capture 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 *f(x,y)*.

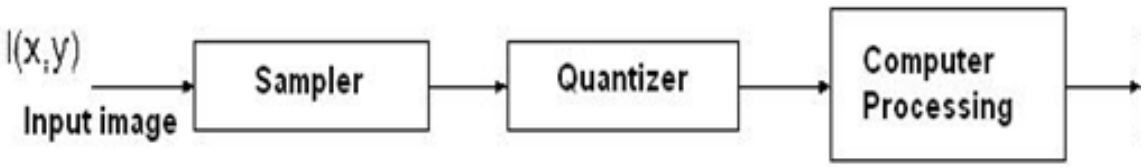
The digital image *f(x,y)* is represented as a two-dimensional array of data,

where each pixel value corresponds to the brightness of the image at the point *(x,y)*. In linear algebra terms, a two-dimensional array like our image model *f(x,y)* is referred to as a matrix, and one row (or column) is called a vector. This image model is for monochrome (one-color, this is what we normally refer to as black and white) image data, we also have other types of image data that require extensions or modifications to this model.

10. Image Digitalization

Typical image processing scenario

- The image is represented by a function of two continuous variables, spatial coordinates (x, y).
- The first function is ***sampling***: we sample the continuous image signal (discretization of the spatial coordinates).
- The result is then injected into the ***Quantization*** function: for a monochrome image, we often choose 256 luminance levels.
- The image is now digitalized, so we can apply the required image processing, which could be binarization, contrast enhancement, etc.

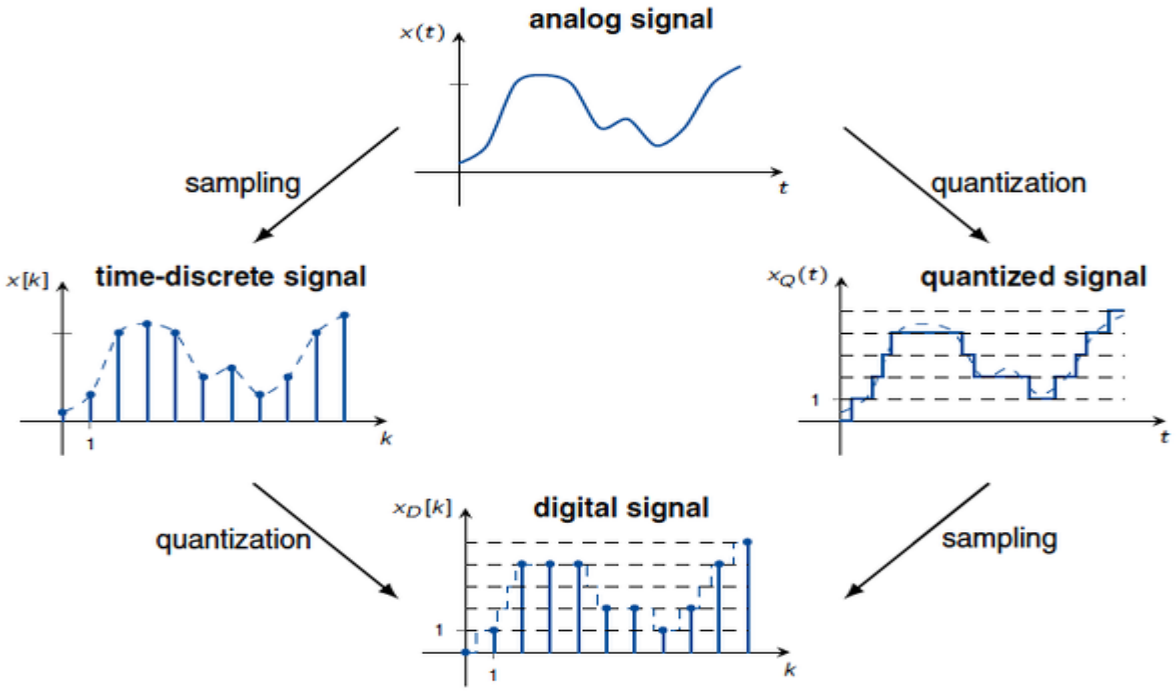


Hence, in order to create an image which is digital, we need to covert continuous data into digital form. There are two steps in which it is done:

1- Sampling

2- Quantization

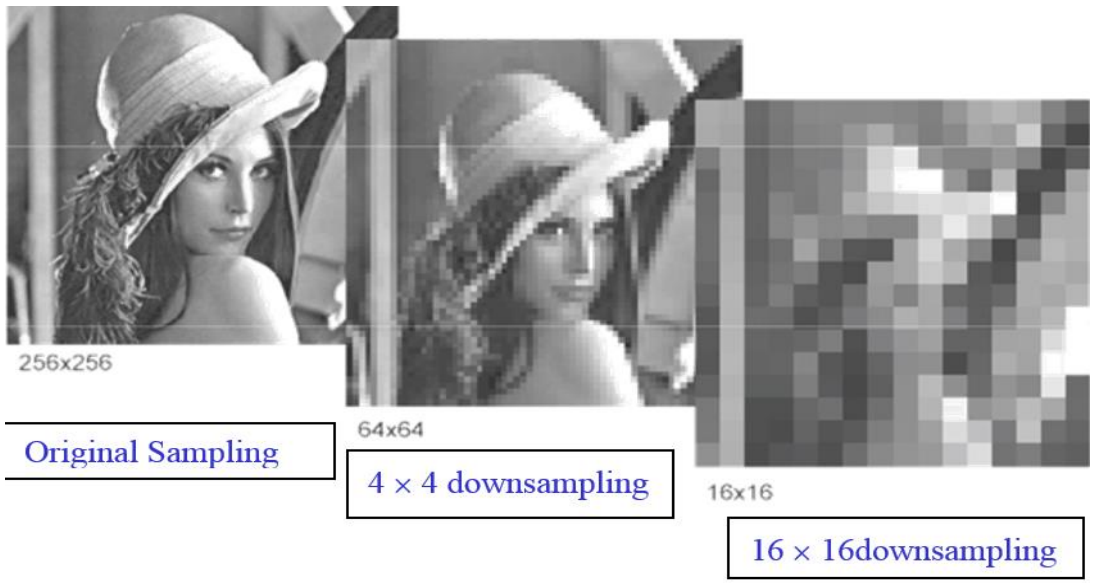
The sampling rate determines the spatial resolution of the digitized image, while the quantization level determines the number of gray levels in the digitized image. A magnitude of the sampled image is expressed as a digital value in image processing. The transition between continuous values of the image function and its digital equivalent is called quantization.



The number of quantization levels should be high enough for human perception of fine shading details in the image. The occurrence of false contours is the main problem in an image which has been quantized with insufficient brightness levels.

1. Sampling (represents => resolution)

Sampling: is a process of measuring the value of the image function *f (x, y)* at discrete ***intervals in space***. Each sample corresponds to a small square area of the image, known as a pixel. two-dimensional pattern to represent the measurements (light intensity or color) that are made in the form of an image numerically.



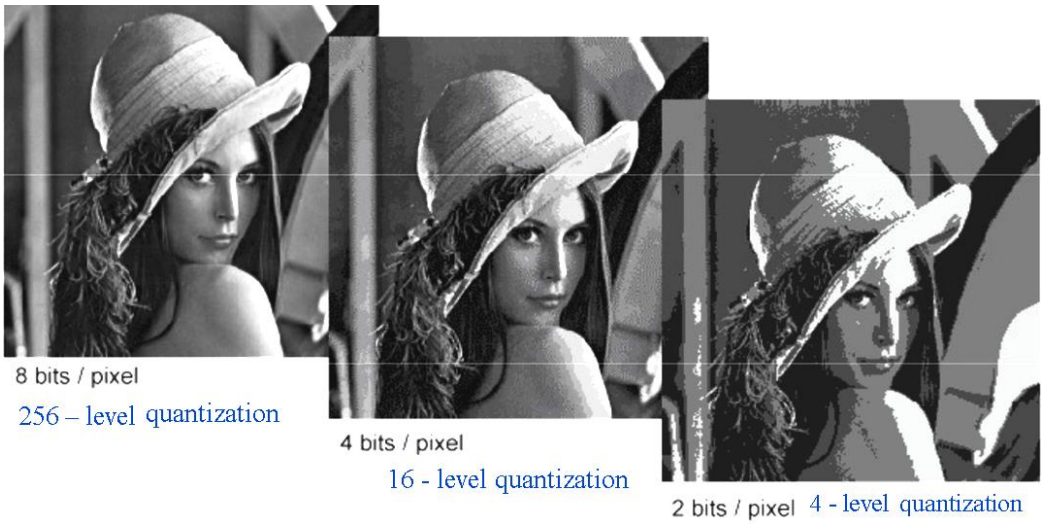
Examples: Image Sampling

In this figure, we have represented the image "Lena" sampled with two different sampling structures. The image on the left is the reference image (spatial dimensions: (256*256) pixels). The second image is sampled with a sampling frequency four times lower for each of the two spatial dimensions. This means it is (64*64) pixels. For display purposes, it has been brought to the same size as the original using a zoom. This is in fact an interpolation of zero- order (each pixel is duplicated 4*4 times, so that on the screen it displays a square of (4*4) identical pixels).

2. Quantization

Quantization: is the process of converting a continuous range of values into a finite range of discreet values. The accuracy with which variations in $f(x, y)$ are represented is determined by The number of quantization levels that we use: the more levels we use, the better the approximation.

As number of bits to represent a pixel intensity is quantized. Suppose 8 bit is used for a pixel, it's equivalent value ranges from 0 to 255 (discrete values). 0 is assigned to pure Black, and 255 is assigned to pure White. Intermediate values are assigned to gray scales as shown in this image.



Quantization of an image

This illustration shows examples of a quantization carried out on the image :

For the image on the left: quantization is followed by a natural binary coding with 8 bits per pixel. There are $2^8 = 256$ reconstruction levels to represent the magnitude of

each pixel. It is the typical case of a monochrome image (only in gray scales).

For the middle image: quantization is carried out with a 4 bits per pixel coding, giving $2^4 = 16$ reconstruction levels. Contours are well rendered but textures are imprecise in some cases. These are areas in the signal with a weak spatial variation, which suffer more visually due to the appearance of false contours (loss on the face and the shoulder).

For the image on the right: quantization is carried out with a 2 bits per pixel coding, so we have $2^2 = 4$ reconstruction levels. The deterioration seen on the previous image is even more flagrant here.

11. Spatial resolution and quantization

What is size of the image? And what is a resolution?

Resolution = width x height
Image Size = width x height x No. of bit per pixel

Quantization = Number of bits per pixel (***Quantization***)

Spatial resolution can be defined as the smallest discernible detail in an image. In short, what spatial resolution refers to is that we cannot compare two different types of images to see that which one is clear or which one is not. If we have to compare the two images, to see which one is more clear or which has a more spatial resolution, we have to compare two images of the ***same size***.

Spatial resolution is determined by the ***sampling process***. The spatial resolution of a digital image reflects the amount of details that one can see in the image (i.e. the ratio of pixel "area" to the area of the image display). If an image is spatially sampled at *X* pixels, then the larger *X* the finer the observed details. Or in other way we can define spatial resolution as the number of independent pixels per inch (PPI).

Effect of reducing the spatial resolution

Decreasing spatial resolution of a digital image, within the same area, may result in what is known as *checkerboard pattern*. Also image details are lost when the spatial resolution is reduced.

To demonstrate the checkerboard pattern effect, we subsample the 1024×1024 image shown in Figure (1) to obtain the image of size 512×512 pixels. The 512×512 is then subsampled to 256×256 image, and so on until 32×32 image. The subsampling process means deleting the appropriate number of rows and columns from the original image. The number of allowed gray levels ***was kept*** at 256 in all the images.

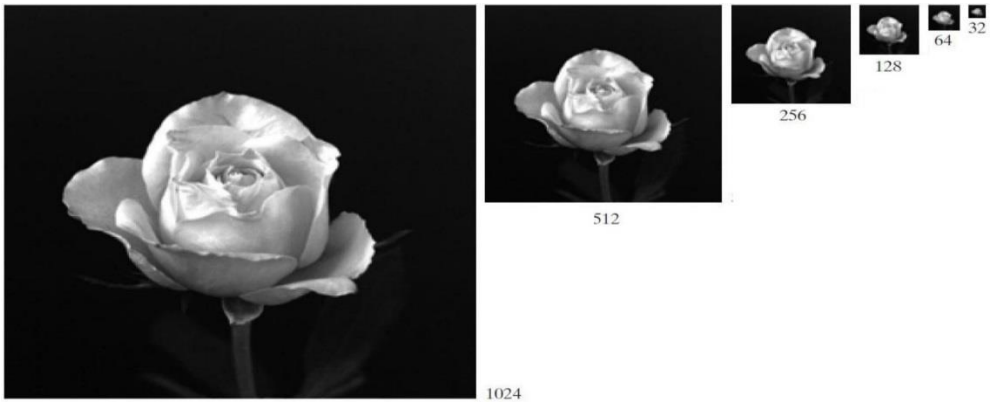


Figure : A 1024×1024, 8-bit image Sub sampled down to size 32×32 pixels.

To see the effects resulting from the reduction in the number of samples, we bring all the subsampled images up to size 1024×1024 by row and column pixel replication. The resulted images are shown in Figure.

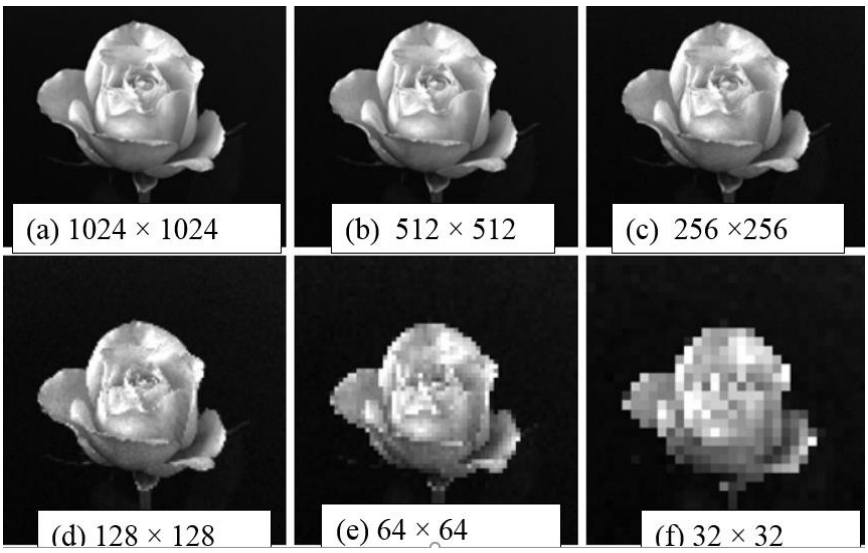


Figure 2. the effects of resulting from the reduction in the number of samples. All images have 8 – bits.

Compare Figure 2(a) with the 512×512 image in Figure 2(b), we find that the level of detail lost is simply too fine to be seen on the printed page at the scale in which these images are shown. Next, the 256×256 image in Figure 2(c) shows a very slight fine checkerboard pattern in the borders between flower petals and the black background. A slightly more pronounced graininess throughout the image also is beginning to appear. These effects are much more visible in the 128×128 image in Figure 2(d), and they become pronounced in the 64×64 and 32×32 images in Figures 2(e) and (f), respectively.

12. Type of image

The image types we will consider are :

1) Binary 2) grayscale 3) color 4) multispectral.

The more general term pixmap refers to a map of pixels, where each one may store more than two colors, thus using more than one bit per pixel. Often bitmap is used for this as well

1. Binary Images (Black and White Images) (monochrome image)

Binary images are the simplest type of images and can take on 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 applications where the only information required for the task is general shape or outline, information.

Binary images are often created from gray-scale images via a threshold operation where every pixel above the threshold value is turned white ('1'), and those below it are turned black ('0').

- Each pixel is stored as a single bit (0 or 1).
- A 640 x 480 bit-mapped image requires 37.5 KB of storage.

2. Gray-Scale Images

Gray-scale images are referred to as monochrome, or one-color, images. They contain brightness information only, no color information. The number of bits used for each pixel determines the number of different brightness levels Available. The typical image contains 8 bits/pixel data, which allows us to have 256 (0-255) different brightness (gray) levels.

This representation provides more than adequate brightness resolution, in terms of the human visual system's requirements and provides a "noise margin" by allowing for approximately twice as many gray levels as required. Additionally, the 8 bit representation is typical due to the fact that the byte which corresponds to 8 bit of data, is the standard small unit in the world of digital computers.

