SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

SCHOOL OF ELECTRICAL AND ELECTRONICS

DEPARTMENT OF ELECTRONICS AND COMMMUNICATIONENGINEERING

UNIT - I

SCSA3007- INTRODUCTION TO VISUAL COMPUTING

UNIT 1 VISUAL COMPUTING INTRODUCTION

Physiological Foundations-Representation of Light and Color-Image and Noise Models- Basics of Fourier series, Sampling Theorem-Vector Quantization- k-Means Clustering- Mixture Models- Gray Level and Color Quantization-Relationships between Pixels- Camera Geometry-2D and 3DTransforms, Projections**.**

Visual Computing

Visual computing is an emerging discipline born mainly from the intense cross-pollination of computational geometry, computer graphics, and computer vision that has been attested over the past few years. Many similar techniques have been independently discovered in those well-established areas.

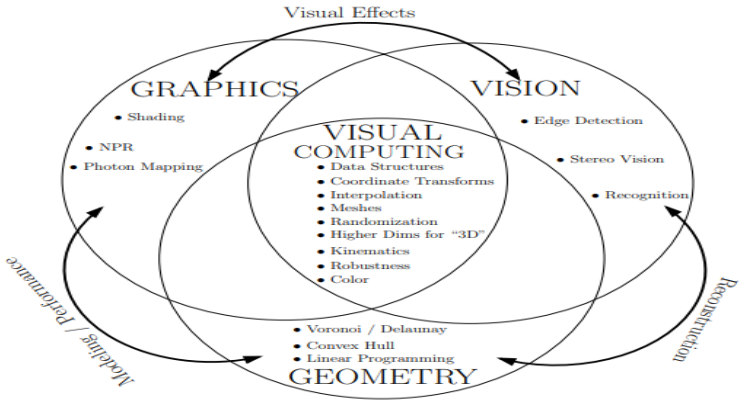


Fig 1.overview of visual computing

Physiological Foundations

computer graphics is producing images and videos that are ultimately perceived by a human, it's mandatory to account for how the human visual system (HVS) is processing this information. The HVS is complex, exhibiting many non-linearities as well as feedback and is only partially understood. While this posses a challenge, it can also be seen as an opportunity which can be exploited in image compression, watermarking, denoising, enhancement, upsampling, etc.

Computational models which can predict the human response to the distortion of visual content are important that depend more on physiological characteristics of brain.

Representation of Light and Color

What the human eye (or virtual camera) sees is a result of light coming off of an object or other light source and striking receptors in the eye. In order to understand and model this process, it is necessary to understand different light sources and the ways that different materials reflect those light sources.

The perceived light intensity and chromaticity via our photoreceptors: cones and rods.

Cones primarily responsible for our photopic vision. They are tuned to specific light wavelengths responsible for color sensing. The sensation of color depends primarily on the composition of light which is a mixture of white light and colored light (which in itself can be a mixture of wavelengths as in the case of purple). The colored light may have a dominant wavelength or **hue** and the extent to which the hue dominates is known as saturation (or chroma). The saturation decreases as the hue is deleted with white light.

There are 3 receptors in the eye that respond to different wavelengths. This leads to attempts to chart colors by a mixture of three primary lights. Figure 2 shows color triangle with the three apexes representing three primary colored lights: blue-violet, orange-red, and green. A great number, but not all colors can be produced by mixing lights of the three primary colors. A specific color, for example an unsaturated greenish blue, can be represented by a point on the triangular grid.

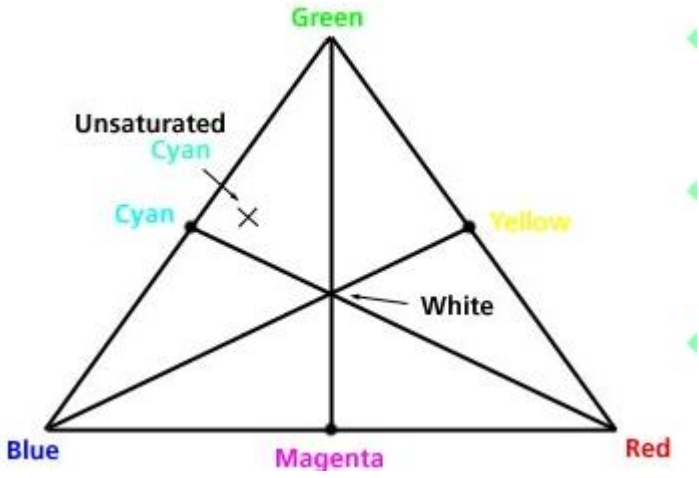


Fig 2. The color triangle

In order to represent all colors, 3 imaginary or "ideal" primaries are used. The Commission Internationale de l'Eclairage (CIE) defined in 1931 (modified in 1967) the chromaticity curve with standard observer and 3 ideal standard sources. The chromaticity diagram is constructed

(Fig. 3) by drawing a color triangle with 3 ideal (but non-existent) primary colors at each corner. The x-axis is the amount of ideal green that would be mixed with blue. The y-axis is the amount of ideal red that would be mixed with blue. A given color is represented by values along the two axes.

Superimposed on the triangle is the CIE chromaticity curve which places the band of pure spectral colors as a solid curved-line from violet up to green down to red. The dashed line connecting 380 nm and 700 nm are the nonspectral colors of purple obtained by mixing violet and red light beams. All the colors that we can see are contained within the area bounded by the solid and dashed lines. The central point W of the diagram is the white produced by an equal mixture of the three primaries.

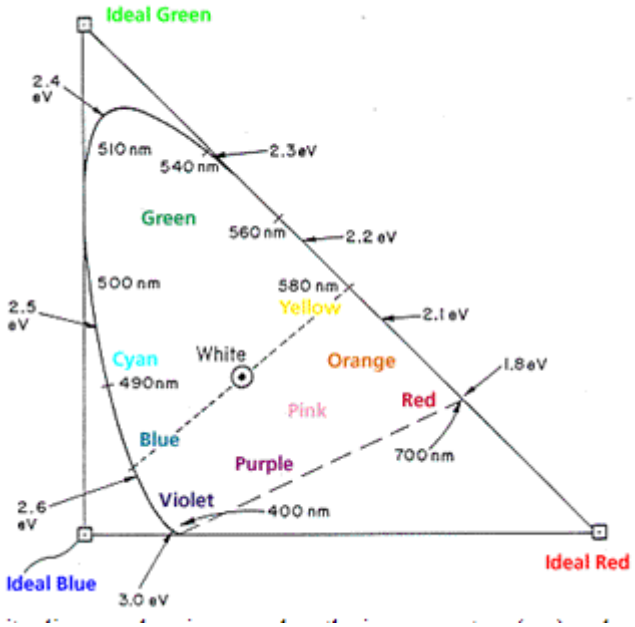


Fig 3. The CIE chromaticity diagram showing wavelengths in nanometers (nm) and energies in electron volts (eV).

The area enclosed by the curved line and dashed segment in figure 3 include all visibile colors. The pure spectral colors lie along the curved edge. A wavelength is denoted with the greek letter λ (lambda). Visible light is made of waves which frequency varies from 380 to about 740 nanometres (a nanometre is 1×10−91×10−9 meter). Any waves which wavelength is below 380 nm or above 740 nm can not be perceived by the human eye. If all the light colors from the visible light spectrum and add them up in the same proportions, then we can recreate white light (figure 2). White light as such doesn't exist. White light is the result of a light source, the sun or the screen of your computer, producing a mixture of light colors from the visible spectrum. In computer screen or television with a magnifying glass, the tiny dots seen is ,

probably red, green and blue and by mixing these colors in different amounts, a large range of colors can be produced.

Light is a Electromagnetic radiation (EMR) moving along rays in space R(l) is EMR, measured in units of power (watts). Light field is describe as the light in the scene by specifying the radiation (or "radiance" along all light rays) arriving at every point in space and from every direction.

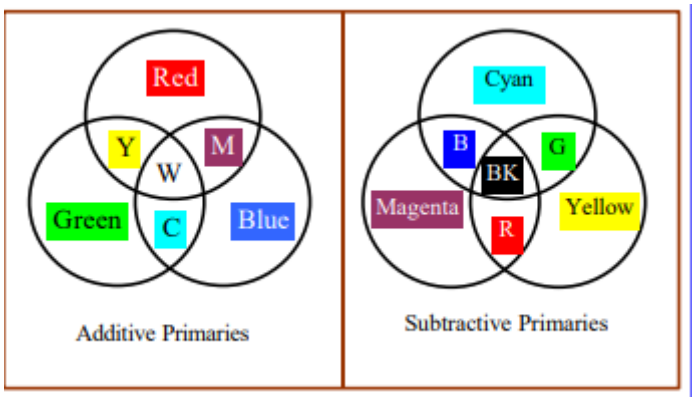


Fig 4 .Diagram of color mixing

Light Properties

Leaving aside ambient light, the light in an environment comes from a light source such as a lamp or the sun. In fact, a lamp and the sun are examples of two essentially different kinds of light source: a **point light** and a **directional light**. A point light source is located at a point in 3D space, and it emits light in all directions from that point. For a directional light, all the light comes from the same direction, so that the rays of light are parallel. The sun is considered to be a directional light source since it is so far away that light rays from the sun are essentially parallel when they get to the Earth .