Each pixel is usually stored as a byte (value between 0 to 255). A dark pixel may have a value of 10; a bright one may be 240 (**dark=0; white=255**)

Example: find image size with 640*480 pixles

Total no. of bits = 640*480* 8 bit=2457600 bit Or 640*480*1 Byte=307200Bytes

Convert to Byte: 2457600/8=307200 Byte

Convert to KByte: 307200 Byte/1024 = 300 KB

3. Color Images

Color images can be modeled as three-band monochrome image data, where each band of data corresponds to a different color. The actual information stored in the digital image data is the 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 bits/pixel 8-bits for each of the three color bands (Red, Green, and Blue). In Figure (2-a) we see a representation of a typical RGB color image. Figure (2-a) illustrates that, in addition to referring to a row or column as a vector, we can refer to a single pixel's Red, Green, and Blue values as a color pixel vector (R, G, B).

For many applications, RGB color information is transformed into a mathematical space that decouples the brightness information from the color information. After this is done, the image information consists of a one-dimensional brightness, or luminance, space and a two-dimensional color space. Now the two-dimensional color space does not contain any brightness information; but it typically contains information regarding the relative amounts of the different colors. An additional benefit of modeling the color information in this manner is that it creates a more people-oriented way of describing the colors.

24-Bit Color Images

How many colors in RGB model?

In a color 24-bit image, each pixel is represented by three bytes, usually representing RGB. Since each value is in the range 0–255, this format supports 256 × 256 × 256, or a total of 16,777,216, possible combined colors. However, such flexibility does result in a storage penalty: a 640 × 480 24-bit color image would require 921.6 kB of storage without any compression. Color images are actually stored as 32-bit images, with the extra byte of data for each pixel used to

store an alpha value representing special effect information (e.g., transparency).

Examples: Color Image uses 32 bits with high = 200 and width = 200 pixels?

Size= 200*200*4 Bytes=160000B/1024=156.25KB.

4. Multispectral Images

Multispectral images typically contain information outside the normal human perceptual range. This may **include** infrared, ultraviolet, X-ray, acoustic, radar data. These are not images in the usual sense because the information represented is not directly visible by the human system. However, the information is often represented in visual form by mapping the different spectral bands to RGB components. If more than three bands of information are in the multispectral image, the dimensionality is reduced by applying a principal component's transform.

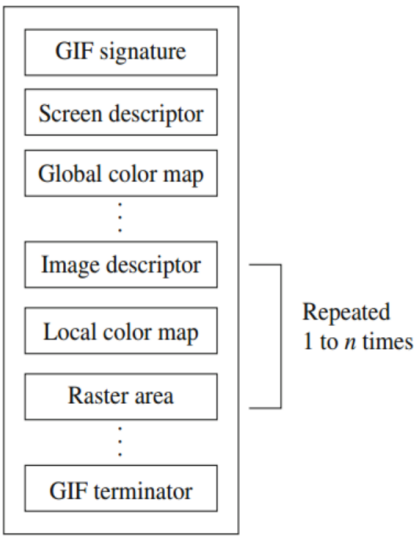
13. Image file formats

Some popular file formats are listed in the table below with their respective extensions:

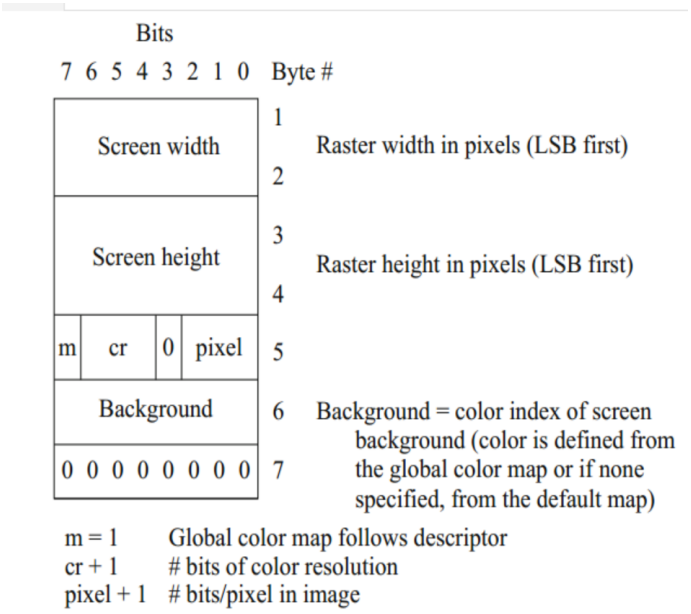
Format Name	Description	Recognized Extensions
GIF	Graphics Interchange Format	.gif
JPEG	Joint Photographic Experts Group	.jpg, .jpeg
TIFF	Tagged Image File Format	.tif, .tiff
BMP	Windows Bitmap	.bmp
PNG	Portable Network Graphics	.png

1- GIF

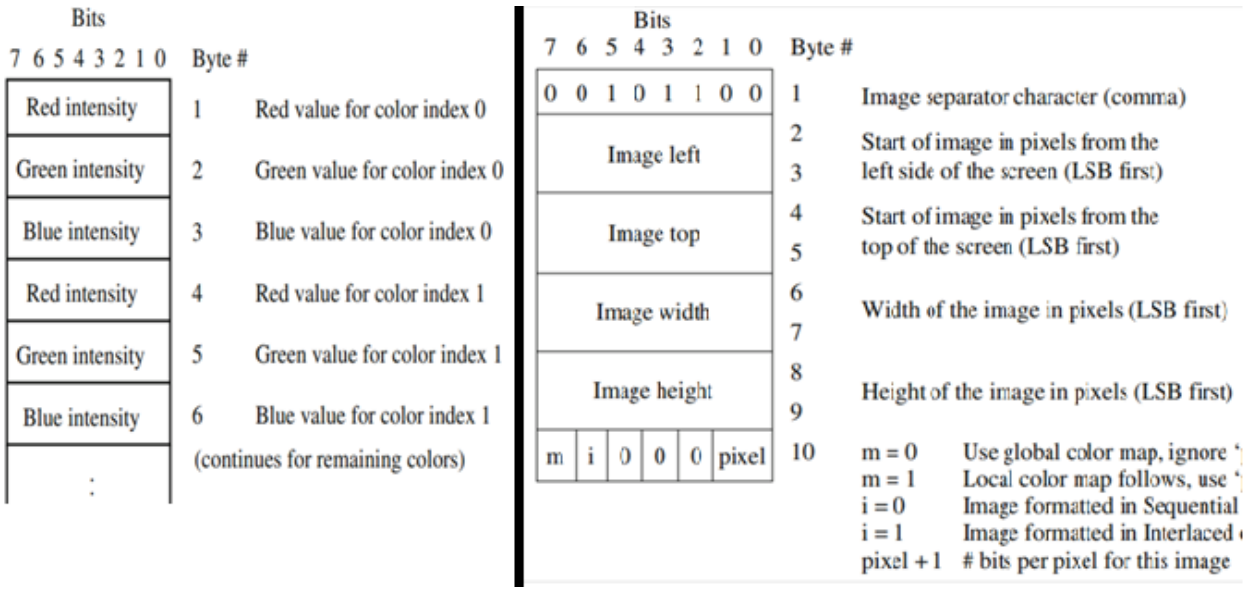
- Graphics Interchange Format (GIF) devised by the UNISYS Corp. and CompuServe, initially for transmitting graphical images over phone lines via modems.
- One of the simplest is the 8-bit GIF format, and we study it because it is easily understood, and also because of its historical connection to the WWW and HTML markup language as the first image
- type recognized by net browsers.
- Limited to only 8-bit (256) colour images, suitable for images with few distinctive colours (e.g., graphics, drawing)
- GIF89a: supports simple animation, transparency index etc.



GIF file format



Screen descriptor



Color map *Image descriptor*

2- JPEG Standard

- A standard for photographic image compression created by the Joint Photographic Experts Group
- Takes advantage of limitations in the human vision system to achieve high rates of compression.
- Lossy compression which allows user to set the desired level of quality/compression.

3- TIFF

- Tagged Image File Format stores many dierent types of images (e.g., bit-map, greyscale, 8-bit & 24-bit RGB, etc.).
- Developed by the Aldus Corp. in the 1980's and later supported by the Microsoft
- TIFF is a typically lossless format.
- It does not provide any major advantages over JPEG and is not as user-controllable it appears to be declining in popularity

4- BMP

- BitMap (BMP) is the major system standard graphics file format for Microsoft Windows.
- used in Microsoft Paint and other programs. It makes use of run-length encoding compression and can fairly efficiently store 24-bit bitmap images.
- Note, however, that BMP has many different modes, including uncompressed 24-bit images.

5- PNG

PNG meant to supersede GIF standard

Features of PNG:

- Support up to 48 bits per pixel - more accurate colors
- Support description of gamma-correction and
- alpha-channel for controls such as transparency
- Support progress display through 8×8 blocks.

14. Arithmetic and Logical Operations on Images (Image Algebra)

These operations are applied on pixel-by-pixel basis. So, to add two images together, we add the value at pixel (0 , 0) in image 1 to the value at pixel (0 , 0) in image 2 and store the result in a new image at pixel (0 , 0). Then we move to the next pixel and repeat the process, continuing until all pixels have been visited.

Clearly, this can work properly only if the two images have identical dimensions. If they do not, then combination is still possible, but a meaningful result can be obtained only in ***the area of overlap***. If our images have dimensions of w_1*h_1, and w_2*h_2 and we assume that their origins are aligned, then the new image will have dimensions w*h, where:

$$w = min\ (w_1, w_2)$$

$$h = min\ (h_1, h_2)$$

Addition and Averaging

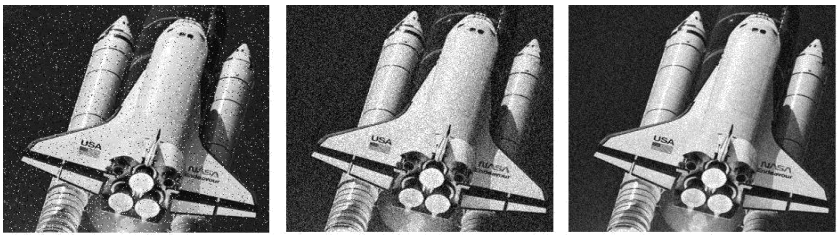
If we add two 8-bit gray scale images, then pixels in the resulting image can have values in the range 0-510. We should therefore choose a 16-bit representation for the output image or divide every pixel value by two. If we do the later, then we are computing an average of the two images.

The main application of image averaging is ***noise removal***. Every image acquired by a real sensor is afflicted to some degree of random noise. However, the level of noise is represented in the image can be reduced, provided that the scene is static and unchanging, by the averaging of multiple observations of that scene. This works because the noisy distribution can be regarded as approximately symmetrical with a mean of zero. As a result, positive perturbations of a pixel's value by a given amount are just as likely as negative perturbations by the same amount, and there will be a tendency for the perturbations to cancel out when several noisy values are added.

Addition can also be used to ***combine the information of two images***, such as an image morphing, in motion pictures.

Algorithm 1: image addition

read input-image1 into in-array1;
read input-image2 into in- array2;
for i = 1 to no-of-rows do
for j=1 to no-of-columns do begin
out-array (i,j) = in-array1(i,j) + in-array2(i,j);
if (out-array (i,j) > 255) then out-array (i,j) = 255;
end
write out-array to out-image;



a *b* *c*

Figure (4) a) noisy image b) average of five observation c) average of ten observation

Subtraction

Subtracting two 8-bit grayscale images can produce values between - 225 and +225. This necessitates the use of 16-bit signed integers in the output image – unless sign is unimportant, in which case we can simply take the modulus of the result and store it using 8-bit integers:

$g(x,y) = |f_1(x,y) - f_2(x,y)|$

The main application for image subtraction is in ***change detection*** (or ***motion detection***). If we make two observations of a scene and compute their difference using the above equation, then changes will be indicated by pixels in the difference image which have ***non-zero values***. Sensor noise, slight changes in illumination and various other factors can result in small differences which are of no significance so it is usual to apply a threshold to the difference image. Differences below this threshold are set to zero. Difference above the threshold can, if desired, be set to the maximum pixel value. Subtraction can also be used in ***medical imaging to remove static***

background information.

Algorithm2: image subtraction

```
read input-image1 into in-array1;
read input-image2 into in- array2;
for i = 1    to no-of-rows do
    for j=1 to no-of-columns do
        begin
            out-array (i,j) = in-array1(i,j) - in-array2(i,j);
            if ( out-array (i,j) < 0 ) then out-array (i,j) = 0; end
write out-array to out-image;
```

Figure (5)a, b) two frames of video sequence c) their difference

Multiplication and Division

Multiplication and division can be used to adjust brightness of an image. Multiplication of pixel values by a number greater than one will brighten the image, and division by a factor greater than one will darken the image. Brightness adjustment is often used as a ***preprocessing step*** in image enhancement.

One of the principle uses of image multiplication (or division) is to ***correct grey-level shading*** resulting from non uniformities in illumination or in the sensor used to acquire the image.

(a) *(b)* *(c)*

Figure a) original image b) image multiplied by 2 c) image divided by 2

15. Logical Operation:

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			
Input 1	1	1	0	0		1	1	0	0		1	1	0	0
Input 2	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 from an image.

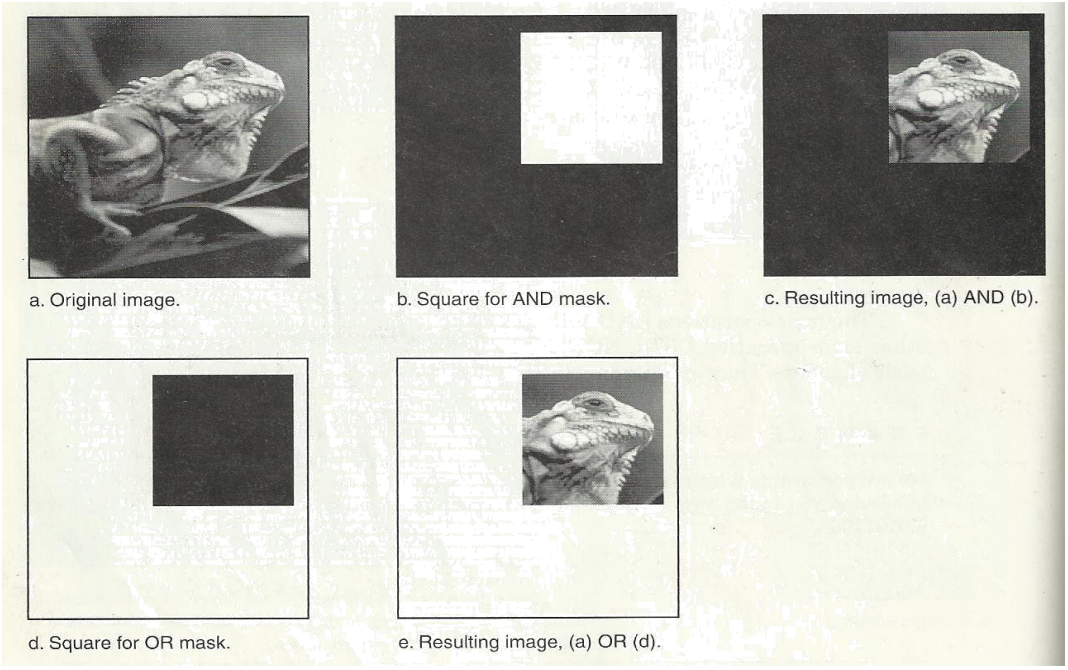


Figure: image masking

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).

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

Exclusive OR ⊗

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

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

NOT

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

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

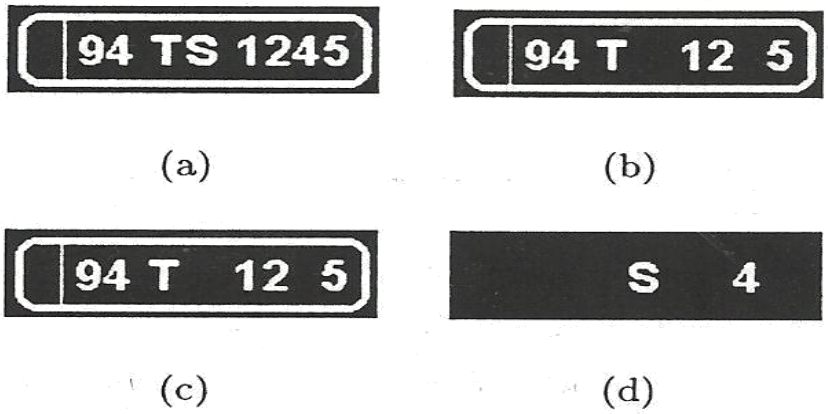


Figure a) input image a(x,y); b) input image b(x,y) ; c) a(x,y) ^ b(x,y) ; d) a(x,y) ^ ~ b(x,y)

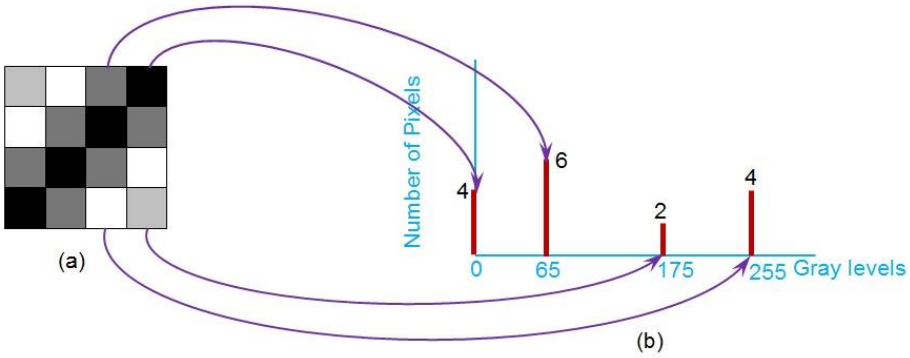
16. Image Histogram

A histogram is a graph that shows the frequency of anything.

A histogram is an accurate representation of the distribution of numerical data. It is an estimate of the probability distribution of a continuous variable (quantitative variable). A histogram is a graph showing the number of pixels in an image at each different intensity value found in that image. It's a bar chart of the count of pixels of every tone of gray that occurs in the image.

For an 8-bit grayscale image there are 256 different possible intensities, and

so the histogram will graphically display 256 numbers showing the distribution of pixels amongst those grayscale values.



The gray level ***histogram*** is showing, the gray level, for each pixel in the image.

> The histogram of an image records the frequency distribution of gray levels in the image.

The histogram of an 8-bit image, can be though of as a table with 256 entries, or " bins", indexed from 0 to 255. in bin 0 we record the number of times a gray level of 0 occurs; in bin 1 we record the number of times a gray level of 1 occurs, and so on, up to bin 255.

An algorithm below shows how we can accumulate in a histogram from an image.

ALGORITHM: for Calculating image Histogram

```
create an array histogram.
For all gray levels , I,do
Histogram [I] =0
Endfor
For all pixels coordinates, x and y , do
Increment Histogram [ f (x,y) ] by 1
Endfor
```

- The histogram of a digital image with *L* total possible intensity levels in the range [*0, G*] is defined as the discrete function

$$h(r_k) = n_k$$

- Where r_k is the *k*th intensity level in the interval [*0,G*] and n_k is the number of pixels in the image whose intensity level is r_k.

Example: Figure shows an image and its histogram.

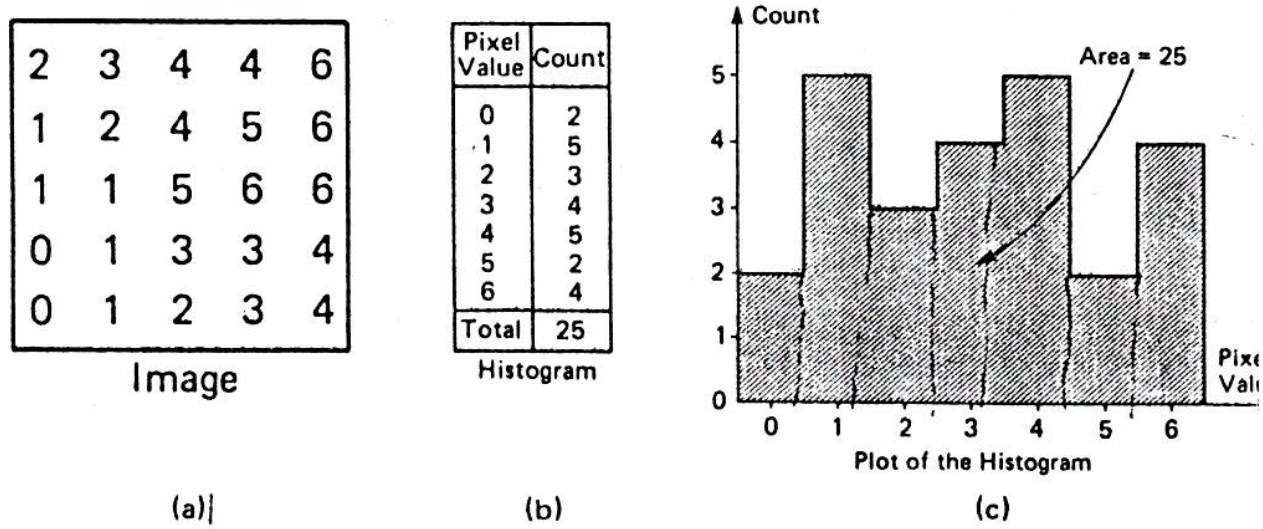


Figure: sub image and its histogram

The shape of the histogram provides us with information about the nature of the image, or sub image if we are considering an object in the image. For example, a ***very narrow*** histogram implies a low contrast, a histogram ***skewed toward the right*** implies a bright image, a histogram ***skewed toward the left*** implies a dark image, and a histogram with ***two major peaks***, implies an object that in contrast with the background.

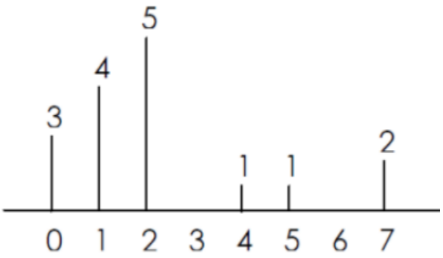
A color histogram counts pixels with a given pixel value in red, green, and blue (RGB). For example, in pseudocode, for images with 8-bit values in each of R, G, B, we can fill a histogram that has 256^3 bins:

```
int hist[256][256][256]; // reset to 0
//image is an appropriate struct
//with byte fields red,green,blue
for i=0..(MAX_Y-1)
for j=0..(MAX_X-1)
{
R = image[i][j].red;
G = image[i][j].green;
B = image[i][j].blue;
hist[R][G][B]++;
}
```

Example : Plot the Histogram of the following example with 4x4 matrix of a 3-bit image.

0	0	0	4
1	1	1	5
1	2	2	7
2	2	2	7

Answer



Properties and usage of histogram

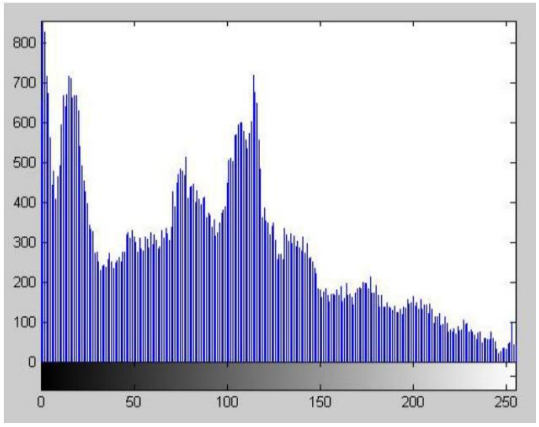
One of the ***principle use*** of the histogram is in the selection of ***threshold*** parameter.

The histogram of an image provides a useful indication of the relative importance of different gray levels in an image, indeed, it is sometimes possible to ***determine*** whether ***brightness*** or ***contrast*** ***adjustment*** is necessary merely by ***examining*** the ***histogram*** and ***not the image*** itself.

When an image is condensed into a histogram, ***all spatial information is discarded.*** The histogram specifies the number of pixels having each gray level but ***gives no hint*** as to ***where those pixels are located*** within the image. Thus the

histogram is ***unique*** for any particular image, but the ***reverse is not true***. Vastly different images could have identical histograms. Such operations as moving objects around within an image typically have no effect on the histogram.

Histograms are used in numerous image processing techniques, such as image enhancement, compression and segmentation.



Note that the horizontal axis of the histogram plot (Figure (b) above) represents gray level values, k, from 0 to 255. The vertical axis represents the values of $h(k)$ i.e. the number of pixels which have the gray level k.

It is customary to **"normalize"** a histogram by dividing each of its values by the total number of pixels in the image, i.e. uses the probability distribution (previously stated) as:

$$p(k) = \frac{h(k)}{M \times N}$$

Thus, $p(k)$ represents the probability of occurrence of gray level k.

As with any probability distribution:

- ***All the values of a normalized histogram*** $p(k)$ ***are less than or equal to 1.***
- ***The sum of all*** $p(k)$ ***values is equal to 1***

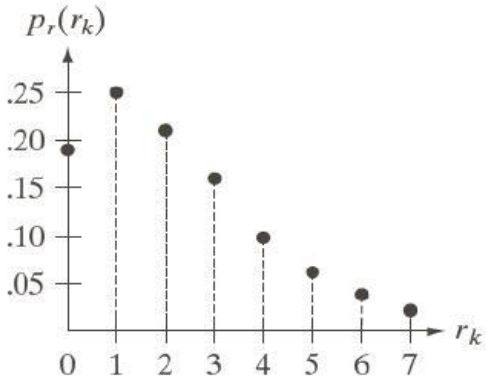
Another way of getting histogram is to plot pixel intensities vs. pixel probabilities. However, probability histogram should be used when comparing the histograms of images with different sizes.

Example: Suppose that a 3-bit image (L = 8) of size 64 × 64 pixels has the gray level (intensity) distribution shown in the table below. ***Perform normalized histogram.***

r_k	n_k
$r_0 = 0$	790
$r_1 = 1$	1023
$r_2 = 2$	850
$r_3 = 3$	656
$r_4 = 4$	329
$r_5 = 5$	245
$r_6 = 6$	122
$r_7 = 7$	81

Solution: $M \times N = 4096$. We compute the normalized histogram:

r_k	n_k	$p(r_k) = n_k/MN$
$r_0 = 0$	790	0.19
$r_1 = 1$	1023	0.25
$r_2 = 2$	850	0.21
$r_3 = 3$	656	0.16
$r_4 = 4$	329	0.08
$r_5 = 5$	245	0.06
$r_6 = 6$	122	0.03
$r_7 = 7$	81	0.02



Normalized histogram

17. Histogram modification and histogram equalization

An alternate perspective to gray-level modification that performs a similar function is referred to as histogram modification. The gray-level histogram of an image is the distribution of the gray levels in an image. In figure 11 we can see an image and its corresponding histogram. In general a histogram with a ***small-spread*** has a ***low-contrast***. And a histogram with a ***wide spread*** has a ***high contrast***, whereas an image with its histogram clustered at the ***low end*** of the range is ***dark***, and a histogram with the values clustered at the ***high end*** of the range corresponds to a ***bright*** image.

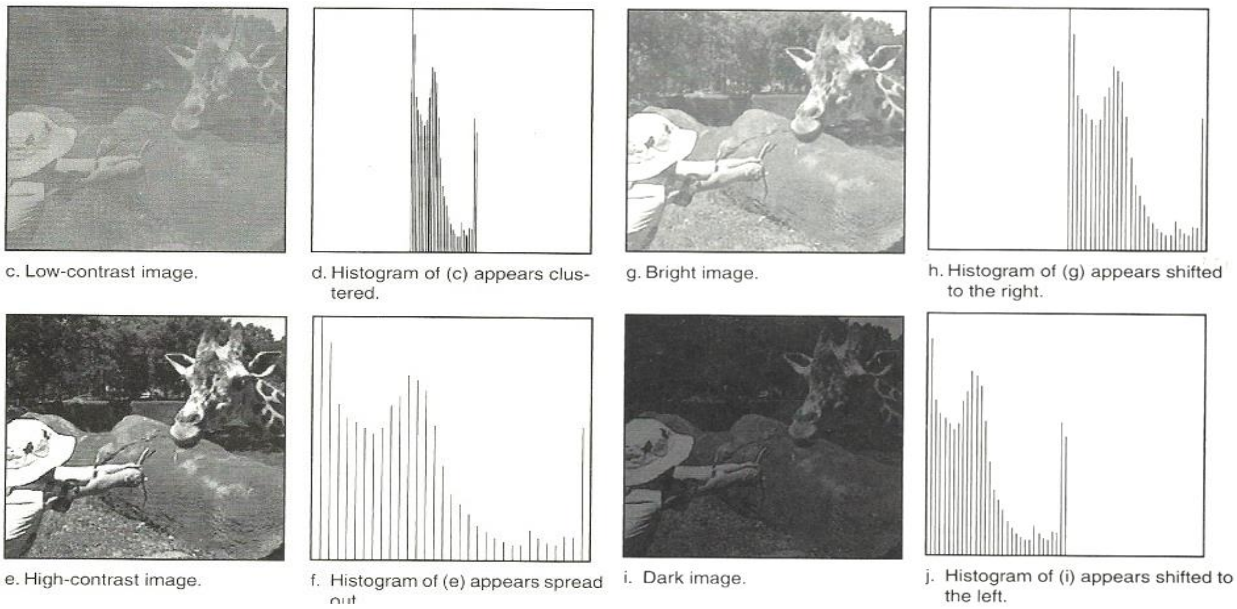


Figure 11: *a variety types of histograms*

The histogram can also be modified by a **mapping function**, which will either **stretch, shrink** (compress), or **slide** the histogram. Histogram stretching and histogram shrinking are forming a gray-level modification, sometimes referred to as histogram scaling. In figure **12** we see a graphical representation of histogram stretch, shrink, and slide.

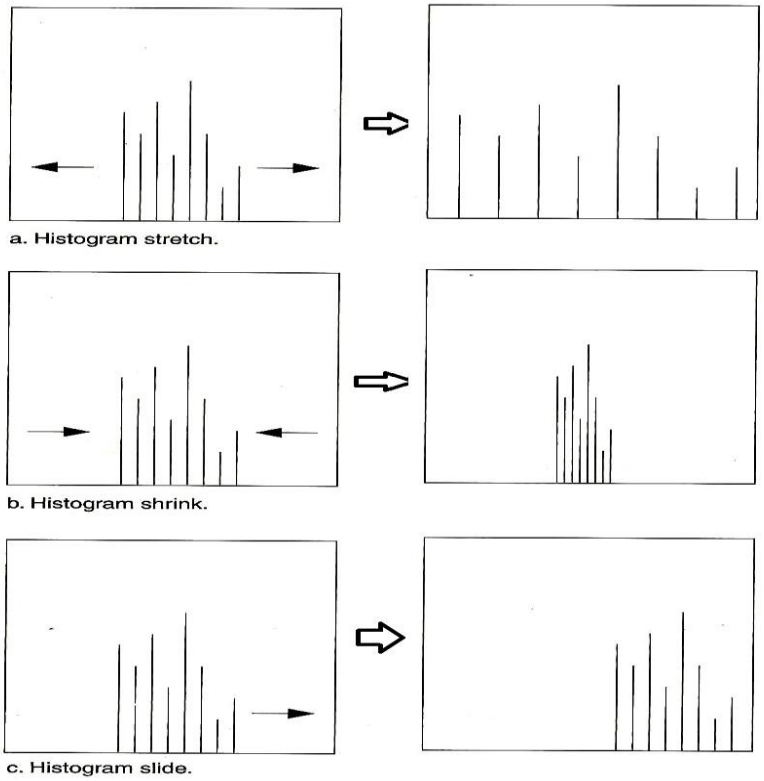


Figure 12: histogram modification

a. Histogram stretch

The mapping function for histogram stretch can be found by the equation :

$$\text{Stretch}(I(r,c)) = \left[\frac{I(r,c) - I(r,c)_{\text{MIN}}}{I(r,c)_{\text{MAX}} - I(r,c)_{\text{MIN}}}\right][\text{MAX} - \text{MIN}] + \text{MIN}$$

Where :

- ***I(r,c)$_{MAX}$*** *is the largest gray-level value in the image I(r,c)*
- ***I(r,c)$_{MIN}$*** *is the smallest gray-level value in the image I(r,c)*
- ***MAX*** *and* ***MIN*** *correspond to the maximum and minimum gray-level values possible (for 8-bitimages these are 0 and 255).*

This equation will take an image and stretch the histogram across the entire gray-level range, which has the effect of increasing the contrast of a low contrast image . ***If a stretch is desired over a smaller range, different MAX and MIN values can be specified.***

In general, histogram stretch will ***increase image contrast***.