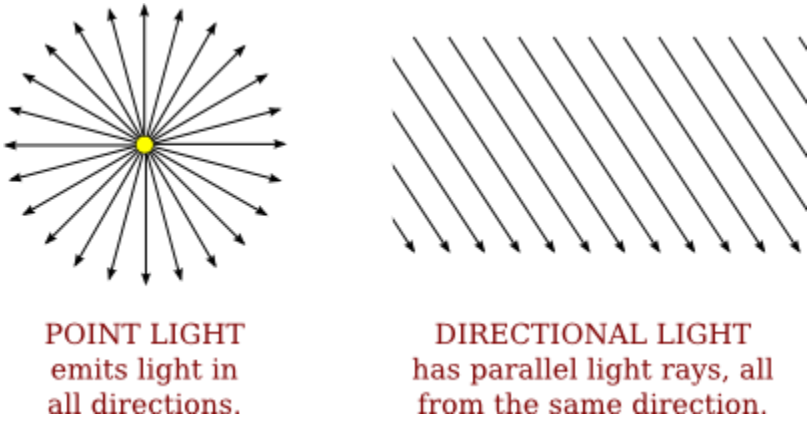


Fig 5 .Types of light sources

A light can have color. In fact, in OpenGL, each light source has three colors: an ambient color, a diffuse color, and a specular color. Just as the color of a material is more properly referred to as reflectivity, color of a light is more properly referred to as **intensity** or energy. More exactly, color refers to how the light's energy is distributed among different wavelengths. Real light can contain an infinite number of different wavelengths; when the wavelengths are separated, you get a spectrum or rainbow containing a continuum of colors. Light as it is usually modeled on a computer contains only the three basic colors, red, green, and blue. So, just like material color, light color is specified by giving three numbers representing the red, green, and blue intensities of the light.

The diffuse intensity of a light is the aspect of the light that interacts with diffuse material color, and the specular intensity of a light is what interacts with specular material color. It is common for the diffuse and specular light intensities to be the same.

The ambient intensity of a light works a little differently. Recall that ambient light is light that is not directly traceable to any light source. Still, it has to come from somewhere and we can imagine that turning on a light should increase the general level of ambient light in the environment. The ambient intensity of a light in OpenGL is added to the general level of ambient light. (There can also be global ambient light, which is not associated with any of the light sources in the scene.) Ambient light interacts with the ambient color of a material, and this interaction has no dependence on the position of the light sources or viewer. So, a light doesn't have to shine on an object for the object's ambient color to be affected by the light source; the light source just has to be turned on.

Image and Noise Models

Digital images

An image may be defined as a two-dimensional function f(xy), where x and y are spatial coordinates, and the value of f at any pair of coordinates (xy) is called the intensity of the image at that point. An image may be continuous with respect to x and y, and also in intensity (analog image). Converting such an image to digital form requires that the coordinates and the intensity be digitized. Digitizing the coordinates is called sampling, while digitizing the intensity is referred to as quantization. Thus, when all this quantities are discrete , the image a digital image. The size of the image is the number of rows by the number of columns, M X N. The indexation of the image follows the following conventions:

Usual $\rightarrow$
$$\begin{pmatrix} a(0,0) & a(0,1) & \dots & a(0,N-2) & a(0,N-1) \\ a(1,0) & a(1,1) & \dots & a(1,N-2) & a(1,N-1) \\ \dots & \dots & \dots & \dots & \dots \\ a(M-1,0) & a(M-1,1) & \dots & a(M-1,N-2) & a(M-1,N-1) \end{pmatrix} \qquad \begin{pmatrix} a(1,1) & a(1,2) & \dots & a(1,N-1) & a(1,N) \\ a(2,1) & a(2,2) & \dots & a(2,N-1) & a(2,N) \\ \dots & \dots & \dots & \dots & \dots \\ a(M,1) & a(M,2) & \dots & a(M,N-1) & a(M,N) \end{pmatrix}$$
$\leftarrow$ Matlab

Digital image processing deals with manipulation of digital images through a digital computer. It is a subfield of signals and systems but focus particularly on images. DIP focuses on developing

a computer system that is able to perform processing on an image. The input of that system is a digital image and the system process that image using efficient algorithms, and gives an image as an output. The most common example is Adobe Photoshop. It is one of the widely used application for processing digital images.

Digital images are often corrupted by different types of noise during its acquisition and transmission phase. Such degradation negatively influences the performance of many image processing techniques and a preprocessing module to filter the images is often required. Noise removal is one of the major concerns in the field of computer vision and image processing. Images are often contaminated by impulsive noise due to noisy sensors or channel transmission errors or faulty storage hardware. The goal of removing impulsive noise is primarily to suppress the noise as well as to preserve the integrity of edges and detailed information.

NOISE MODELS

Spatially independent noise models

– Gaussian noise

– Rayleigh noise

– Erlang (Gamma) noise

– Exponential noise

– Impulse (salt-and-pepper) noise

Spatially dependent noise model

– Periodic noise

Gaussian Noise Model: It is also called as electronic noise because it arises in amplifiers or detectors. Gaussian noise caused by natural sources such as thermal vibration of atoms and discrete nature of radiation of warm objects. Gaussian noise generally disturbs the gray values in digital images. That is why Gaussian noise model essentially designed and characteristics by its PDF or normalizes histogram with respect to gray value. This is given as

$$P(g) = \sqrt{\frac{1}{2\pi\sigma^2}} e^{-\frac{(g-\mu)^2}{2\sigma^2}}$$

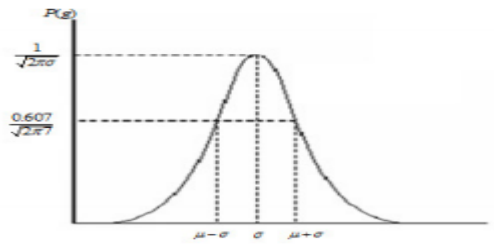


Fig 6. Gaussian distribution

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

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

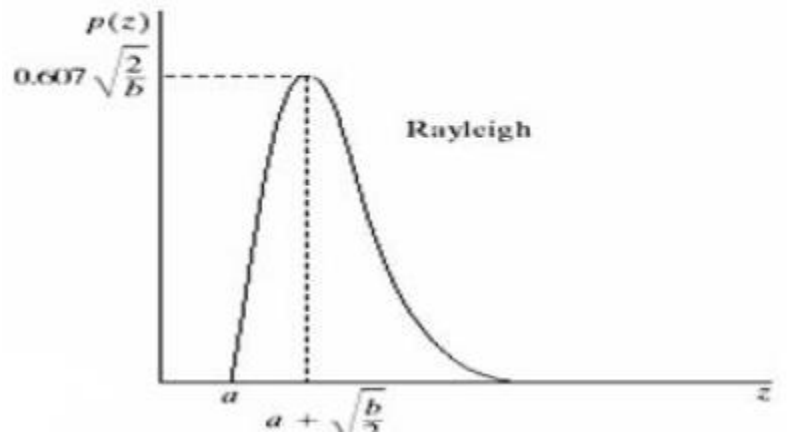


Fig 7 Rayleigh distribution

Erlang (gamma) noise The probability density function of Erlang noise is given by

$$p(z) = \begin{cases} \frac{a^b z^{b-1}}{(b-1)!} e^{-az} & \text{for } z >= 0 \\ 0 & \text{for } z < 0 \end{cases} \qquad \mu = \frac{b}{a} \quad \sigma^2 = \frac{b}{a^2}$$

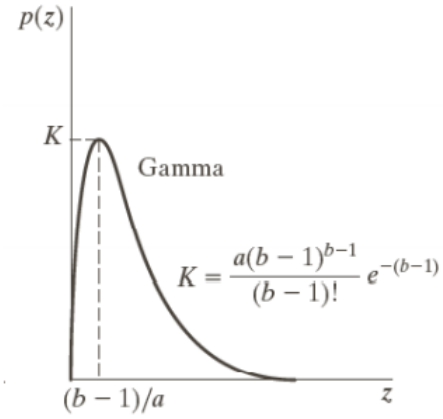


Fig 8 probability density function of Erlang noise

Its shape is similar to Rayleigh disruption.