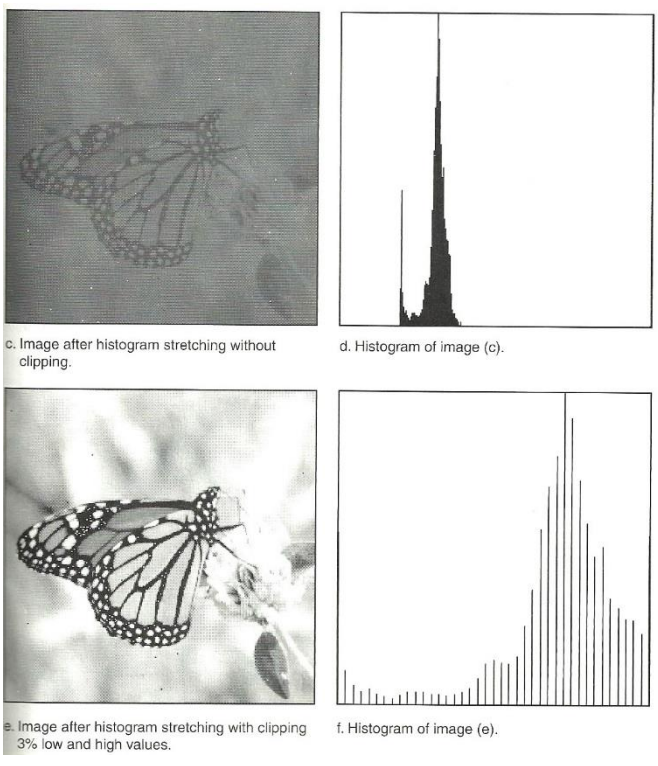


Figure 13: histogram stretch

b. Histogram shrink

The opposite of a histogram stretch is a histogram shrink, which will decrease image contrast by compressing the gray levels. The mapping function for a histogram shrink can be found by the following equation:

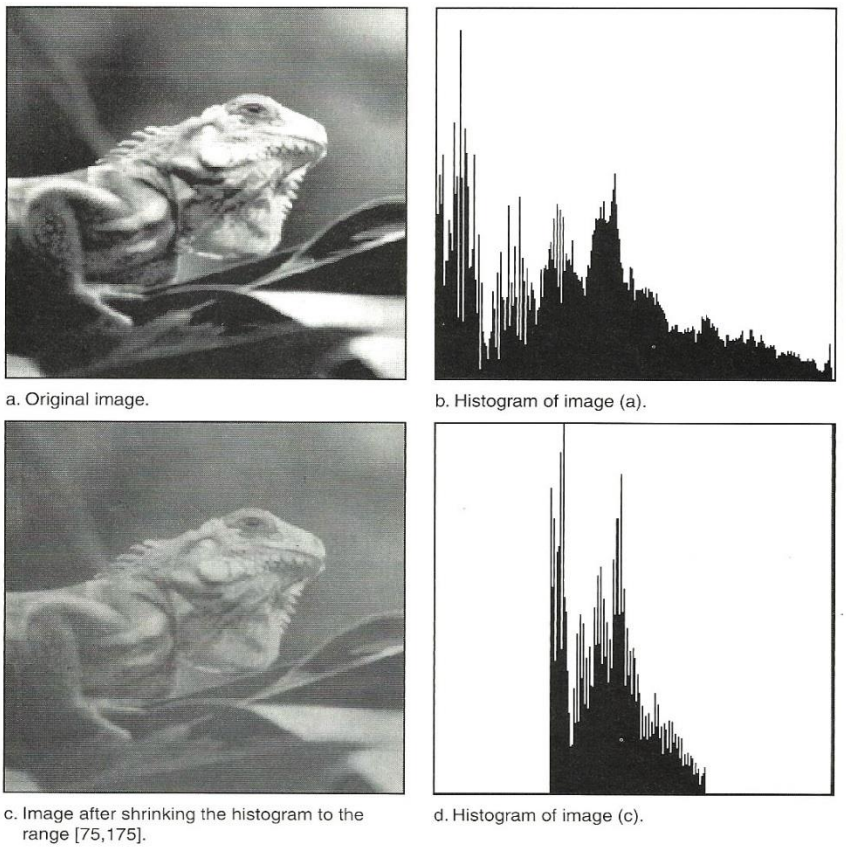
$$\text{Shrink}(I(r,c)) = \left[\frac{\text{Shrink}_{MAX} - \text{Shrink}_{MIN}}{I(r,c)_{MAX} - I(r,c)_{MIN}}\right]\left[I(r,c) - I(r,c)_{MIN}\right] + \text{Shrink}_{MIN}$$

Where : $\boldsymbol{I(r,c)_{MAX}}$ is the largest gray-level value in the image I(r,c)

$\boldsymbol{I(r,c)_{MIN}}$ is the smallest gray-level value in the image I(r,c)

$\mathbf{Shrink_{MAX}}$ and $\mathbf{Shrink_{MIN}}$ correspond to the maximum and minimum gray-level values ***desired*** in the compressed histogram.

In general, this process produces an image of ***reduced contrast*** an may not seem to be useful as an image enhancement tool.



a. Original image.

b. Histogram of image (a).

c. Image after shrinking the histogram to the range [75,175].

d. Histogram of image (c).

Figure 14: histogram shrink

c. Histogram slide

The histogram slide technique can be used to make an image either ***darker*** or ***lighter*** but retain the relationship between gray-levels values. This can be accomplished by simply **adding** or **subtracting** a fixed number from all the gray level values as follow:

$$Slide(I(r,c)) = I(r,c) + OFFSET$$

Where OFFSET value is the amount to slide the histogram**.**

In this equation, we assume that any values slide past the minimum and maximum value will be clipped to the respective minimum or maximum. Aposative OFFSET value will increase the overall brightness, whereas a negative OFFSET will create a darker image. Figure 15 shows a dark image that has been brightened by a histogram slide with a positive OFFSET value.

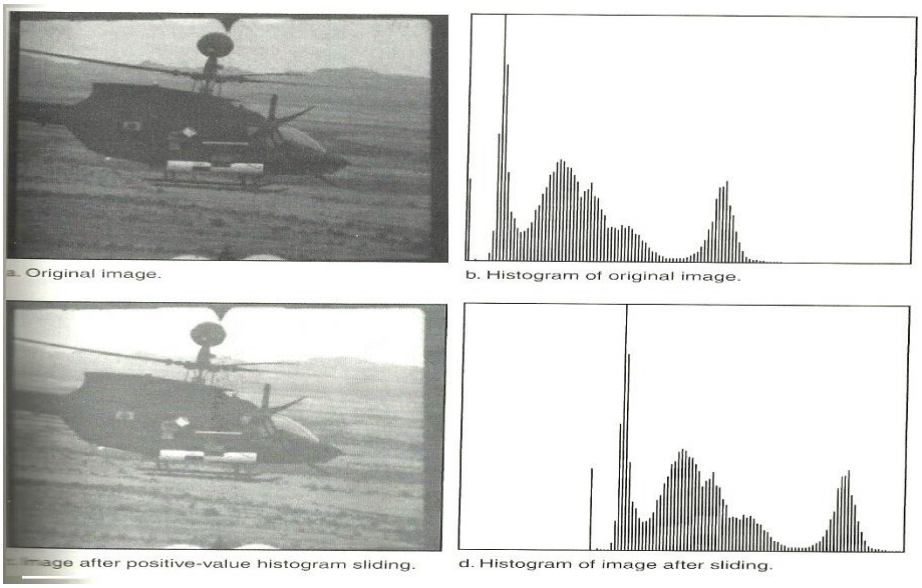


Figure 15: histogram slide

Example: Apply histogram stretching for the following sub image :

$$\begin{bmatrix} 7 & 12 & 8 \\ 20 & 9 & 6 \\ 10 & 15 & 1 \end{bmatrix}$$

Where: ***Max*** =255 ; ***Min*** =0

Solution:

$$St(r,c) = \left[\frac{I(r,c) - I(r,c)_{min}}{I(r,c)_{max} - I(r,c)_{min}}\right](Max - Min) + Min$$

$I(r,c)_{min}$ = 1 ; $I(r,c)_{max}$ = 20 ; ***Max*** =255 ; ***Min*** =0

$I_{(0,0)}$ = [7-1 / 20-1] * [255 – 0] + 0 = 80.5
$I_{(0,1)}$ = [12-1 / 20-1] * [255 – 0] + 0 = 147.6
$I_{(0,2)}$ = [8-1 / 20-1] * [255 – 0] + 0 = 93.9
$I_{(1,0)}$ = [20-1 / 20-1] * [255 – 0] + 0 = 255
$I_{(1,1)}$ = [9-1 / 20-1] * [255 – 0] = 107.3
$I_{(1,2)}$ = [6-1 / 20-1] * [255 – 0] + 0 = 67.1
$I_{(2,0)}$ = [10-1 / 20-1] * [255 – 0] + 0 = 120.7
$I_{(2,1)}$ = [15-1 / 20-1] * [255 – 0] + 0 = 187.8
$I_{(2,1)}$ = [1-1 / 20-1] * [255 – 0] + 0 = 0

$$\begin{bmatrix} 80 & 147 & 93 \\ 255 & 107 & 67 \\ 120 & 187 & 0 \end{bmatrix}$$

Example: Apply histogram shrink for the following sub image :

$$\begin{bmatrix} 70 & 120 & 80 \\ 200 & 90 & 60 \\ 100 & 150 & 10 \end{bmatrix}$$

Where: **Shrink max** =100 ; **shrink min** =20

Solution:

$$\text{Shrink}(I(r,c)) = \left[\frac{\text{Shrink}_{MAX} - \text{Shrink}_{MIN}}{I(r,c)_{MAX} - I(r,c)_{MIN}}\right]\left[I(r,c) - I(r,c)_{MIN}\right] + \text{Shrink}_{MIN}$$

$I(r,c)_{min}$ = 10 ; $I(r,c)_{max}$ = 200 ; **Shrink max** =100 ; **shrink min** =20

$I_{(0,0)}$ = [100-20 / 200-10] * [70 – 10] + 20= 45.2
$I_{(0,1)}$ = [100-20 / 200-10] * [120 – 10] + 20= 66.3
$I_{(0,2)}$ = [100-20 / 200-10] * [80 – 10] + 20= 49.4
$I_{(1,0)}$ = [100-20 / 200-10] * [200 – 10] + 20= 100
$I_{(1,1)}$ = [100-20 / 200-10] * [90 – 10] + 20=53.68

$I_{(1,2)}$ = [100-20 / 200-10] * [**60** – 10] + 20= 41.05

$I_{(2,0)}$ = [100-20 / 200-10] * [**100** – 10] + 20= 57.89

$I_{(2,1)}$ = [100-20 / 200-10] * [**150** – 10] + 20= 78.94

$I_{(2,2)}$ = [100-20 / 200-10] * [**10** – 10] + 20= 20

$$\begin{bmatrix} 45 & 66 & 49 \\ 100 & 53 & 41 \\ 57 & 78 & 20 \end{bmatrix}$$

Example: Apply histogram slide for the following sub image, where OFFSET= 10 :

$$\begin{bmatrix} 7 & 12 & 8 \\ 20 & 9 & 6 \\ 10 & 15 & 1 \end{bmatrix}$$

Solution:

Slide($I(r,c)$) $= I(r,c)$ + OFFSET

$$\begin{bmatrix} 17 & 22 & 18 \\ 30 & 19 & 16 \\ 20 & 25 & 11 \end{bmatrix}$$

Histogram equalization

Histogram equalization: is a technique for adjusting image intensities to enhance **contrast**.

Histogram equalization often produces unrealistic effects in photographs; however, it is very useful for scientific images like thermal, satellite or x-ray images.

To find the histogram equalization must follow:

1- Count the total number of pixels associated with each pixel intensity.

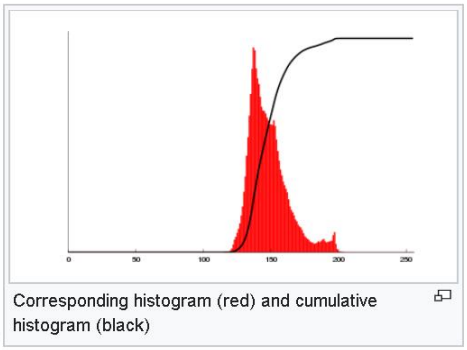
2- Cumulative distribution function (CDF)

3- Calculate as transformation function

$$.h(v) = cut\left(\frac{cdf(v)-cdf_{min}}{(M-N)-1} \times (L-1)\right)$$

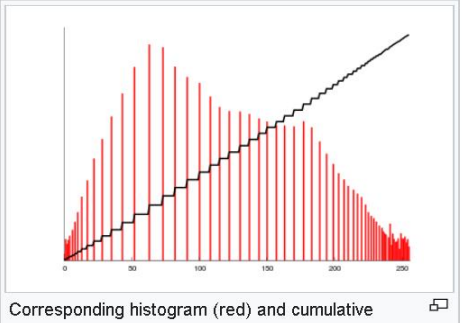
- cdf_{min} is the minimum non-zero value of the cumulative distribution function.
- $M*N$ gives the image's number of pixels.
- L is the number of grey levels used (in most cases 256)

Before Histogram Equalization



Corresponding histogram (red) and cumulative histogram (black)

After Histogram Equalization



Corresponding histogram (red) and cumulative

Example:

Apply histogram equalization for the following sub image, where image is gray scale :

$$\begin{bmatrix} 50 & 55 & 150 & 150 \\ 51 & 50 & 55 & 55 \\ 70 & 80 & 90 & 100 \\ 50 & 55 & 70 & 80 \end{bmatrix}$$

Solution:

$$h(v) = cut\left(\frac{cdf(v) - cdf_{min}}{(M-N)-1} \times (L-1)\right)$$

$M*N$ = 4 * 4 = 16 L = 256 (because its gray scale)

cdf_{min} = **3**

$$h(50) = cut\left(\frac{3-3}{16-1}\right) * (256-1) = 0$$

$$h(51) = cut\left(\frac{4-3}{16-1}\right) * (256-1) = 17$$

$$h(55) = cut\left(\frac{8-3}{16-1}\right) * (256-1) = 51$$

$$h(70) = cut\left(\frac{10-3}{16-1}\right) * (256-1) = 119$$

$$h(80) = cut\left(\frac{12-3}{15}\right) * (255) = 153$$

$$h(90) = cut\left(\frac{13-3}{15}\right) * (255) = 170$$

$$h(100) = cut\left(\frac{14-3}{15}\right) * (255) = 187$$

$$h(150) = cut\left(\frac{16-3}{15}\right) * (255) = 221$$

Pixel Intensity	Count	Cdf_r	H(r)
50	3	3	**0**
51	1	4	**17**
55	4	8	**51**
70	2	10	**119**
80	2	12	**153**
90	1	13	**170**
100	1	14	**187**
150	2	16	**221**

$$\begin{bmatrix} 0 & 51 & 221 & 221 \\ 17 & 0 & 51 & 51 \\ 119 & 153 & 170 & 187 \\ 0 & 51 & 119 & 143 \end{bmatrix}$$

المحور الرابع

18 & 19. Image compression techniques

Image compression is a type of data compression applied to digital images, to reduce their cost for **storage or transmission**. Data compression refers to the process of reducing the amount of data required to represent a given quantity of information.

- The reduced file is called the compressed file and is used to reconstruct the image
- Resulting in the decompressed image is the original image, is called the uncompressed image file.

$$Compression\ Ratio = \frac{Uncompressed\ File\ Size}{Compressed\ File\ Size} = \frac{size_U}{size_C}$$

Example: the original image is 256×256 pixel, single band (gray scale), 8-bit per pixel. This file is 65,536 bytes (64K). After compression the image file is 6,554 byte. The compression ratio is:

Size U/Size C = 65536/6554= 9.999=10 this can be written as **10:1**

This is called a "10 to 1" compression or a "10 times compression", or it can be stated as "compressing the image to 1/10 original size.

Image compression types

There are two primary types of image compression methods and they are:

1. Lossless Compression

- This compression is called lossless because no data are lost (with no loss in image quality)
- And the original image can be recreated exactly from the compressed data.
- Lossless compression is generally used for text or spreadsheet files, where losing words or financial data could pose a problem
- The Graphics Interchange File (GIF) is an image format used lossless compression

2. Lossy Compression.

- Lossy Compression is the class of data encoding methods that allows a loss in some of data images.
- The uncompressed image cannot be same the original image file.
- Lossy methods can provide high degrees of compression and result in smaller compressed files, but some number of the original pixels, sound waves or video frames are removed forever.

Compression System Model

The compression system model consists of two parts: the compressor (Encoding) and the decompressor (Decoding).

Compressor: consists of preprocessing stage and encoding stage.

Decompressor: consists of decoding stage followed by a post processing stage, as following figure:

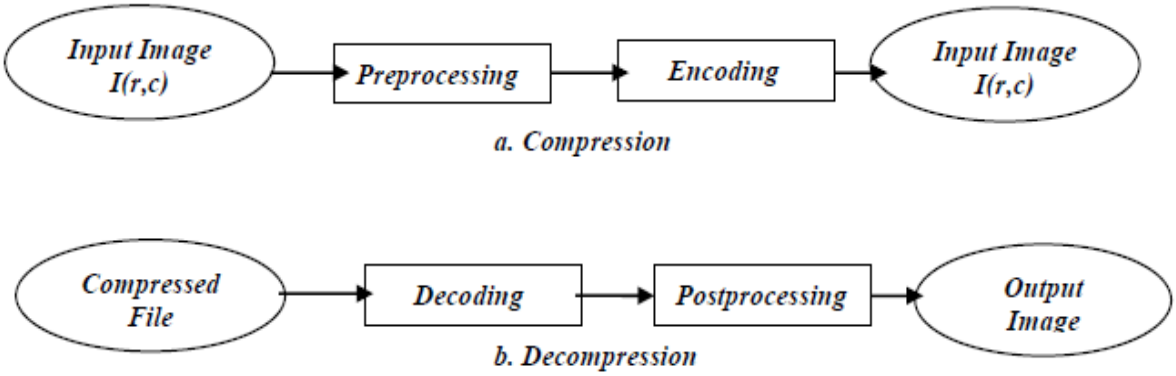


Figure (1): Compression System Model.

Before encoding, *preprocessing* is performed to prepare the image for the encoding process. After the compressed file has been decoded, *post processing* can be performed to eliminate some of the undesirable artifacts brought about by the compression process. Often, many practical compression algorithms are a combination of a number of different individual compression techniques.

Entropy

An important concept here is the idea of measuring the average information in an image, referred to as entropy. The entropy of N×N image can be calculated by this equation.

$$Entropy = -\sum_{i=0}^{L-1} P_i \log_2(P_i) \qquad \text{(in bits per pixel)}$$

Where

P_i = The probability of the i_{th} gray level n_k/N^2

n_k= the total number of pixels with gray value k.

L= the total number of gray levels (e.g. 256 for 8-bits)

Example

Let L=8, meaning that there are 3 bits/ pixel in the original image. Let that number of pixel at each gray level value are equal (they have the same probability) that is:

$$P_0=P_1=P_2=\ldots\ldots=P_7=1/8$$

Now, we can calculate the entropy as follows:

$$Entropy = -\sum_{i=0}^{7} P_i \log(P_i) \quad = -\sum_{i=0}^{7} \frac{1}{8} \log_2(\frac{1}{8}) = 3$$

This tell us that the theoretical minimum for lossless coding for this image is 8 bit/pixel

Note: *$Log_2(x)$ can be found by taking Log_{10} (x) and multiplying by 3.33 bacuase $1/Log_{10}$ (2) =3.32192809488736234*

Lossless Compression Algorithms

1. Run-length Encoding (RLE)

- This encoding method is frequently applied to graphics-type images (or pixels in a scan line) - simple compression algorithm in its own right.
- It is also a component used in JPEG compression pipeline.
- Example: Original sequence: 11112223333311112222

  can be encoded as: (4,1),(3,2),(6,3),(4,1),(4,2)

How Much Compression? The savings are dependent on the data: In the worst case (random noise) encoding is heavier than original file.

2. **Simple Repetition Suppression**

- Replace series with a token and a count number of occurrences.
- Usually need to have a special flag to denote when the repeated token appears.
- Simplicity is its downfall: poor compression ratios.
- Compression savings depend on the content of the data.

Example:

894000000000000000000000000000000

We can replace with: **894f32** /where f is the flag for zero.

3. Pattern Substitution

- This is a simple form of statistical encoding.
- Here we substitute a frequently repeating pattern(s) with a code.
- *The code is shorter than the pattern giving us compression.*
- The simplest scheme could employ predefined codes:

Example: Basic Pattern Substitution

Replace all occurrences of pattern of characters `and' with the predefined code '&',

So: ***and you and I*** becomes: **& you & I**

4-Shanon-Fano Method

A Shannon–Fano tree is built according to a specification designed to define an effective code table. The actual algorithm is simple:

1. For a given list of symbols, develop a corresponding list of probabilities or frequency counts so that each symbol's relative frequency of occurrence is known.
2. Sort the lists of symbols according to frequency, with the most frequently occurring symbols at the left and the least common at the right.
3. Divide the list into two parts, with the total frequency counts of the left part being as close to the total of the right as possible.

4. The left part of the list is assigned the binary digit 0, and the right part is assigned the digit 1. This means that the codes for the symbols in the first part will all start with 0, and the codes in the second part will all start with 1.
5. Recursively apply the steps 3 and 4 to each of the two halves, subdividing groups and adding bits to the codes until each symbol has become a corresponding code leaf on the tree.

Example: The source of information A generates the symbols {A0, A1, A2, A3 and A4} with the corresponding probabilities {0.4, 0.3, 0.15, 0.1 and 0.05}. Encoding the source symbols using binary encoder and Shannon-Fano encoder gives:

Source Symbol	P_i	Binary Code	Shannon-Fano
A0	0.4	000	0
A1	0.3	001	10
A2	0.15	010	110
A3	0.1	011	1110
A4	0.05	100	1111
L_{avg}	H = 2.0087	3	2.05

The Entropy of the source is:

$$H = -\sum_{i=0}^{4} Pi \log_2 Pi = 2.0087 \text{ bit/symbol}$$

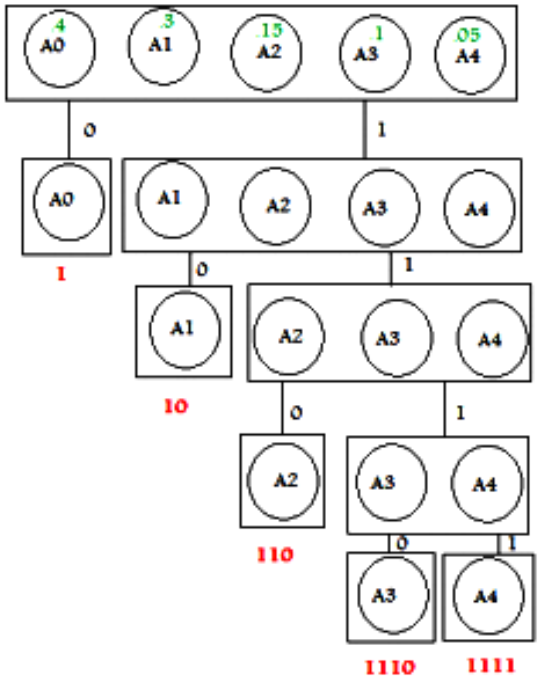
Since we have 5 symbols ($5<8=2^3$), we need 3 bits at least to represent each symbol in binary (fixed-length code). Hence the average length of the binary code is

$$Lavg = \sum_{i=0}^{4} Pi\, li = 3\,(0.4 + 0.3 + 0.15 + 0.1 + 0.05) = 3 \text{ bit/symbol}$$

Thus the efficiency of the binary code is

$$\eta = \frac{H}{Lavg} = \frac{2.0087}{3} = 67\%$$

Shannon-Fano code is a top-down approach. Constructing the code tree, we get



The average length of the Shannon-Fano code is

$$\text{Lavg} = \sum_{i=0}^{4} \text{Pi li} = 0.4 * 1 + 0.3 * 2 + 0.15 * 3 + 0.1 * 4 + 0.05 * 4 = 2.05 \text{ bit/symbol}$$

Thus the efficiency of the Shannon-Fano code is

$$\eta = \frac{H}{\text{Lavg}} = \frac{2.0087}{2.05} = 98\%$$

This example demonstrates that the efficiency of the Shannon-Fano encoder is much higher than that of the binary encoder.

Huffman Code

The Huffman coding algorithm comprises two steps, reduction and splitting. These steps can be summarized as follows:

1) Reduction

a) List the symbols in descending order of probability.

b) Reduce the *r* least probable symbols to one symbol with a probability equal to their combined probability.

c) Reorder in descending order of probability at each stage.

d) Repeat the reduction step until only two symbols remain.

2) Splitting

a) Assign *r* ,... 1, 0 to the *r* final symbols and work backwards.

b) Expand or lengthen the code to cope with each successive split.

Example: Design Huffman codes for A = {$a1$, $a2$, … … . $a5$}, having the probabilities {0.2, 0.4, 0.2, 0.1, 0.1}.

Symbol	Step 1	Step 2	Step 3	Step 4	Codeword
a_2	0.4 →	0.4 →	0.4 →	0.6 0	1
a_1	0.2 →	0.2 →	0.4 0	0.4 1	01
a_3	0.2 →	0.2 0	0.2 1		000
a_4	0.1 0	0.2 1			0010
a_5	0.1 1				0011

Letter	Probability	Codeword
a_2	0.4	1
a_1	0.2	01
a_3	0.2	000
a_4	0.1	0010
a_5	0.1	0011

The average code word length:

$$Lavg = \sum_{i=0}^{4} Pi\ li$$

$$L_{avg} = \sum_{i=0}^{4} P_i l_i$$

L = 0.4 × 1 + 0.2 × 2 + 0.2 × 3 + 0.1 × 4 + 0.1 × 4 = **2.2 *bits/symbol***

The source entropy:

$$\boldsymbol{H(S) = \sum p_i \log_2 \frac{1}{p_i}} \quad \text{or} \quad \boldsymbol{H(S) = -\sum p_i \log_2 p_i}$$

$(Y) = -[0.4ln0.4 + 2 \times 0.2ln0.2 + 2 \times 0.1ln0.1]/ln2 =$ **2.12193 bits/symbol**

The code efficiency:

$$\eta = \frac{\mathrm{H}}{\mathrm{Lavg}}$$

η =(2.12193 / 2.2)× 100 = 96.45%

المحور الخامس

Basics of Digital Audio (20th 21st)

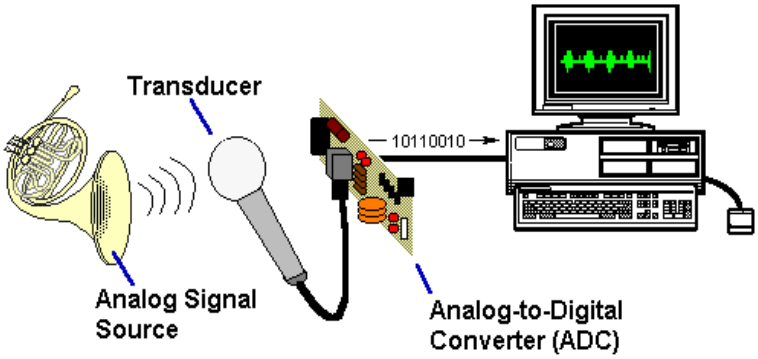
Sound is a wave phenomenon like light, but it is macroscopic and involves molecules of air being compressed and expanded under the action of some physical device.

Sound is a vibration that typically propagates as an audible wave of pressure, through a transmission medium such as a gas, liquid or solid.

The simplest kind of sound wave is a **sine wave**. Pure sine waves rarely exist in the natural world, but they are a useful place to start because all other sounds can be broken down into combinations of sine waves. Even though such pressure waves are longitudinal, they still have ordinary wave properties and behaviors, such as reflection (bouncing), refraction (change of angle when entering a medium with a different density) and diffraction (bending around an obstacle).

1- Detecting Sound: Since sound consists of measurable pressures at any 3D point, we can detect it by measuring the pressure level at a location, using a transducer to convert pressure to voltage levels.

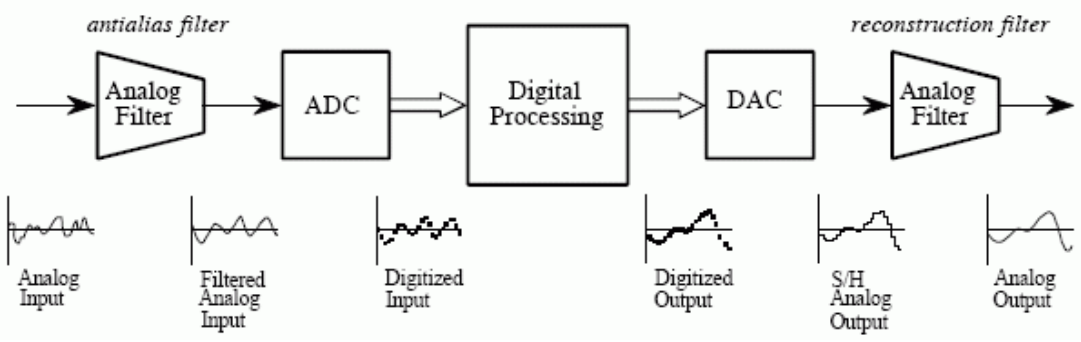
Microphone: Receives sound and converts it to an analog signal.



If we wish to use a digital version of sound waves we must form digitized representations of audio information.

2- The computer needs discrete entities: Analog-to-Digital converter is used Dedicated Hardware (e.g. Sound card). Also known as Digital Sampling.

- Anti-aliasing filters (major part of Analog Conditioning) are needed at the input to remove frequencies above the sampling limit that would result in aliasing. The anti-aliasing filter at the output removes the aliases that result from the sampling.
- After the anti-aliasing filter, the analog/digital converter (ADC) quantises the continuous input into discrete levels.
- After digital processing, the output of the system is given to a digital/analog converter (DAC) which converts the discrete levels into continuous voltages or currents.
- This output must also be filtered with a low pass filter to remove the aliases from the sampling. Subsequent processing can include further filtering, mixing, or other operations.

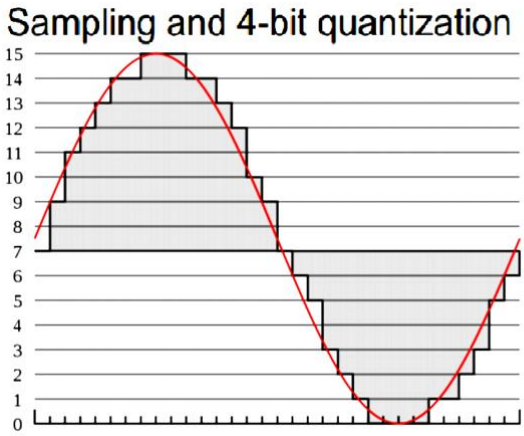


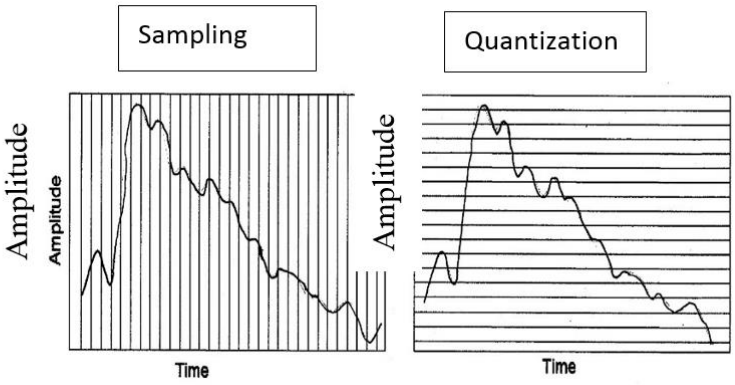
Digitization means conversion to a stream of numbers, and preferably these numbers should be integers for efficiency.

Sampling is digitizing the analog signal in the time dimension;

Quantization is digitizing the analog signal in the amplitude dimension.

16 bits = 65,536 levels (Quantization)

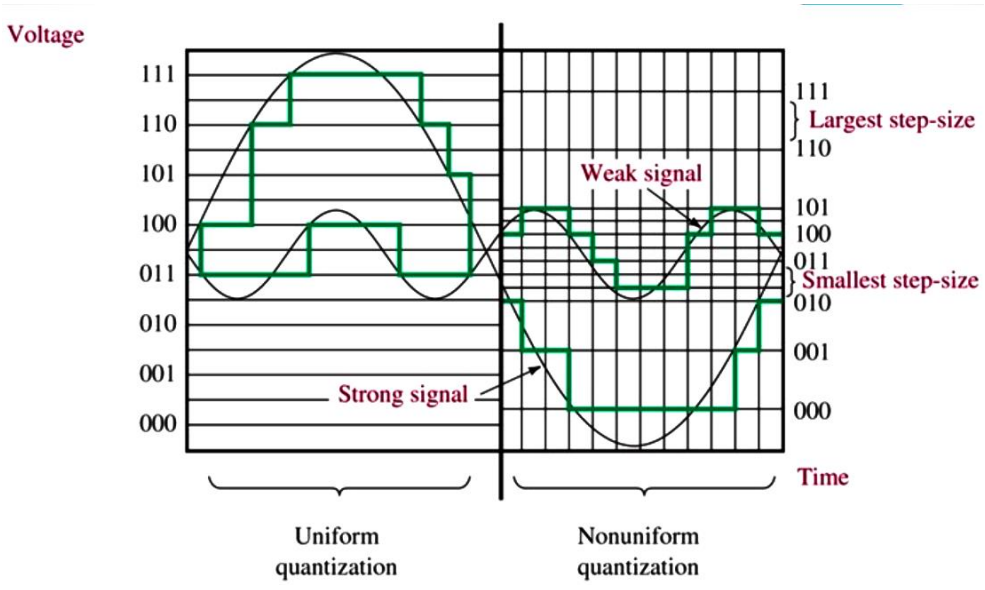




Sampling	Quantization
Sampling rate: Number of samples per second (measured in Hz)	3-bit quantization gives 8 possible sample values
E.g., CD standard audio uses a sampling rate of 44,100 Hz (44100 samples per second)	E.g., CD standard audio uses 16-bit quantization giving 65536 values.