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

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

$$\mu = \frac{1}{a}$$

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

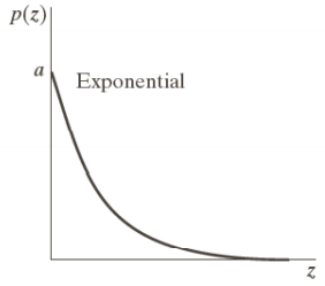


Fig 9 probability density function of Exponential Noise

Uniform noise is not often encountered in real-world imaging systems, but provides a useful comparison with Gaussian noise. The linear average is a comparatively poor estimator for the mean of a uniform distribution. This implies that nonlinear filters should be better at removing uniform noise than Gaussian noise. The Uniform pdf is given by:

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

$$\mu = \frac{a+b}{2}$$

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

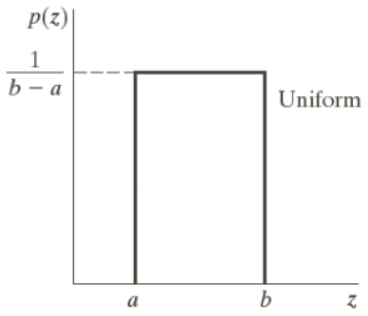


Fig 9 probability density function of Uniform noise

Impulse (Salt and Pepper)Noise: In this case, the noise is signal dependent, and is multiplied to the image. The PDF of bipolar (impulse) noise is given by

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

if b>a, gray level b will appear as a light dot in image. Level a will appear like a dark dot. If either Pa or Pb is zero , the impulse noise is called unipolar. If neither probability is zero and especially if they are approximately equal, the impulse noise values will resemble salt and pepper granules randomly distributed over the image.

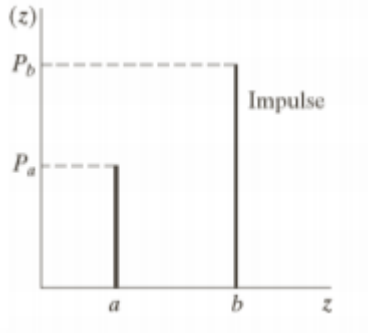


Fig 9 probability density function of Impulse noise

Periodic Noise Periodic noise in an image arises typically from electrical or electromechanical interference during image acquisition. The periodic noise can be reduced significantly via frequency domain filtering.

Estimation of Noise Parameters The parameters of periodic noise can be estimated by inspection of the Fourier spectrum of the image. Periodic noise tends to produce frequency

spikes, which are detectable even by visual analysis. In simplistic cases, it is also possible to infer the periodicity of noise components directly from the image. Automated analysis is possible if the noise spikes are either exceptionally pronounced, or when knowledge is available about the general location of the frequency components of the interference. It is often necessary to estimate the noise probability density functions for a particular imaging arrangement. When images already generated by a sensor are available, it may be possible to estimate the parameters of the probability density functions from small patches of reasonably constant background intensity.

The simplest use of the data from the image strips is for calculating the mean and variance of intensity levels. Let S denote a stripe and PS (Zi), i = 0,1,2,...,L- 1, denote the probability estimates of the intensities of the pixels in S , then the mean and variance of the pixels in S are

$$\bar{z} = \sum_{i=0}^{L-1} z_i p_S(z_i)$$

and

$$\sigma^2 = \sum_{i=0}^{L-1} (z_i - \bar{z})^2 p_S(z_i) .$$

The shape of the histogram identifies the closest probability density function match. The Gaussian probability density function is completely specified by these two parameters.

Basics of Fourier series

Fourier Series of Periodic Functions (Almost) any periodic function g(x) with fundamentalλ frequency ω0 can be described as a sum of sinusoids

$$g(x) = \sum_{k=0}^{\infty} [A_k \cos(k\omega_0 x) + B_k \sin(k\omega_0 x)]$$

Infinite sum of — Cosines — Sines

This infinite sum is called a Fourier Series Summed sines and cosines are multiples of the fundamentalλ frequency (harmonics) Ak and Bk called Fourier coefficients. Not known initially but derived from original function g(x) during Fourier analysis

THE TWO DIMENSIONAL FOURIER TRANSFORM

The Fourier transform is extended to a function f (x, y) of two variables. If f (x, y) is continuous and integrable and F(u, v) is integrable, the following Fourier transform pair exists:

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

$$f(x,y) = \frac{1}{(2\pi)^2}\int_{-\infty}^{\infty}\int F(u,v)e^{j2\pi(ux+vy)}\,dudv$$

In general F(u, v) is a complex-valued function of two real frequency variables u, v and hence, it can be written as:

$$F(u,v) = R(u,v) + jI(u,v)$$

The amplitude spectrum, phase spectrum and power spectrum, respectively, are defined as follows.

SAMPLING THEOREM

The most basic requirement for computer processing of images is that the images must be available in digital form i.e. arrays of integer numbers. For digitization the given image is sampled on a discrete grid and each sample or pixel is quantized to an integer value representing a gray level. The digitized image can then be processed by the computer.

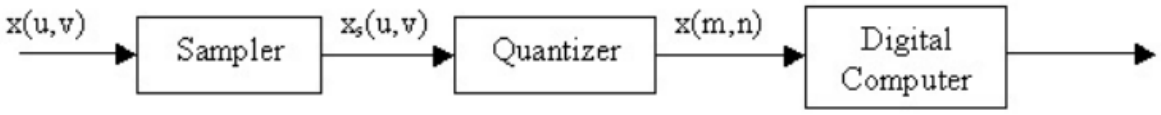


Fig 10. Steps involved in converting analog to digital image

2-D Sampling Theorem Definition: An image x(u, v) is called "bandlimited" if its FT X(ω1, ω2)

$$X(\omega_1,\omega_2) = 0 \quad |\omega_1| > \omega_{10}, |\omega_2| > \omega_{20}$$

is zero outside a bounded region in the frequency plane i.e.

ω_{10}, ω_{20}: Bandlimits of the image. If the spectrum is circularly symmetric then the single spatial frequency $[\omega_0 \stackrel{\triangle}{=} \omega_{10} = \omega_{20}]$ is the bandwidth.

$$s_a(u,v;\Delta u,\Delta v) \triangleq \sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty}\delta(u-m\Delta u, v-n\Delta v)$$

with the "sampling intervals" $\Delta u, \Delta v$. The sampled image is

$$\begin{aligned} x_s(u,v) &= x(u,v)s_a(u,v;\Delta u,\Delta v) \\ &= \sum\sum_{m,n=-\infty}^{\infty} x(m\Delta u, n\Delta v)\delta(u-m\Delta u, v-n\Delta v) \end{aligned}$$

$\Delta u, \Delta v$ is another sampling function with spacing $\frac{2\pi}{\Delta u}, \frac{2\pi}{\Delta v}$. Then, using the convolution in frequency domain we get

$$X_s(\omega_1,\omega_2) = \frac{1}{\Delta u \Delta v}\sum\sum_{k,l=-\infty}^{\infty} X(\omega_1 - \frac{2\pi k}{\Delta u}, \omega_2 - \frac{2\pi l}{\Delta v})$$

Let us define discrete frequency variables (i.e. in discrete domain) $\Omega_1 \triangleq \omega_1 \Delta u, \Omega_2 \triangleq \omega_2 \Delta v$, then we get 2-D Discrete Space Fourier Transform (DSFT),

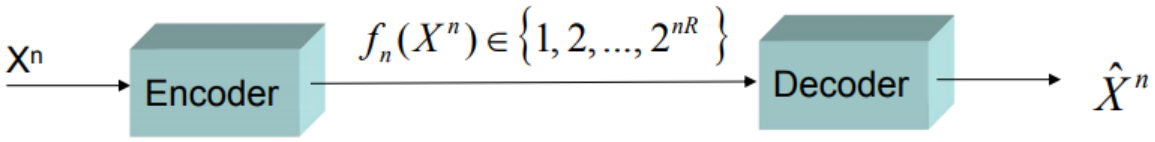
$$X_s(\Omega_1,\Omega_2) = \frac{1}{\Delta u \Delta v}\sum\sum_{k,l=-\infty}^{\infty} X(\frac{\Omega_1 - 2\pi k}{\Delta u}, \frac{\Omega_2 - 2\pi l}{\Delta v})$$

Undersampling and Aliasing Effects If the sampling frequencies are below the Nyquist rates i.e. ωus < 2ω10 and ωvs < 2ω20, then the periodic replications of X(ω1, ω2) will overlap, resulting in a distorted spectrum Xs(ω1, ω2) from which X(ω1, ω2) cannot be recovered. The frequencies above half the sampling frequencies, that is, above ωus 2 , ωvs 2 are called the "fold-over frequencies". This overlapping causes some of the high frequencies (or fold-over frequencies) in the original image to appear as low frequencies (below ωus/ 2 , ωvs /2) in the sampled image. This phenomenon is called "Aliasing". Aliasing cannot be removed by post filtering but can be avoided by pre low pass filtering the image so that its bandlimits are less than one-half of the sampling frequencies. In images aliasing causes edge smearing and loss of details.