We have discussed only **uniform sampling**, with equally spaced sampling intervals. Typical uniform quantization rates are 8-bit and 16-bit; 8-bit quantization divides the vertical axis into 256 levels, and 16-bit divides it into 65,536 levels.

Non-uniform sampling is also possible. This is not used for sampling in time, but is used for quantization (the μ-law). We call it non-linear If its logarithmic. The non-linear scale is used because small amplitude signals are more likely to occur than large amplitude signals, and they are less likely to mask any noise.



Data rate = sample rate * quantization * channel

Q) Compare rates for CD vs. mono audio?

Mono audio: ***Data rate =*** 8000 samples/second * 8 bits/sample * 1 channel

= 7.8 kBytes / second

CD: ***Data rate =*** 44,100 samples/second * 16 bits/sample * 2 channels

= 172.26 KBytes / second ~= 10MB / minute

File size = (Data rate * time) + Header file

= sample rate * quantization * channel* time in second

Q) what is file size for CD data rate with 10 minutes

File size= 172.26 Kbytes * 60_{second}*$10_{minunts}$= 100.9Mbyts

Audio Quality vs. Data Rate

Quality	*Sample Rate (kHz)*	*Bits per Sample*	*Mono/ Stereo*	*Data Rate (kBytes/sec) (uncompressed)*	*Frequency Band*
Telephone	8	8	Mono	8	200-3400 Hz
AM Radio	11.025	8	Mono	11.0	540-1700KHz
FM Radio	22.050	16	Stereo	88.2	
CD	44.1	16	Stereo	176.4	20-20000 Hz
DAT	48	16	Stereo	192.0	20-20000 Hz

Nyquist Theorem (22nd)

As a simple illustration, Fig. (1,a) shows a single sinusoid: it is a single, pure, frequency (only electronic instruments can create such boring sounds). Now if the sampling rate just equals the actual frequency, we can see from Fig. (1,b) that a false signal is detected: it is simply a constant, with zero frequency. On the other hand, we sample at 1.5 times the frequency, Fig. (1,c) shows that we obtain an **incorrect (*alias*) frequency** that is lower than the correct one. it is half the correct one (the wavelength, from peak to peak, is double that of the actual signal).

An alias is any artifact that does not belong to the original signal. Thus, for correct sampling we must use a sampling rate equal to at least **twice the maximum frequency** content in the signal. This is called the **Nyquist rate**.

$$f_s \geq 2f_c$$

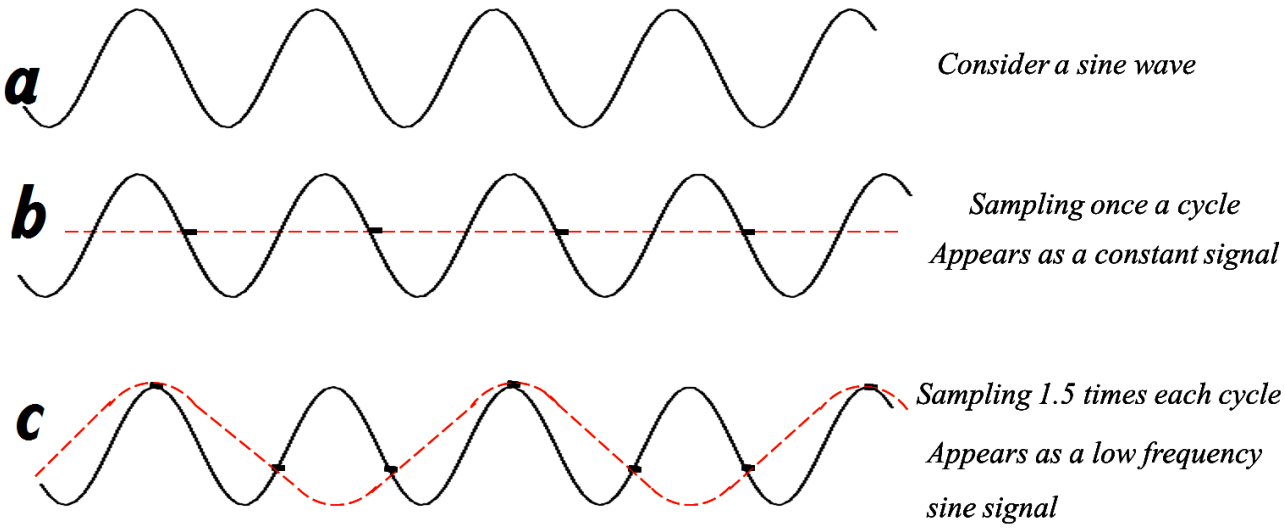


Fig. 1: Aliasing:

(a) a single frequency;

(b) sampling at exactly the frequency produces a constant;

(c) sampling at 1.5 times per cycle produces an alias frequency that is perceived

What will happen if we get Nyquist Sampling Wrong?

Digital Sampling Artifacts Arise - Effect known as Aliasing which affects Audio,

Generally, if a signal is band-limited—that is, if it has a **lower limit *f1*** and an **upper limit *f2*** of frequency components in the signal—then we need a sampling rate of at least **2(*f2* − *f1*).**

Nyquist theorem is used to calculate the optimum sampling rate in order to obtain good audio quality.

For audio, typical sampling rates are from 8 kHz (8,000 samples per second) to 48 kHz. This range is determined by the Nyquist theorem. Humans have a range of hearing from 20 Hz (low) to 20,000 Hz (high)

Q: Why are CD Sample Rates 44.1 KHz?

The upper range of human hearing is around 20Hz - 22 KHz. Apply Nyquist Theorem ***(2× frequencies)***. Therefore, sampling at twice the maximum frequency (44 KHz) we could have achieved good audio quality.

Suppose we have a fixed sampling rate. Since it would be impossible to recover frequencies higher than half the sampling rate in any event, most systems have an anti-aliasing filter that restricts the frequency content of the sampler's input to a range at or below half the sampling frequency. The frequency equal to half the Nyquist rate is called the Nyquist frequency.

$$f_{alias} = f_{sampling} - f_{true}, \quad for \quad f_{true} < f_{sampling} < 2 \times f_{true}$$

Example,

If the true frequency is 5.5 kHz and the sampling frequency is 8 kHz, then the alias frequency is 2.5 kHz: So, if again the sampling frequency is less than twice the true frequency and is less than the true frequency, then the alias frequency equals ***n*** times the sampling frequency minus the true frequency, where ***n*** is the lowest

integer that makes ***n*** times the sampling frequency larger than the true frequency.

For example, when the true frequency is between 1.0 and 1.5 times the sampling frequency, the alias frequency equals the true frequency minus the sampling frequency.

In general, the apparent frequency of a sinusoid is the lowest frequency of a sinusoid that has exactly the same samples as the input sinusoid.

Data Level

A digital signal has eight levels. How many bits are needed per level?

$$Number\ of\ bits\ per\ level\ = log_2 8 = 3$$

Noiseless - Nyquist Theorem

Nyquist gives the upper bound for the bit rate of a transmission system by calculating the bit rate directly from the number of bits in a symbol (or signal levels) and the bandwidth of the system.

Nyquist theorem states that for a noiseless channel:

$$C = 2\ B\ log_2 2^n$$

C= capacity in bps, **B = bandwidth in Hz,** **n= number of bits**

Example: ***Consider a noiseless channel with a bandwidth of 3000 Hz transmitting a signal with two signal levels.***

Number of bits per level= $\log_2 2 = 1$

C = 2 X (3000) $\log_2 2^1$= 6000 bps.

Example: Consider the same noiseless channel transmitting a signal with four signal levels

(for each level, we send 2 bits).

Number of bits per level= $\log_2 4 = 2$

C = 2 X (3000) $\log_2 2^2$= 1200 bps.

Example: ***Consider the telephone channel having bandwidth B = 4 kHz. Assuming there is no noise, determine channel capacity for the following encoding levels: (i) 2, and (ii) 128.***

(i) C = 2B = 2×4000 = 8 Kbits/s

(ii) C = 2×4000×$\log_2$128 = 8000×7 = 56 Kbits/s

Example: **Television channels are 6 MHz wide. How many bits/sec can be sent if four-level digital signals are used? Assume a noiseless channel.**

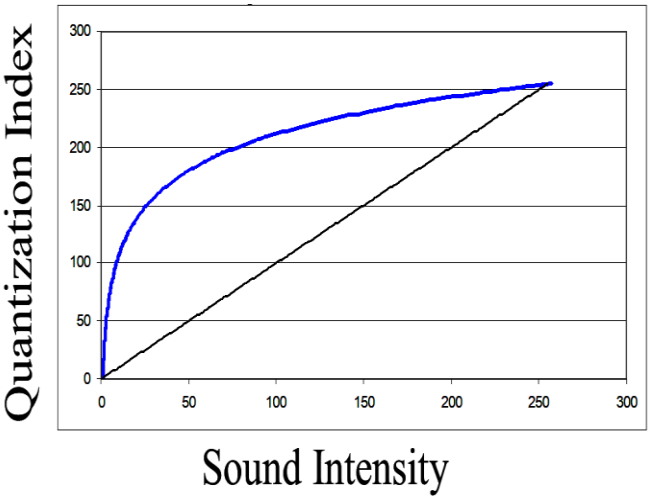
C = 2 X (6×10^6) $\log_2 4$= 24 Mbps.

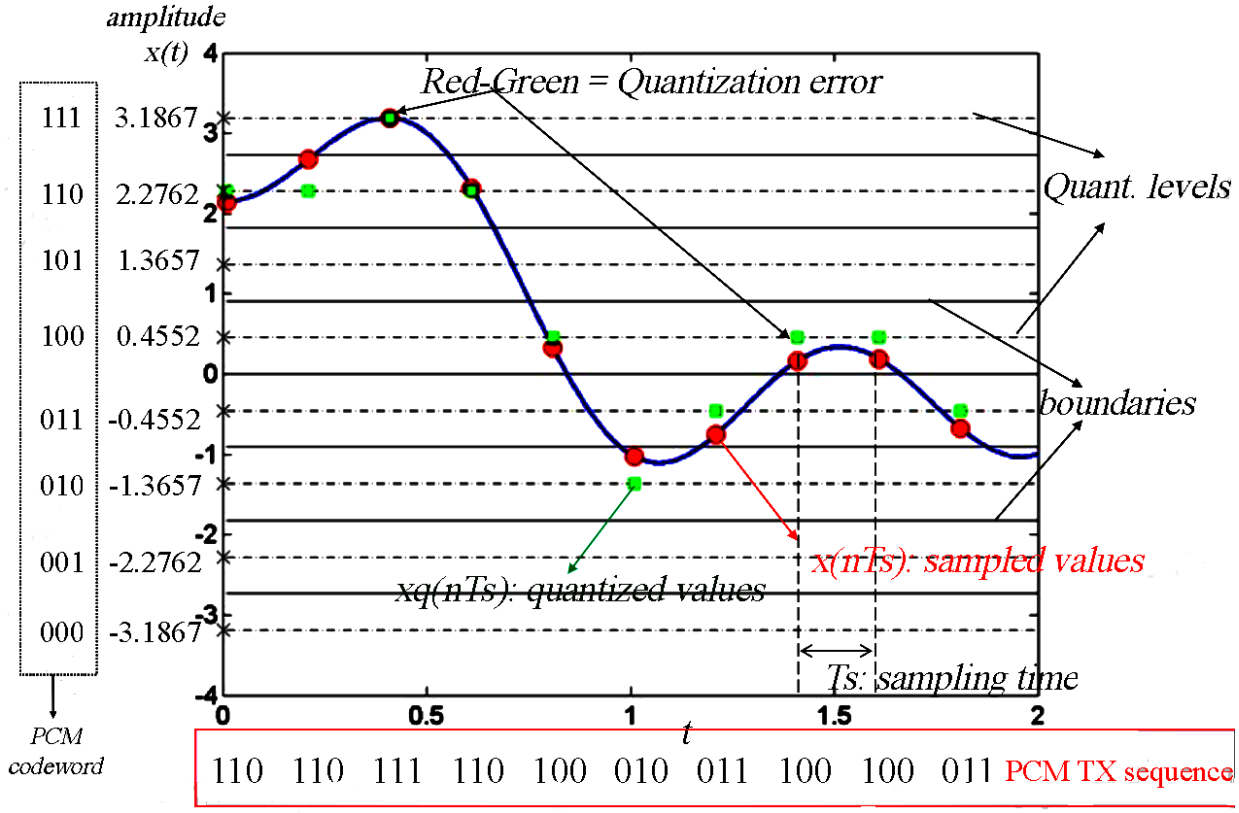
Synthetic Sound (23rd)

Quantization and transmission of audio (24th)

Quantization

- Since we quantize, we may choose to create either an accurate or less accurate representation of sound magnitude values.
- To compress the data, by assigning a bit stream that uses fewer bits for the most prevalent signal values.
- Quantization process introduces a certain amount of error or distortion into the signal samples.
- Perceptual Quantization (u-Law)
- Want intensity values logarithmically mapped over N quantization units





Quantization and transformation of data are collectively known as ***coding of the data***. For audio, the μ-law technique for companding audio signals is usually combined with a simple algorithm that exploits the temporal redundancy present in audio signals.

Example: CD audio, which uses 16-bit samples at a 44,100 Hz sampling rate. There are two parallel streams, one for each channel, to produce stereo. What is the transmission rate of CD-quality audio?

As long as you understand the terms involved, this is a straightforward math problem. For each of the two channels, there are 44,100 samples per second. Each of these samples requires 16 bits. Transmission rates are normally described in terms of the number of bits per second that must flow from the source to the destination. In our case:

44100 samples per second * 16 bits per sample * 2 channels = 1411.2 kbps

5-different Voice Digitization - techniques:

1) **PAM** = pulse amplitude modulation
2) **PDM** = pulse duration modulation
3) **PPM** = Pulse position modulation
4) **PCM** = Pulse code modulation
5) **ADPCM** = Adaptive differential PCM

In general, ***producing quantized sampled output for audio is called Pulse Code Modulation, or PCM.***

- PCM is an extension of PAM wherein each analog sample is quantized into a discrete value for representation as a digital code word.
- PAM system can be converted to PCM if we add ADC at the source and DAC at the destination.

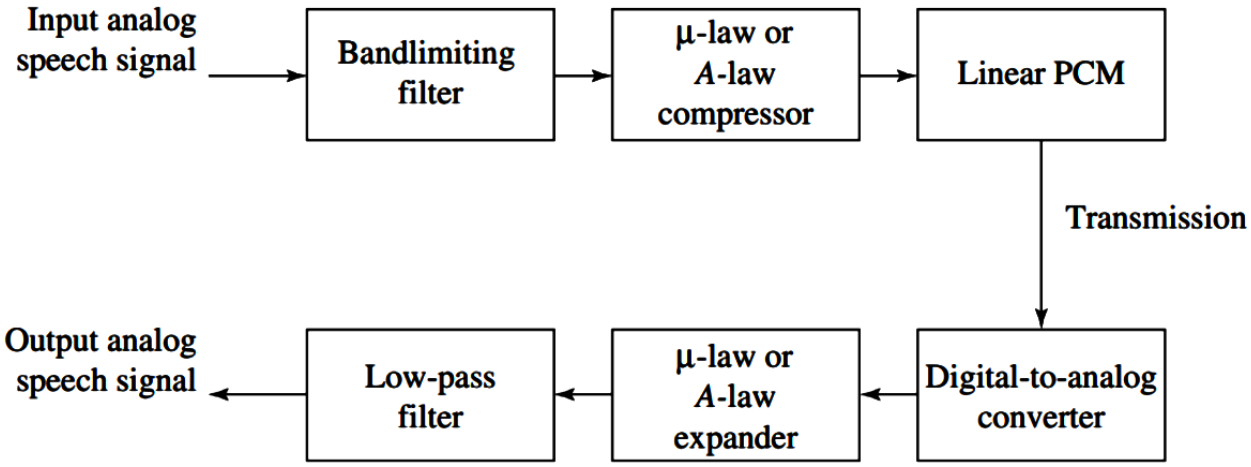


Figure: PCM signal encoding and decoding

Differential Pulse Code Modulation (DPCM)

What if we look at sample differences, not the samples themselves?

$$d_t = x_t - x_{t-1}$$

Differences tend to be smaller. Use 4 bits instead of 12 bits. Changes between adjacent samples small.