VECTOR QUANTIZATION

Quantization is the process of mapping a continuous or discrete scalar or vector, produced by a source, into a set of digital symbols that can be transmitted or stored using a finite number of

bits. In the case of continuous sources (with values in R or R n) quantization must necessarily be used if the output of the source is to be communicated over a digital channel.

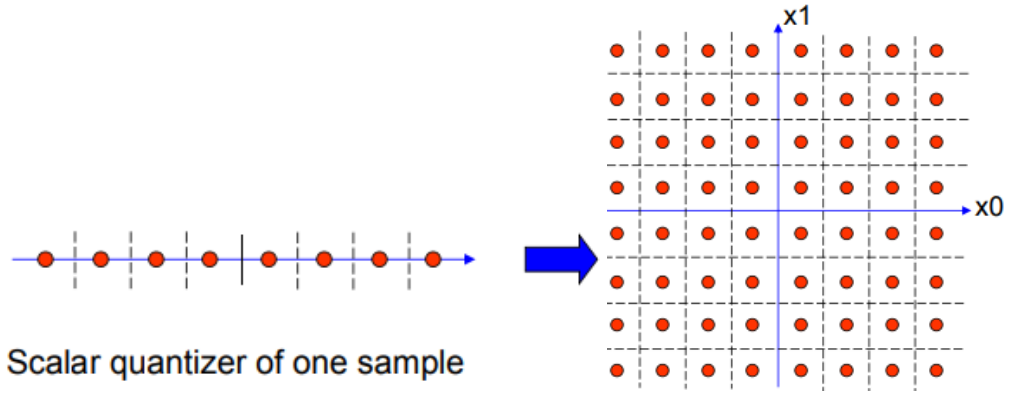


Encoder: Represent a sequence Xn = {X1, X2, …, Xn} by an index. fn(Xn) in {1, 2, …, 2nR}.

Codebook: the collection of all codewords.

Decoder: Map fn(Xn) to a reconstruction sequence (codeword).

Consider the quantization of two neighboring samples of a source: 8 quantization bins. Bit rate: 3 bits / sample. If uniform scalar quantization is used for each sample, the 2-D sampling space is partitioned into 64 rectangular regions



Vector quantization is a form of data compression that represents data vectors by a smaller set of codebook vectors. Each data vector is then represented by its nearest codebook vector. The goal of vector quantization is to represent the data with the fewest code book vectors while losing as little information as possible.

Vector quantization of unlabelled data seeks to minimize the reconstruction error. This can be accomplished with Competitive learning an iterative learning algorithm for vector quantization that has been shown to perform gradient descent on the following energy function

$$\int ||x - w_{s^*(x)}||^2 p(x) dx.$$

where p(x) is the probability distribution of the input patterns and Ws are the reference or codebook vectors and s*(x) is defined by $\|x - w_{s^*(x)}\| \leq \|x - w_i\|$ (for all i). This minimizes the square reconstruction error of unlabelled data and may work reasonably well for classification tasks if the patterns in the different classes are segregated. In many classification tasks, however, the different member patterns may not be segregated into separate clusters for each class. In these cases it is more important that members of the same class be represented by the same codebook vector than that the reconstruction error is minimized. To do this, the quantizer can make use of the labelled data to encourage appropriate quantization.

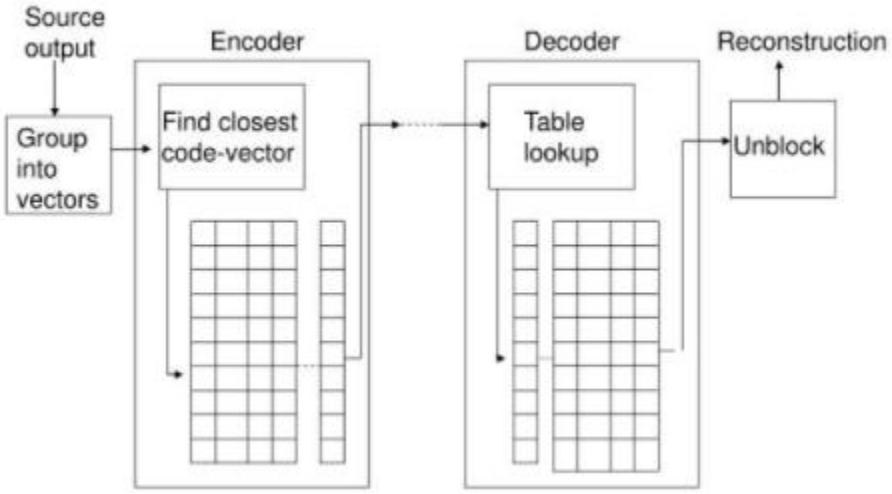


Fig 12 block diagram for Vector quantization

K-MEANS CLUSTERING

Clustering: The organization of unlabeled data into similarity groups called clusters. A cluster is a collection of data items which are "similar" between them, and "dissimilar" to data items in other clusters.

Fig.13 Clustering

K-means algorithm is an iterative algorithm that tries to partition the dataset into Kpre-defined distinct non-overlapping subgroups (clusters) where each data point belongs to only one group. It

tries to make the intra-cluster data points as similar as possible while also keeping the clusters as different (far) as possible. It assigns data points to a cluster such that the sum of the squared distance between the data points and the cluster's centroid (arithmetic mean of all the data points that belong to that cluster) is at the minimum. The less variation we have within clusters, the more homogeneous (similar) the data points are within the same cluster.

In k-means clustering, it partitions a collection of data into a k number group of data11, 12. It classifies a given set of data into k number of disjoint cluster. K-means algorithm consists of two separate phases. In the first phase it calculates the k centroid and in the second phase it takes each point to the cluster which has nearest centroid from the respective data point. There are different methods to define the distance of the nearest centroid and one of the most used methods is Euclidean distance. Once the grouping is done it recalculate the new centroid of each cluster and based on that centroid, a new Euclidean distance is calculated between each center and each data point and assigns the points in the cluster which have minimum Euclidean distance. Each cluster in the partition is defined by its member objects and by its centroid. The centroid for each cluster is the point to which the sum of distances from all the objects in that cluster is minimized. So K-means is an iterative algorithm in which it minimizes the sum of distances from each object to its cluster centroid, over all clusters. Let us consider an image with resolution of X x Y and the image has to be cluster into k number of cluster. Let p(x, y) be an input pixels to be cluster and ck be the cluster centers. The algorithm for k-means clustering is following as:

1. Initialize number of cluster k and centre.

2. For each pixel of an image, calculate the Euclidean distance d, between the center and each pixel of an image using the relation given below.

$$d = \| p(x, y) - c_k \|$$

3. Assign all the pixels to the nearest centre based on distance d.

4. After all pixels have been assigned, recalculate new position of the centre using the relation given below.

$$c_k = \frac{1}{k} \sum_{y \in c_k} \sum_{x \in c_k} p(x, y)$$

5. Repeat the process until it satisfies the tolerance or error value.

6. Reshape the cluster pixels into image.

Although k-means has the great advantage of being easy to implement, it has some drawbacks. The quality of the final clustering results is depends on the arbitrary selection of initial centroid. So if the initial centroid is randomly chosen, it will get different result for different initial centers. So the initial center will be carefully chosen so that we get our desire segmentation. And also computational complexity is another term which we need to consider while designing the K-means clustering. It relies on the number of data elements, number of clusters and number of iteration.

Disadvantages

1) The learning algorithm requires apriori specification of the number of cluster centers.

2) The use of Exclusive Assignment - If there are two highly overlapping data then k-means will not be able to resolve that there are two clusters.

3) The learning algorithm is not invariant to non-linear transformations i.e. with different representation of data we get different results (data represented in form of cartesian co-ordinates and polar co-ordinates will give different results).

4) Euclidean distance measures can unequally weight underlying factors.

5) The learning algorithm provides the local optima of the squared error function.

6) Randomly choosing of the cluster center cannot lead us to the fruitful result.

7) Applicable only when mean is defined i.e. fails for categorical data.

8) Unable to handle noisy data and outliers.

9) Algorithm fails for non-linear data set.

Gaussian Mixture Models (GMM)

k-means does not account for variance. By variance, we are referring to the width of the bell shape curve. A *Gaussian Mixture* is a function that is comprised of several Gaussians, each identified by $k \in \{1,\ldots, K\}$, where K is the number of clusters of our dataset. Each Gaussian k in the mixture is comprised of the following parameters:

- A mean μ that defines its centre.
- A covariance Σ that defines its width. This would be equivalent to the dimensions of an ellipsoid in a multivariate scenario.
- A mixing probability π that defines how big or small the Gaussian function will be.

The GMM is well suited to modeling clusters of points. Each cluster is assigned a Gaussian, with its mean somewhere in the middle of the cluster, and a standard deviation that measures the spread of that cluster. The GMM is an unsupervised clustering method. It can extract coherent regions in feature space and corresponding meaningful structures in the input data space, where each region is represented by a Gaussian distribution.

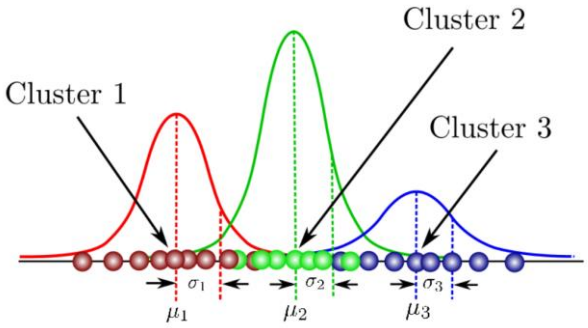


Fig 14.parameters of Gaussian Mixture model

The mixing coefficients are themselves probabilities and must meet this condition:

$$\sum_{k=1}^{K} \pi_k = 1$$

In general, the Gaussian density function is given by:

$$\mathcal{N}(\mathbf{x}|\mu,\Sigma) = \frac{1}{(2\pi)^{D/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mu)^T\Sigma^{-1}(\mathbf{x}-\mu)\right)$$

Where **x** represents our data points, *D* is the number of dimensions of each data point. μ and Σ are the mean and covariance, respectively.

A Gaussian mixture model is a weighted sum of M component Gaussian densities as given by the equation

$$p(\mathbf{x}|\lambda) = \sum_{i=1}^{M} w_i \, g(\mathbf{x}|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i),$$

where x is a D-dimensional continuous-valued data vector (i.e. measurement or features), wi , i = 1, . . . , M, are the mixture weights, and g(x|μi , Σi), i = 1, . . . , M, are the component Gaussian densities. Each component density is a D-variate Gaussian function of the form,

$$g(\mathbf{x}|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) = \frac{1}{(2\pi)^{D/2}|\Sigma_i|^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_i)' \, \Sigma_i^{-1} \, (\mathbf{x}-\boldsymbol{\mu}_i)\right\},$$

with mean vector μi and covariance matrix Σi . The mixture weights satisfy the constraint that $\sum_{i=1}^{M} w_i = 1.$. The complete Gaussian mixture model is parameterized by the mean vectors, covariance matrices and mixture weights from all component densities. These parameters are collectively represented by the notation,

$$\lambda = \{w_i, \, \boldsymbol{\mu}_i, \, \Sigma_i\} \qquad i = 1, \ldots, M.$$

The covariance matrices, Σi , can be full rank or constrained to be diagonal. Additionally, parameters can be shared, or tied, among the Gaussian components, such as having a common covariance matrix for all components, The choice of model configuration (number of components, full or diagonal covariance matrices, and parameter tying) is often determined by the amount of data available for estimating the GMM parameters. It is also important to note that because the component Gaussian are acting together to model the overall feature density, full covariance matrices are not necessary even if the features are not statistically independent. The linear combination of diagonal covariance basis Gaussians is capable of modeling the correlations between feature vector elements. The effect of using a set of M full covariance matrix Gaussians can be equally obtained by using a larger set of diagonal covariance Gaussians.

Gray Level and Color Quantization

Color image quantization is used to reduce the number of colors of a digital image with aminimal visual distortion. Color quantization can also be defined as a lossy image compres-sion operation. Until lately, quantization was used to reproduce 24 bit images on graphicshardware

with a limited number of simultaneous colors (e.g frame buffer displays with 4 or8 bit colormaps). Even though 24 bit graphics hardware is becoming more common, colorquantization still maintains its practical value. It lessens space requirements for storage of image data and reduces transmission band width requirements in multimedia applications.

Given a color image I, let us denote by CI the set of its colors and by M the cardinality of CI. The quantization of Iinto K colors, with K < M (and usually K << M) consists in selecting a set of K representative colors and replacing the color of each pixel of the original image by a suitable representative color. Since first applications of quantization were used to display full color images on low-cost color output devices, quantization algorithms had from the beginning to face to two con-straints. On one hand, the quantized image must be computed at the time the image is displayed. This makes computational efficiency of critical importance. On the other hand, the visual distortion between the original image and the reproduced one has to be as small as possible. The trade off between computational times and quantized image quality is appli-cation dependent and many quantization methods have been designed according to various constraints on this trade off. One straightforward way to obtain quantized images with low computational times con-sists to use a preselected set of representative colors.

Digital Image Representation: Digital image is a finite collection of discrete samples (pixels) of any observable object. The pixels represent a two- or higher dimensional “view” of the object, each pixel having its own discrete value in a finite range. The pixel values may represent the amount of visible light, infra-red light, absorption of x-rays, electrons, or any other measurable value such as ultrasound wave impulses. The image does not need to have any visual sense; it is sufficient that the samples form a two-dimensional spatial structure that may be illustrated as an image. The images may be obtained by a digital camera, scanner, electron microscope, ultrasound stethoscope, or any other optical or non-optical sensor. Examples of digital image are Digital photographs, Satellite images, Radiological images (x-rays, mammograms), Binary images, Fax images, Engineering drawings, Computer graphics, CAD drawings, and vector graphics in general are not considered in this course even though their reproduction is a possible source of an image. In fact, one goal of intermediate level image processing may be to reconstruct a model (Eg: Vector Representation) for a given digital image.

Relationship between Pixels:

We consider several important relationships between pixels in a digital image. Neighbors of a Pixel: A pixel p at coordinates (x,y) has four horizontal and vertical neighbors whose coordinates are givenby: (x+1,y), (x-1, y), (x, y+1), (x,y-1)

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

This set of pixels, called the 4-neighbors or p, is denoted by N4(p). Each pixel is one unit distance from (x,y) and some of the neighbors of p lie outside the digital image if (x,y) is on the

border of the image. The four diagonal neighbors of p have coordinates and are denoted by ND(p). (x+1, y+1), (x+1, y-1), (x-1, y+1), (x-1, y-1).

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

These points, together with the 4-neighbors, are called the 8-neighbors of p, denoted by N8(p).

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

As before, some of the points in ND(p) and N8(p) fall outside the image if (x,y) is on the border of the image.

Adjacency and Connectivity: Let v be the set of gray –level values used to define adjacency, in a binary image, V={1}. In a gray-scale image, the idea is the same, but V typically contains more elements, for example, V= {180, 181, 182, …,200}. If the possible intensity values 0 – 255, V set can be any subset of these 256 values. if we are reference to adjacency of pixel with value. Three types of Adjacency:

4- Adjacency – two pixel P and Q with value from V are 4 –adjacency if A is in the setN4(P)

8- Adjacency – two pixel P and Q with value from V are 8 –adjacency if A is in the set N8(P).

M-adjacency –two pixel P and Q with value from V are m – adjacency if (i) Q is in N4(p) or (ii) Q is in ND(q) and the (iii) Set N4(p) ∩ N4(q) has no pixel whose values are fromV. Mixed adjacency is a modification of 8-adjacency. It is introduced to eliminate the ambiguities that often arise when 8-adjacency isused. For example:

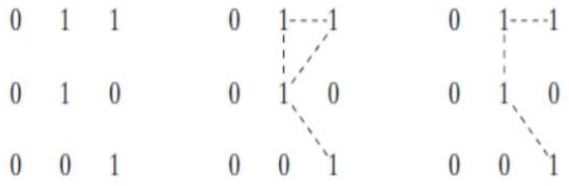


Fig: 15.(a) Arrangement of pixels (b) pixels that are 8-adjacent (shown dashed) to the center pixel (c) m-adjacency

Types of Adjacency: In this example, we can note that to connect between two pixels (finding a path between twopixels): In 8-adjacency way, you can find multiple paths between twopixels

While, in m-adjacency, you can find only one path between two pixels So, m-adjacency has eliminated the multiple path connection that has been generated by the8-adjacency. Two subsets S1 and S2 are adjacent, if some pixel in S1 is adjacent to some pixel in S2. Adjacent means, either 4-, 8- orm-adjacency.

Digital Path: A digital path (or curve) from pixel p with coordinate (x,y) to pixel q with coordinate (s,t) is a sequence of distinct pixels with coordinates (x0,y0), (x1,y1), …, (xn, yn) where (x0,y0) = (x,y) and (xn, yn) = (s,t) and pixels (xi, yi) and (xi-1, yi-1) are adjacent for $1 \leq i \leq n$, n is the length of thepath. If (x0,y0) = (xn, yn), the path isclosed. We can specify 4, 8or m-paths depending on the type of adjacency specified.

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

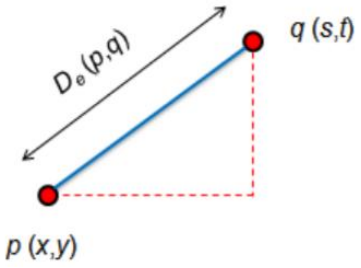
Region: Let R be a subset of pixels in an image, we call R a region of the image if R is a connected set. Boundary: The boundary (also called border or contour) of a region R is the set of pixels in the region that have one or more neighbors that are not in R. If R happens to be an entire image, then its boundary is defined as the set of pixels in the first and last rows and columns in the image. This extra definition is required because an image has no neighbors beyond its borders. Normally, when we refer to a region, we are referring to subset of an image, and any pixels in the boundary of the region that happen to coincide with the border of the image are included implicitly as part of the region boundary.