Value uses full bits, changes use fewer bits

Example:

(a) Full bits:-> 220, 218, 221, 219, 220, 221, 222, 218,… Result: originally for encoding sequence 0-255 numbers need 8 bits;

(b) Changes:-> (Difference sequence) 220, -2, +3, -2, +1, +1, +1, -4....

Difference coding: need only 3 bits

Compression of Audio (25th)

The Nyquist theorem states how frequently we must sample in time to be able to recover the original sound.

Data compression implies sending or storing a smaller number of bits. Although many methods are used for this purpose, in general these methods can be

divided into two broad categories: lossless and Lossy methods. Audio compression can be used for speech or music. For speech, we need to compress a 64 kHz digitized signal, while for music we need to compress a 1.411 MHz signal. Two categories of techniques are used for audio compression: predictive encoding and perceptual encoding.

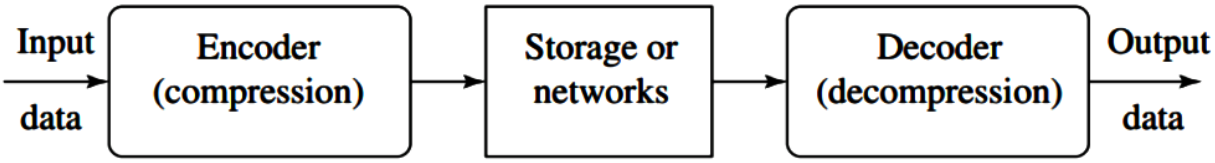


Figure: A general data compression scheme

lossy audio compression

Our eyes and ears cannot distinguish subtle changes. In such cases, we can use a lossy data compression method. These methods are cheaper, they take less time and space when it comes to sending millions of bits per second for images and video. Several methods have been developed using lossy compression techniques. JPEG (Joint Photographic Experts Group) encoding is used to compress pictures and graphics, MPEG (Moving Picture Experts Group) encoding is used to compress video, and MP3 (MPEG audio layer 3) for audio compression.

lossless audio compression

In lossless data compression, the integrity of the data is preserved. The original data and the data after compression and decompression are exactly the same because, in these methods, the compression and decompression algorithms are exact inverses of each other: no part of the data is lost in the process. Redundant data is removed in compression and added during decompression. Lossless compression methods are normally used when we cannot afford to lose any data.

1. µ-Law and A-Law Companding

These two method encode audio samples, by means of nonlinear quantization. The µ-law encoder inputs 14-bit samples and outputs 8-bit codewords. The A- law inputs 13-bit samples and outputs 8-bit codewords. G.711 standard is an 8-bit codewords whose format is shown :-

P	S2	S1	S0	Q3	Q2	Q1	Q0

- Bit P is sign bit of the output.
- Bits S2, S1, and S0 are the segment code.
- Bits Q3 through Q0 are the quantization code.

Algorithm (1): µ-Law encoder

Input : 14 bit samples

Output: 8 bit codewords

Begin:

step1: adding a bias 33to the absolute value of the input sample.

step2: determining the bit position of the most significant 1-bit among bits 5 through 12 of the input.

step3: subtracting 5 from that position.

Step4: The 4-bit quantization code is set to the four bits following the bit position determined in step 2.

Step5: The encoder ignores the remaining bits of the input sample, and it inverts codeword before it is output.

Algorithm (2): µ-Law decoder

Input : 8 bit codewords

Output: 14 bit samples

Begin:

Step1: Multiply the quantization code by 2 and add the bias 33 to result.

Step2: Multiply the result by 2 raised to the power of the segment code.

Step3: Decrement the result by the bias 33.

Step4: Use bit P to determine the sign of the result.

End

Example: input sample -656

P = 1 because sample is negative.

|-656|+33 = 689

$689 = 0001010110001_2$

				Q3	Q2	Q1	Q0					
0	0	0	1	0	1	0	1	1	0	0	0	1
12	11	10	9	8	7	6	5	4	3	2	1	0

The most significant 1-bit in positions 5 through 12 is found at position 9. The segment code value = 9-5 = 4.

The quantization code is the four bits 0101 at positions 8-5, and the remaining five bits 10001 are ignored. The 8-bit codeword (which is later inverted) becomes

P	S2	S1	S0	Q3	Q2	Q1	Q0
1	1	0	0	0	1	0	1

μ - Law decoder

1. The quantization code is $101_2 = 5$, so 5 x 2 + 33 = 43
2. The segment code is $100_2 = 4$, so 43 x 2^4 = 688.
3. Decrement by the bias 688 — 33 = 655.
4. Bit P is 1, so the final result is —655. Thus, the quantization error (the noise) is 1; very small.

المحور السادس

Video Basics (26th)

INTRODUCTION:

Video can be defined as the photographic images that are played back at speeds of 15, 25, 30 frames per second and provide the appearance of full motion (see **Figure 1**). **A video consists of a time-ordered sequence of frames (images).**

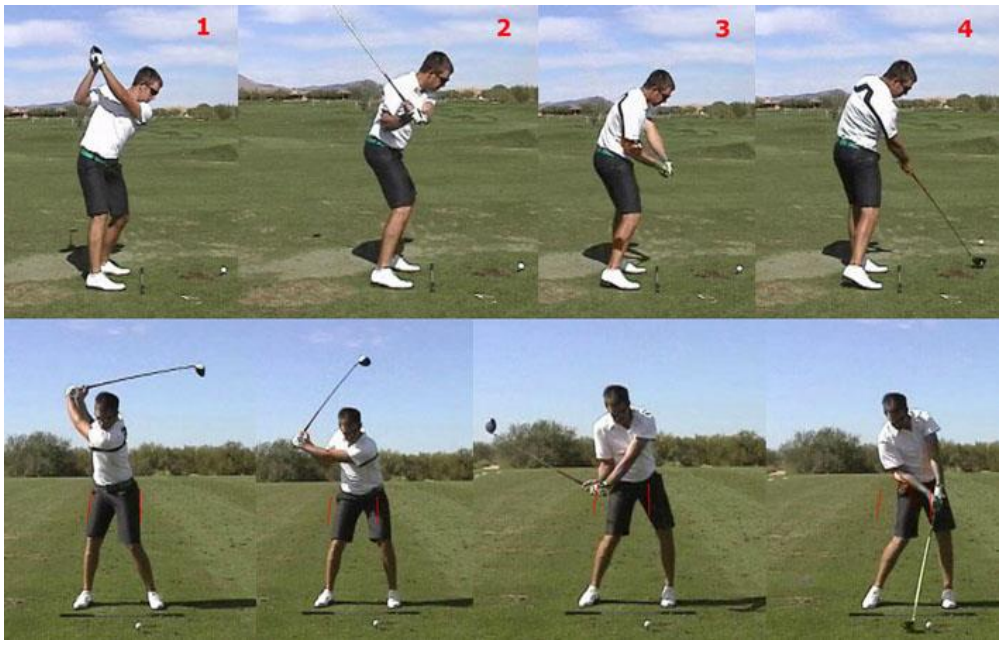


Figure 1: Video Frame

DIGITAL VIDEO:

Digital video refers to the capturing, manipulation, and storing of moving images that can be displayed on computer screens. This requires that the moving images be digitally handled by the computer. The word digital refers to a system based on discontinuous events (sampling), as opposed to analogue, a continuous event. Visual **pixels** are the basic unit in digital video, where each color component sorted digitally in each pixel.

5 Things To Consider When Playing Video:

- **Frame rate**: is the frequency (normally per second) at which frames in a television picture, film, or video sequence are displayed.
- **Video resolution**: is the number of distinct pixels in each dimension that can be displayed.
- **Video colouring model (colour space):** A colour model is an abstract mathematical model describing the way colours can be represented as tuples of numbers.
- **Raw video:** It is the unprocessed data from a camera's image sensor. Most

photographers prefer shooting raw film due to the high quality of images that the camera sensor could possibly produce.

- **Aspect Ratio:** The aspect ratio of an image describes the proportional relationship between its **width** and its **height**. It is commonly expressed as two numbers separated by a colon, as in 16:9.

Frame Dimensions, Number of Lines, and Resolution

A video frame is composed of lines. In digital video, each line is sampled to create a number of pixels (samples) per line. The more lines per frame, the higher the image resolution. The more pixels per line, the higher the resolution of each line.

Number of Lines

In analog video, NTSC uses 525 lines, whereas PAL uses 625, many lines are not actually used for picture information, so the total numbers relevant for the picture are somewhat smaller: 486 lines for NTSC and 576 lines for PAL. HD formats defined by the ATSC have either 1080 or 720 active picture lines per frame.

Pixels per Line

In digital video formats, each line is sampled a number of times. In an attempt to create a single digital VTR that could digitize and record both NTSC and PAL signals. Therefore, a digital NTSC video frame is 720 pixels x 486 lines, and a PAL video frame is 720 pixels x 576 lines.

HD video with 1080 lines uses 1920 pixels per line (1920 x 1080). HD video with 720 lines uses 1280 pixels per line (1280 x 720). Both of these formats have an aspect ratio of 16:9.

Common video frame sizes are shown in the table.

Width	Height	Pixel aspect ratio	Screen aspect ratio	Description
320	240	1:1	4:3	Used for web distribution or offline video editing.
640	480	1:1	4:3	An early standard for analog-to-digital video editing, and an ATSC video specification.
720[1]	480	Height greater than width	4:3	NTSC DV and DVD image dimensions. Also part of the ATSC video specification.
720[1]	486	Height greater than width	4:3	NTSC SD video dimensions used for professional digital formats such as Digital Betacam, D-1, and D-5.
720[1]	576	Width	4:3	PAL SD video dimensions used for digital

		greater than height		formats such as Digital Betacam, D-1, and D5, as well as DVD and DV.
1280	720	1:1	16:9	An HD video format, capable of higher frame rates in exchange for smaller image dimensions.
1920	1080	1:1	16:9	An HD video format with very high resolution.
960	720	4:3	16:9	Some 720p formats (such as DVCPRO HD and HDV) subsample 1280 pixels to 960 to minimize the data rate.
1440 1280	1080	4:3 3:2	16:9	Some 1080-line formats (such as HDV and DVCPRO HD) subsample 1920 pixels to 1440 or even 1280 to minimize the data rate.

In order to know calculate the size of a given video clip (without compression), we use the following equation:

video size (bits) = width x height x bit per pixel x frame rate x duration (time)

Where:

Bit per pixel = 24 for RGB, 8 for gray-level and 1 for binary image

Frame rate or frame per second = 15, 25, 30 fps or even more

Duration = time in seconds

Example:

Find the size in Kbyte for a 10 minutes Full HD (1920x1080) video sequence (RGB type) with a frame rate = 30 fps.

Solutions:

RGB type = 24 bits=3Bytes

Time in seconds = 10 x 60 = 600

Video size = 1980 x 1080 x 3 x 30 x 600

Video size = = 115473600000 Bytes

To convert to KBytes

Video size = 115473600000/1024

= 112767187.5 Kbytes

Video Rate (bits) = width x height x bit per pixel x frame rate

Example:
What is the bit rate for high-definition TV (HDTV)?

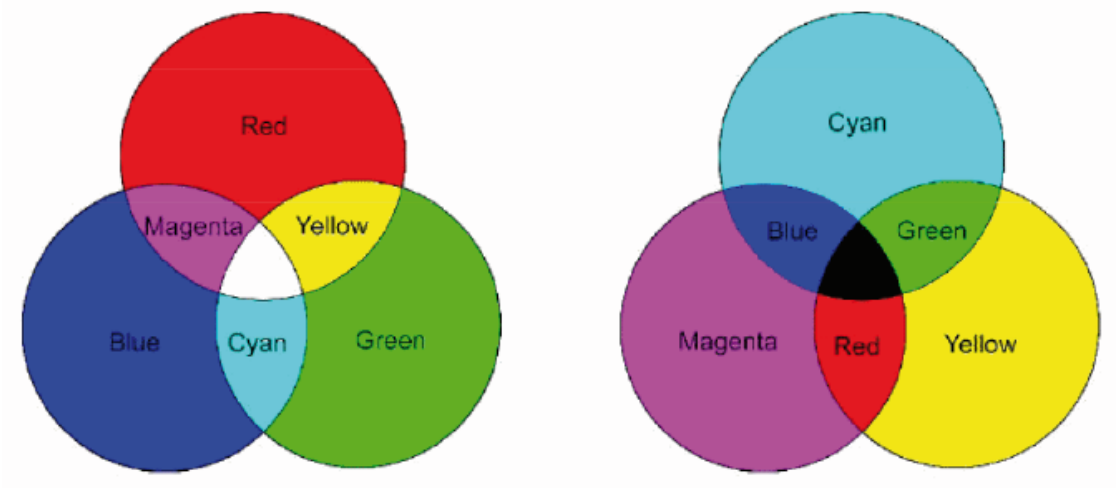
Solution: 1920 x 1080 X 30 x 24 = 1492992000 = 1.39 Gbps
HDTV uses digital signals to broadcast high quality video signals. The HDTV screen is normally a ratio of 16: 9.
There are 1920 by 1080 pixels per screen, and the screen is renewed 30 times per second. Twenty-four bits represents one color pixel.

27th Video Color Models

Color Model

1. **RGB Model**

- Generate all others colours based on three basic colours which are Red , Green, and Blue.
- (Newer) Color LCD panels (typically thin-film-transistor liquid-crystal displays (TFT LCD)): Transistor switch for each (R, G or B) pixel
- Used in display (monitor) LED, data-show etc.



2. **CMY and CMYK model**

- Combinations of additive and subtractive colors
- Cyan, Magenta, and Yellow (CMY) are complementary colors of RGB - Subtractive Primaries.
- CMY model is mostly used in printing devices where the color pigments on the paper absorb certain colors.
- E.g., convert White from (1, 1, 1) in RGB to (0, 0, 0) in CMY.
- Used in color printer

$$\begin{bmatrix} C \\ M \\ Y \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} C \\ M \\ Y \end{bmatrix}$$

(R G B)=➔((1-R)=C,(1-G)=M,(1-B)=y)

CMYK model : *An improved Printing Color Model Sometimes, an alternative CMYK model (K stands for Black) is used in color printing*

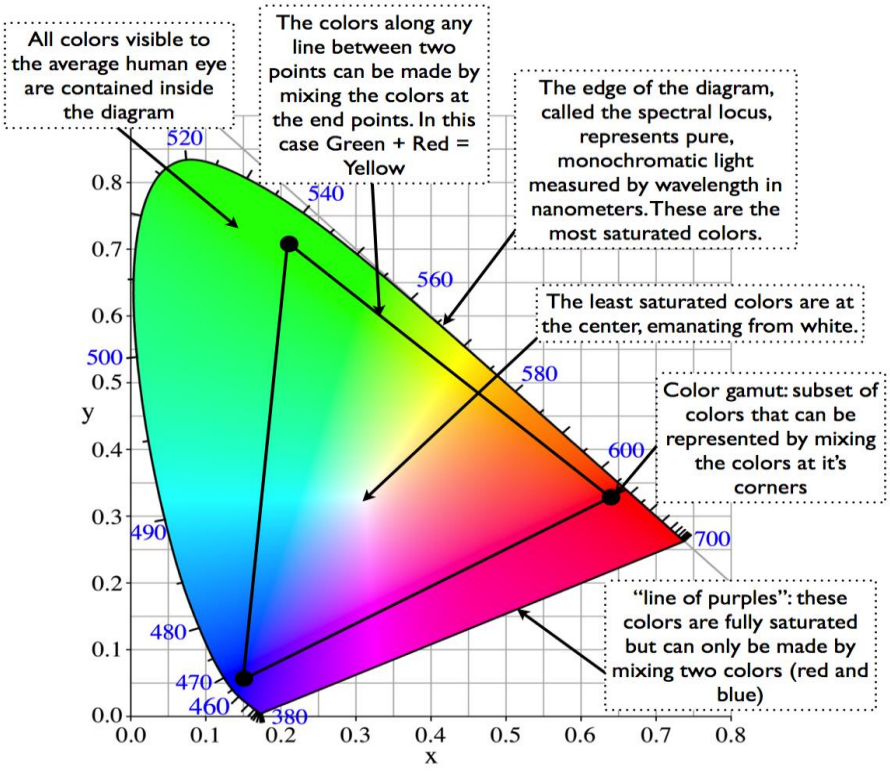
Original Color Image

C, M, Y, K image Intensities

3. CIE Color model

- In 1931, the CIE defined three standard primaries (X, Y, Z)
- The Y primary was intentionally chosen to be identical to the luminous-efficiency function of human eyes (Perceptual Model).