Distance Measures:

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

D [p.q] ≥ 0 {D[p.q] = 0iff p=q}

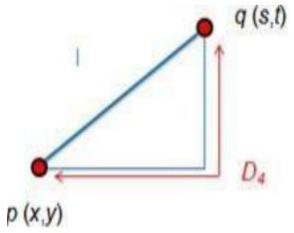
D [p.q] = D [p.q] and D [p.q] ≥ 0 {D[p.q]+D(q,z) The Euclidean Distance between p and q is defined as: De(p,q) = [(x – s)2 + (y - t)2] 1/2 Pixels having a distance less than or equal to some value r from (x,y) are the points contained in a disk of radius „ r „centered at (x,y).



The D_4distance (also called city-block distance) between p and q is defined as:

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

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



Camera Geometry

Standard perspective projection

If the world coordinates of a point are (X,Y,Z) and the image coordinates are (x,y), then

x = fX/Z and y = fY/Z

The model is non-linear.

In terms of projective coordinates:

$$\lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \in \mathcal{P}^2 \text{ and } \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \in \mathcal{P}^3$$

are homogeneous coordinates. The model is linear in projective geometry.

Basics of Projective Geometry

Affine and Euclidean geometries

Given a coordinate system, n-dimensional real affine space is the set of all points parameterized by x = (x1,... ,xn) t ∈ R n . • An affine transformation is expressed as

$$\mathbf{x}' = \mathbf{A}\mathbf{x} + \mathbf{b}$$

where A is a n × n (usually) non-singular matrix and b is a n × 1 vector representing a translation.

• In the special case of when A is a rotation (i.e., $AA^t = A^tA = I$, the transformation is Euclidean.

• Transformation of one point (or one axis) completely determines an Euclidean transformation, an affine transformation in n dimensions is completely determined by a mapping of n + 1 points (3 points for a plane).

• It is easy to verify that an affine transformation preserves parallelism and ratios of lengths along parallel directions. In fact, coordinates in an affine geometry are defined in terms of these fundamental invariants. An Euclidean transformation, in addition to the above, also preserves lengths and angles.

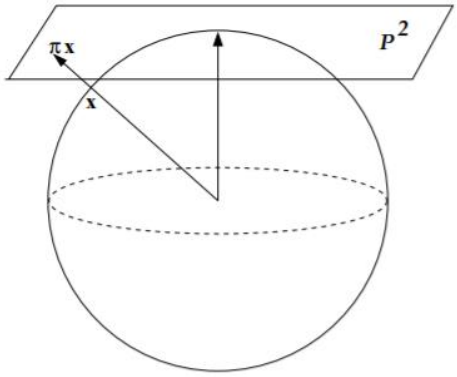
Projective geometry

• The projective plane P^2 is the set of all pairs {x, −x} of antipodal points in S^2 .

• Two alternative definitions of P^2 , equivalent to the preceding one are 1. The set of all lines through the origin in R^3 .

2. The set of all equivalence classes of ordered triples (x1, x2, x3) of numbers (i.e., vectors in R^3) not all zero, where two vectors are equivalent if they are proportional.

• The space P^2 can be thought of as the infinite plane tangent to the space S 2 and passing through the point $(0, 0, 1)^t$.



- Let $\pi : \mathcal{S}^2 \to \mathcal{P}^2$ be the mapping that sends $\mathbf{x}$ to $\{\mathbf{x}, -\mathbf{x}\}$. The π is a two-to-one map of $\mathcal{S}^2$ onto $\mathcal{P}^2$.
- A line of $\mathcal{P}^2$ is a set of the form $\pi\mathbf{l}$, where $\mathbf{l}$ is a line of $\mathcal{S}^2$. Clearly, $\pi\mathbf{x}$ lies on $\pi\mathbf{l}$ if and only if $\xi^t\mathbf{x} = 0$.
- **Homogeneous coordinates:** In general, points of real n-dimensional **projective space**, $\mathcal{P}^n$,

  are represented by $n+1$ component column vectors $(x_1, \ldots, x_n, x_{n+1}) \in \mathbb{R}^{n+1}$ such that at least one x_i is non-zero and $(x_1, \ldots, x_n, x_{n+1})$ and $(\lambda x_1, \ldots, \lambda x_n, \lambda x_{n+1})$ represent the same point of $\mathcal{P}^n$ for all $\lambda \neq 0$.
- $(x_1, \ldots, x_n, x_{n+1})$ is the homogeneous representation of a projective point.

2D Transformation

Transformation means changing some graphics into something else by applying rules. We can have various types of transformations such as translation, scaling up or down, rotation, shearing, etc. When a transformation takes place on a 2D plane, it is called 2D transformation.

Transformations play an important role in computer graphics to reposition the graphics on the screen and change their size or orientation.

Homogenous Coordinates

To perform a sequence of transformation such as translation followed by rotation and scaling, we need to follow a sequential process −

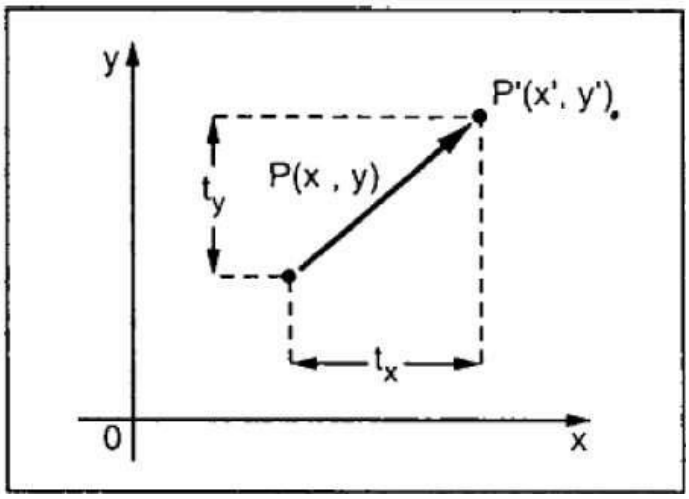
- Translate the coordinates,
- Rotate the translated coordinates, and then
- Scale the rotated coordinates to complete the composite transformation.

To shorten this process, we have to use 3×3 transformation matrix instead of 2×2 transformation matrix. To convert a 2×2 matrix to 3×3 matrix, we have to add an extra dummy coordinate W.

In this way, we can represent the point by 3 numbers instead of 2 numbers, which is called **Homogenous Coordinate** system. In this system, we can represent all the transformation equations in matrix multiplication. Any Cartesian point PX,YX,Y can be converted to homogenous coordinates by P' (X_h, Y_h, h).

Translation

A translation moves an object to a different position on the screen. You can translate a point in 2D by adding translation coordinate (t_x, t_y) to the original coordinate X,YX,Y to get the new coordinate X′,Y′X′,Y′.



From the above figure, you can write that −

X' = X + t_x

Y' = Y + t_y

The pair (t_x, t_y) is called the translation vector or shift vector. The above equations can also be represented using the column vectors.

P= X][Y] p' = [X′][Y′] T = [tx][ty]

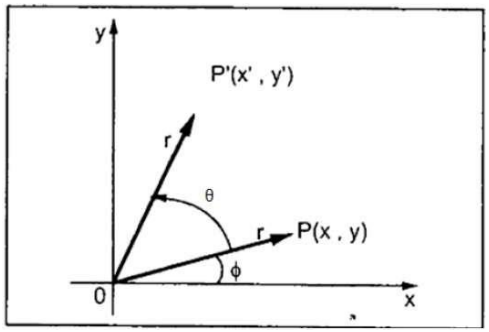
We can write it as −

P' = P + T

Rotation

In rotation, we rotate the object at particular angle θ thetatheta from its origin. From the following figure, we can see that the point PX,YX,Y is located at angle φ from the horizontal X coordinate with distance r from the origin.

Let us suppose you want to rotate it at the angle θ. After rotating it to a new location, you will get a new point P' X′,Y′X′,Y′.



Using standard trigonometric the original coordinate of point PX,YX,Y can be represented as −

X=rcosϕ.....(1)

Y=rsinϕ.....(2)

Same way we can represent the point P' X′,Y′X′,Y′ as −

x′=rcos(ϕ+θ)=rcosϕcosθ−rsinϕsinθ.....(3)

y′=rsin(ϕ+θ)=rcosϕsinθ+rsinϕcosθ.....(4)

Substituting equation 1 & 2 in 3 & 4 respectively, we will get

x′=xcosθ−ysinθ

y′=xsinθ+ycosθ

Representing the above equation in matrix form,

$$[X'Y'] = [XY]\begin{bmatrix} cos\theta & sin\theta \\ -sin\theta & cos\theta \end{bmatrix} OR$$

P' = P . R

Where R is the rotation matrix

$$R = \begin{bmatrix} cos\theta & sin\theta \\ -sin\theta & cos\theta \end{bmatrix}$$

The rotation angle can be positive and negative.