- All visible colors are in a horseshoe shaped cone in the X-Y-Z space. Consider the plane X+Y+Z=1 and project it onto the X-Y plane, we get the CIE chromaticity diagram.
- The edges represent the pure colors



Anatomy of a CIE Chromaticity Diagram

4. YUV Color Model

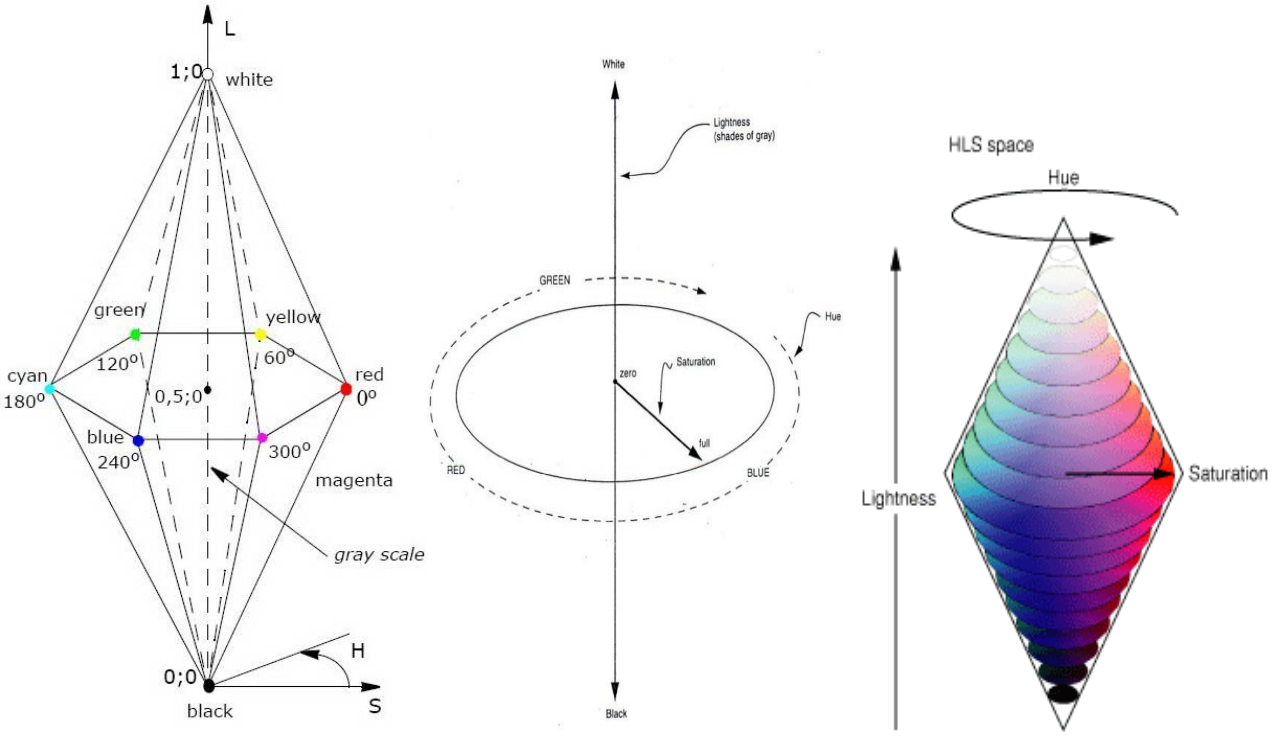
- Digital video standard established in 1982
- Video is represented by a sequence of fields (odd and even lines). Two fields make a frame.
- Works in PAL (50 fields/sec) or NTSC (60 fields/sec)
- Uses the Y, U, V color space.

$$\begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.299 & -0.587 & 0.886 \\ 0.701 & -0.587 & -0.114 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

5. HSL color model

For example, the **H**ue/**S**aturation/**L**ightness (HSL color transform allows us to describe colors in terms that became more readily understand (see Figure (2-b). The ***lightness*** is the brightness of the color, and the ***hue*** is what we normally think of as "color" (for example green, blue, or orange). The ***saturation*** is a measure of how much white is in the color (for example, pink is red with more white, so it is less saturated than a pure red). Most people can relate to this method of describing color. For example, "a deep, bright orange" would a 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 component R=245, G=110, and B = 20, most people would have no idea how this color appears. Because the HSL color space was developed based on heuristics relating to human perception, various methods are available to transform RGB pixel values into the HSL color space. Most of these are algorithmic in nature and are geometric approximations to mapping the RGB color cube into some HSL-type color space.



Figure(2) a HSL image

Video Compression

Why need video compression?

Imagine that One movie video without compression 720 x 480 pixels per frame Full color with 30 frames per second for 90 minutes needs 167.96 G Bytes !! (storage[1]). In addition, the bit rate for high-definition TV (HDTV) needs 1.39 Gbps. Which is huge bit rate required. (transmssion[2]).

$$Compression\ Ratio = \frac{Uncompressed\ File\ Size}{Compressed\ File\ Size} = \frac{size_U}{size_C}$$

The following techniques are commonly employed to achieve desirable reductions in image data:

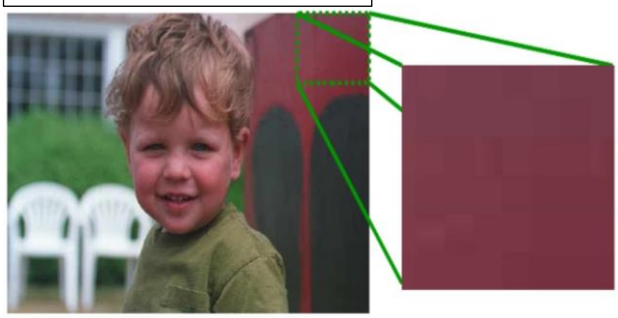
1) Reduce color nuances within the image
2) Reduce the color resolution with respect to the prevailing light intensity
3) Remove small, invisible parts, of the picture
4) Compare adjacent images and remove details that are unchnaged between two images

The first three are image based compression techniques, where only one frame is evaluated and compressed at a time. The main one that does the real image compression is the **Discrete Cosine Transform (DCT)** followed by a quantization that removes the redundant information (the "invisible" parts).

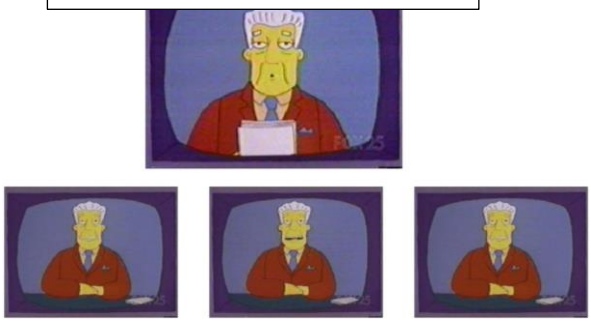
In video there is redundancy data.

a) **Spatial redundancy**: take advantage of similarity among most neighboring pixls
b) **Temporal redundancy:** take advantage of similarity between frames

a)Spatial redundancy



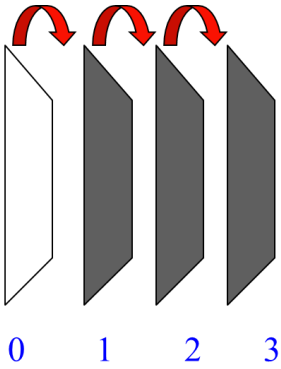
b)Temporal redundancy



To compress video should be either within a frame or multi-frame

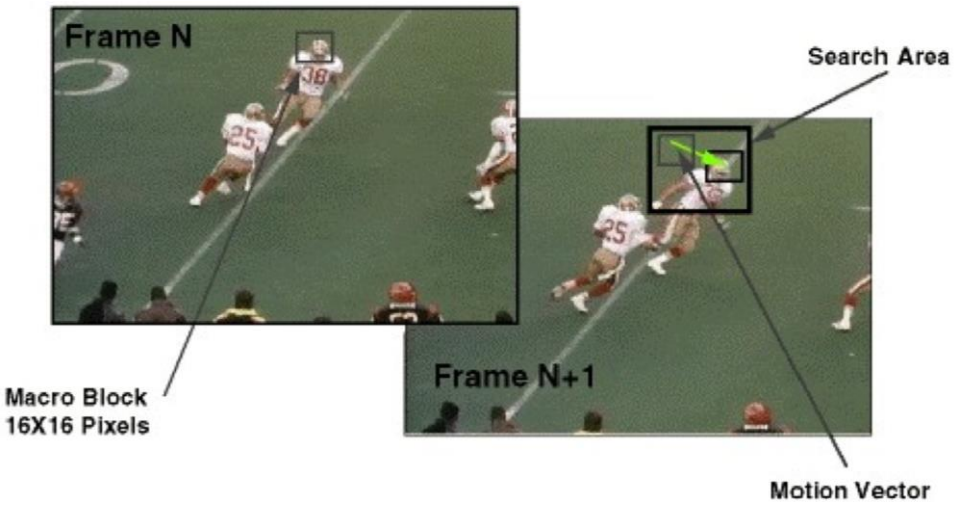
1- Compresses each frame in isolation, treating it as a bitmapped image. Based on quantization of DCT coefficients

2- Compresses sequences of frames by only storing differences between them. Based on Motion Compensation (MC).



$Frame_0$ (still image)
Difference $frame_1 = (Frame_1 – Frame_0)$
Difference $frame_2 = (Frame_2 – Frame_1)$
If no movement in the scene, all difference frames are 0.
Can be greatly compressed!

A motion vector (MV) describes the offset between the location of the block being coded (in the current frame) and the location of the best-match block in the reference frame



An obvious solution for video compression would be predictive coding based on previous frames. For example, suppose we simply created a predictor such that the prediction equals the previous frame. Then compression proceeds by subtracting images: instead of subtracting the image from itself (i.e., use a derivative), we subtract in time order and code the residual error. And this works. Suppose most of the video is unchanging in time.

The idea of looking for the football player in the next frame is called motion estimation, and the concept of shifting pieces of the frame around so as to best subtract away the player is called motion compensation.

The basic principle for video compression is the image-to-image prediction. The first image is called an I-frame and is self-contained, having no dependency outside of that image. The following frames may use part of the first image as a reference. An image that is predicted from one reference image is called a P-frame and an image that is bidirectionally predicted from two reference images is called a B-frame.

- **I**-frames: I (**I**ntracoded) frames, self-contained
- **P**-frames: (**P**redicted) from last I or P reference frame

- **B**-frames: (**B**idirectional); predicted from two references one in the past and one in the future, and thus out of order decoding is needed

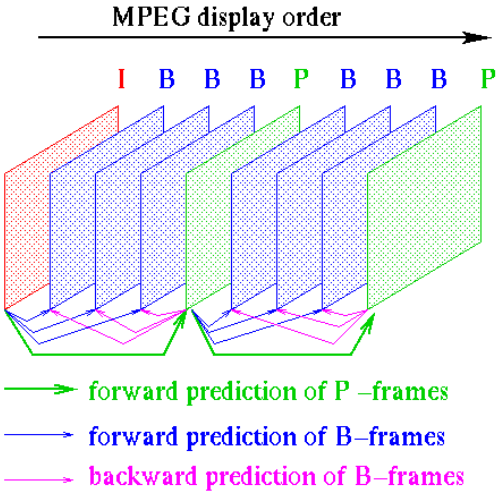
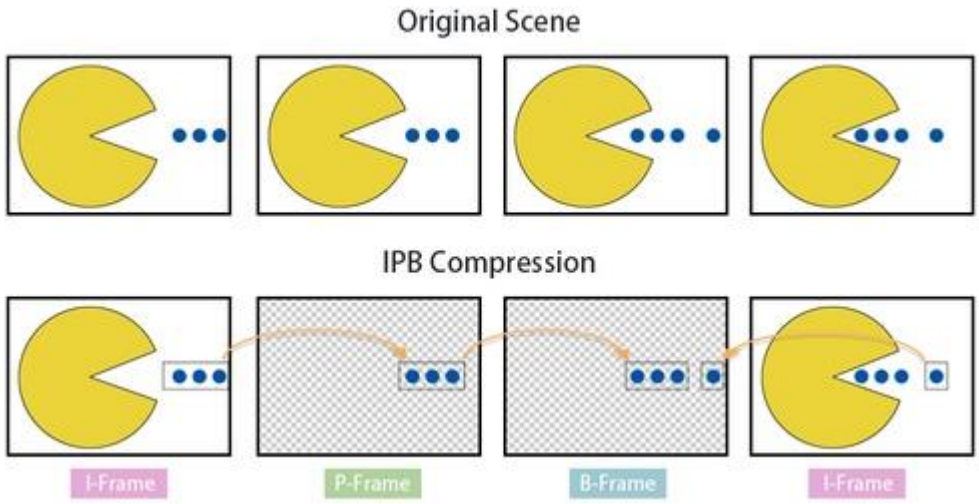


Figure: The illustration above shows how a typical sequence with I-, B-, and P-frames may look. Note that a P-frame may only reference a preceding I- or P-frame, while a B-frame may reference both preceding and succeeding I- and P-frames.

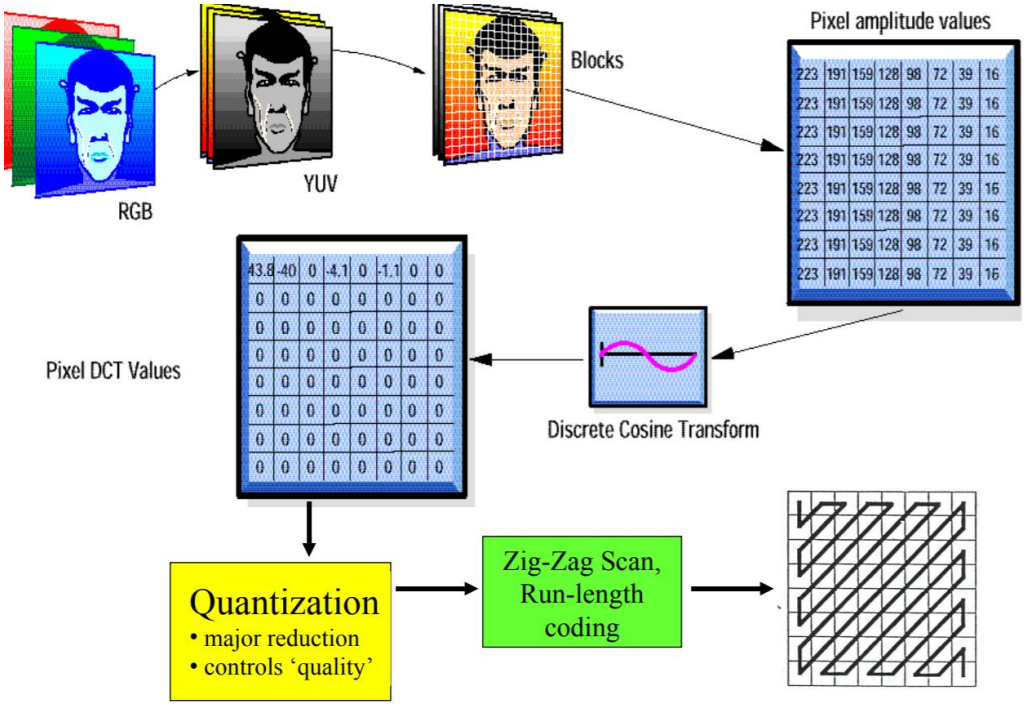


The video decoder restores the video by decoding the bit stream frame by frame. Decoding must always start with an I-frame, which can be decoded independently, while P- and B-frames must be decoded together with current reference image(s).

Steps to compress video

1- Divide Image to blocks (16x16 or 8x8)
2- Use DCT based techniques for spatial redundancy removal (Intra-frame compression).
3- Use MC (Motion Compensation) techniques for temporal redundancy removal (Inter-frame compression).
4- Final stage is two dimensional run-length coding.

Encoding of I-frames


RGB
YUV
Blocks
Pixel amplitude values
223 191 159 128 98 72 39 16
Discrete Cosine Transform
Pixel DCT Values
43.8 -40 0 -4.1 0 -1.1 0 0
Quantization
• major reduction
• controls 'quality'
Zig-Zag Scan,
Run-length
coding