For positive rotation angle, we can use the above rotation matrix. However, for negative angle rotation, the matrix will change as shown below −

$$R = \begin{bmatrix} cos(-\theta) & sin(-\theta) \\ -sin(-\theta) & cos(-\theta) \end{bmatrix}$$

$$= \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix} (\because cos(-\theta) = cos\theta \; and \; sin(-\theta) = -sin\theta)$$

Scaling

To change the size of an object, scaling transformation is used. In the scaling process, you either expand or compress the dimensions of the object. Scaling can be achieved by multiplying the original coordinates of the object with the scaling factor to get the desired result.

Let us assume that the original coordinates are X,YX,Y, the scaling factors are (S_X, S_Y), and the produced coordinates are X′,Y′X′,Y′. This can be mathematically represented as shown below −

X' = X . S_X and Y' = Y . S_Y

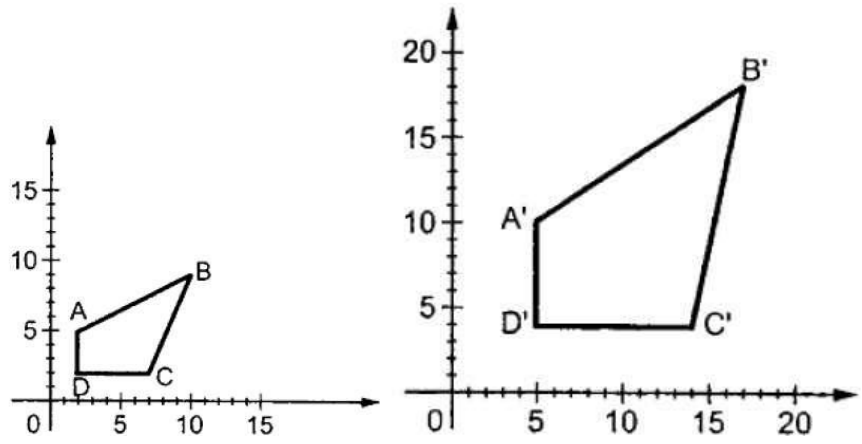
The scaling factor S_X, S_Y scales the object in X and Y direction respectively. The above equations can also be represented in matrix form as below −

$$\begin{pmatrix} X' \\ Y' \end{pmatrix} = \begin{pmatrix} X \\ Y \end{pmatrix} \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix}$$

OR

P' = P . S

Where S is the scaling matrix. The scaling process is shown in the following figure.

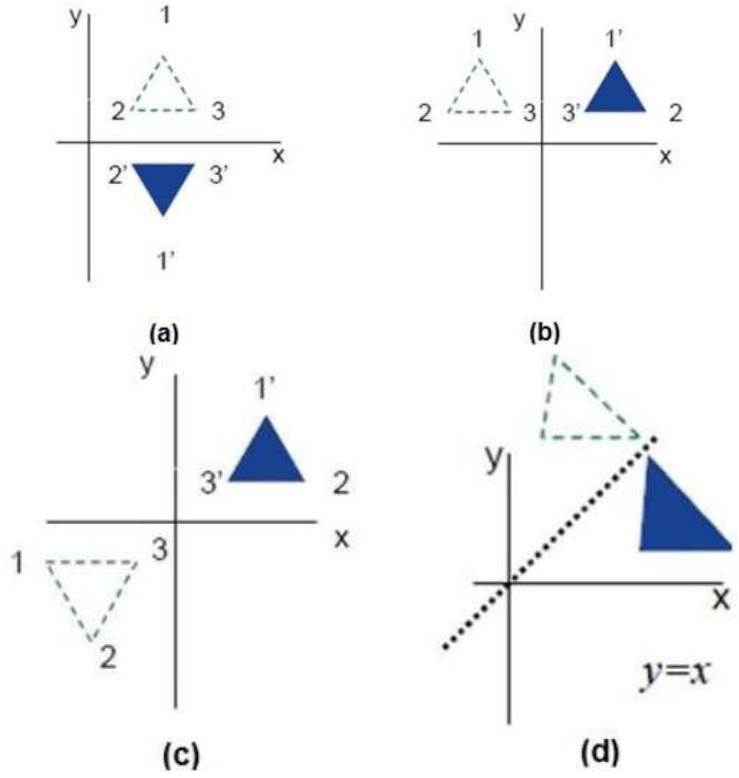


If we provide values less than 1 to the scaling factor S, then we can reduce the size of the object. If we provide values greater than 1, then we can increase the size of the object.

Reflection

Reflection is the mirror image of original object. In other words, we can say that it is a rotation operation with 180°. In reflection transformation, the size of the object does not change.

The following figures show reflections with respect to X and Y axes, and about the origin respectively.

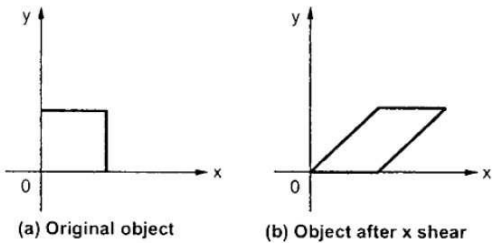


Shear

A transformation that slants the shape of an object is called the shear transformation. There are two shear transformations **X-Shear** and **Y-Shear**. One shifts X coordinates values and other shifts Y coordinate values. However; in both the cases only one coordinate changes its coordinates and other preserves its values. Shearing is also termed as **Skewing**.

X-Shear

The X-Shear preserves the Y coordinate and changes are made to X coordinates, which causes the vertical lines to tilt right or left as shown in below figure.



The transformation matrix for X-Shear can be represented as −

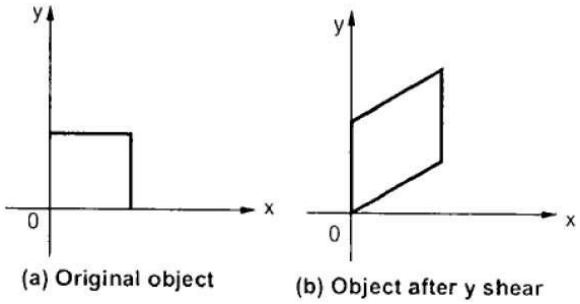
$$X_{sh} = \begin{bmatrix} 1 & shx & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$Y' = Y + Sh_y \cdot X$$

$$X' = X$$

Y-Shear

The Y-Shear preserves the X coordinates and changes the Y coordinates which causes the horizontal lines to transform into lines which slopes up or down as shown in the following figure.



The Y-Shear can be represented in matrix from as −

$$Y_{sh} \begin{bmatrix} 1 & 0 & 0 \\ shy & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$X' = X + Sh_x \cdot Y$$

$$Y' = Y$$

Composite Transformation

If a transformation of the plane T1 is followed by a second plane transformation T2, then the result itself may be represented by a single transformation T which is the composition of T1 and T2 taken in that order. This is written as T = T1·T2.

Composite transformation can be achieved by concatenation of transformation matrices to obtain a combined transformation matrix.

A combined matrix −

[T][X] = [X] [T1] [T2] [T3] [T4] …. [Tn]

Where [Ti] is any combination of

- Translation
- Scaling
- Shearing
- Rotation
- Reflection

The change in the order of transformation would lead to different results, as in general matrix multiplication is not cumulative, that is [A] . [B] ≠ [B] . [A] and the order of multiplication. The basic purpose of composing transformations is to gain efficiency by applying a single composed transformation to a point, rather than applying a series of transformation, one after another.

For example, to rotate an object about an arbitrary point (X_p, Y_p), we have to carry out three steps −

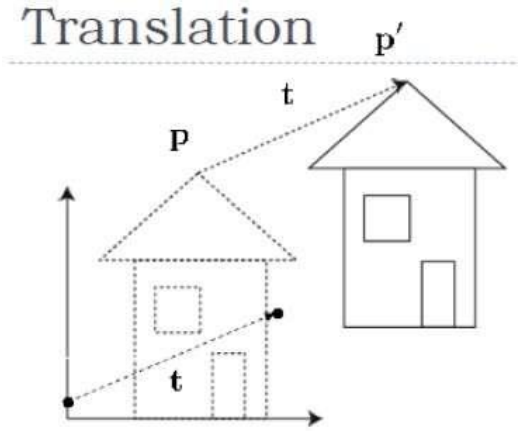
- Translate point (X_p, Y_p) to the origin.
- Rotate it about the origin.
- Finally, translate the center of rotation back where it belonged.

3D TRANSFORM

Translation

In 3D translation, we transfer the Z coordinate along with the X and Y coordinates. The process for translation in 3D is similar to 2D translation. A translation moves an object into a different position on the screen.

The following figure shows the effect of translation −



A point can be translated in 3D by adding translation coordinate (tx,ty,tz)(tx,ty,tz) to the original coordinate X,Y,ZX,Y,Z to get the new coordinate X′,Y′,Z′X′,Y′,Z′.

$$T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ t_x & t_y & t_z & 1 \end{bmatrix}$$

P' = P·T

$$[X'\ Y'\ Z'\ 1] = [X\ Y\ Z\ 1] \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ t_x & t_y & t_z & 1 \end{bmatrix}$$

$$= [X + t_x \quad Y + t_y \quad Z + t_z \quad 1]$$

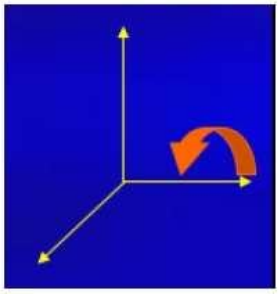
Rotation

3D rotation is not same as 2D rotation. In 3D rotation, we have to specify the angle of rotation along with the axis of rotation. We can perform 3D rotation about X, Y, and Z axes. They are represented in the matrix form as below −

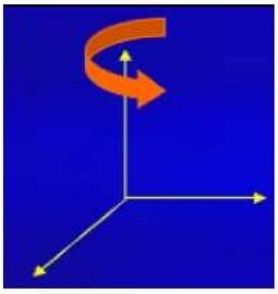
$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos\theta & -sin\theta & 0 \\ 0 & sin\theta & cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad R_y(\theta) = \begin{bmatrix} cos\theta & 0 & sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -sin\theta & 0 & cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad R_z(\theta)$$

$$= \begin{bmatrix} cos\theta & -sin\theta & 0 & 0 \\ sin\theta & cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

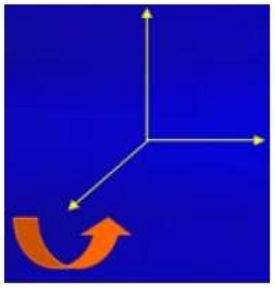
The following figure explains the rotation about various axes −



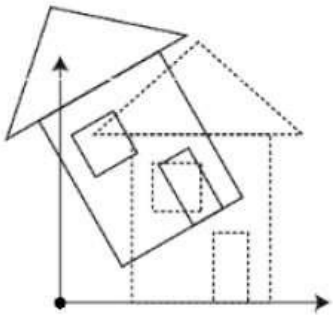
Rotation about x-axis



Rotation about y-axis

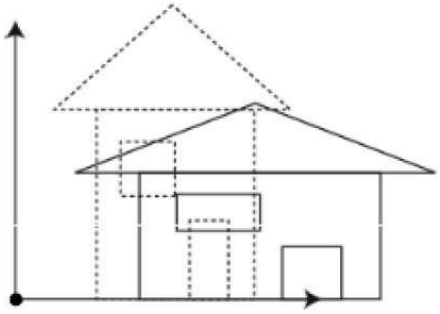


Rotation about z-axis



Scaling

You can change the size of an object using scaling transformation. In the scaling process, you either expand or compress the dimensions of the object. Scaling can be achieved by multiplying the original coordinates of the object with the scaling factor to get the desired result. The following figure shows the effect of 3D scaling −



In 3D scaling operation, three coordinates are used. Let us assume that the original coordinates are X,Y,ZX,Y,Z, scaling factors are (SX,SY,Sz)(SX,SY,Sz) respectively, and the produced coordinates are X′,Y′,Z′X′,Y′,Z′. This can be mathematically represented as shown below −

$$S = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

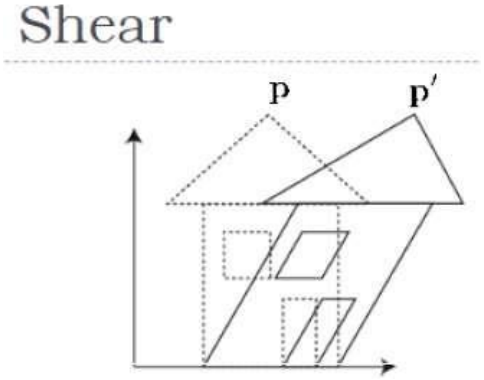
$$P' = P \cdot S$$

$$[X' \quad Y' \quad Z' \quad 1] = [X \quad Y \quad Z \quad 1] \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= [X.S_x \quad Y.S_y \quad Z.S_z \quad 1]$$

Shear

A transformation that slants the shape of an object is called the shear transformation. Like in 2D shear, we can shear an object along the X-axis, Y-axis, or Z-axis in 3D.



As shown in the above figure, there is a coordinate P. You can shear it to get a new coordinate P', which can be represented in 3D matrix form as below −

$$Sh = \begin{bmatrix} 1 & sh_x^y & sh_x^z & 0 \\ sh_y^x & 1 & sh_y^z & 0 \\ sh_z^x & sh_z^y & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

P' = P · Sh

$$X' = X + Sh_x^y Y + Sh_x^z Z$$

$$Y' = Sh_y^x X + Y + sh_y^z Z$$

$$Z' = Sh_z^x X + Sh_z^y Y + Z$$

Transformation Matrices

Transformation matrix is a basic tool for transformation. A matrix with n x m dimensions is multiplied with the coordinate of objects. Usually 3 x 3 or 4 x 4 matrices are used for transformation. For example, consider the following matrix for various operation.

$T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ t_x & t_y & t_z & 1 \end{bmatrix}$	$S = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	$Sh = \begin{bmatrix} 1 & sh_x^y & sh_x^z & 0 \\ sh_y^x & 1 & sh_y^z & 0 \\ sh_z^x & sh_z^y & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
Translation Matrix	**Scaling Matrix**	**Shear Matrix**

$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos\theta & -sin\theta & 0 \\ 0 & sin\theta & cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	$R_y(\theta) = \begin{bmatrix} cos\theta & 0 & sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -sin\theta & 0 & cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	$R_z(\theta) = \begin{bmatrix} cos\theta & -sin\theta & 0 & 0 \\ sin\theta & cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
Rotation Matrix		

TEXT / REFERENCE BOOKS

1. Donald D Hearn, M. Pauline Baker, Computer Graphics C version, Pearson Education.
2. Dave Shreiner, Mason Woo, Jackie Neider, Tom Davis, OpenGL Programming Guide: The Official Guide to Learning OpenGL, (2013).

3. James D. Foley, Andries van Dam, Steven K. Feiner, John F. Hughes, Computer Graphics: Principles & Practice in C, Addison Wesley Longman.
4. Zhigang Xiang, Roy A Plastock, Computer Graphics, Schaums Outline, TMH.

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

SCHOOL OF ELECTRICAL AND ELECTRONICS

DEPARTMENT OF ELECTRONICS AND COMMMUNICATIONENGINEERING

UNIT - II

SCSA3007- INTRODUCTION TO VISUAL COMPUTING

UNIT 2 DIGITAL IMAGE PROCESSING

Digital Image Filtering-Image Transforms-Image Enhancement and Restoration, Wiener Filters, Nonlinear Image Processing (Median filtering)-Nonlinear Diffusion, Gauss-Laplace Pyramid, Wavelets-Scale Space, (Image and Video Compression Image Segmentation-Optical Flow-Stereo Vision-Template Matching, Point Matching

Digital Image Filtering

In image processing filters are mainly used to suppress either the high frequencies in the image, *i.e.* smoothing the image, or the low frequencies, *i.e.* enhancing or detecting edges in the image.

An image can be filtered either in the frequency or in the spatial domain.

The first involves transforming the image into the frequency domain, multiplying it with the frequency filter function and re-transforming the result into the spatial domain. The filter function is shaped so as to attenuate some frequencies and enhance others. For example, a simple lowpass function is *1* for frequencies smaller than the *cut-off frequency* and *0* for all others.

The corresponding process in the spatial domain is to convolve the input image $f(i,j)$ with the filter function $h(i,j)$. This can be written as

$$g(i,j) = h(i,j) \odot f(i,j)$$

The mathematical operation is identical to the multiplication in the frequency space, but the results of the digital implementations vary, since we have to approximate the filter function with a discrete and finite kernel.

The discrete convolution can be defined as a *`shift and multiply'* operation, where we shift the kernel over the image and multiply its value with the corresponding pixel values of the image. For a square kernel with size $M \times M$, we can calculate the output image with the following formula:

$$g(i,j) = \sum_{m=-\frac{M}{2}}^{\frac{M}{2}} \sum_{n-=\frac{M}{2}}^{\frac{M}{2}} h(m,n)\ f(i-m, j-n)$$

Various standard kernels exist for specific applications, where the size and the form of the kernel determine the characteristics of the operation. The most important of them are discussed in this chapter. The kernels for two examples, the mean and the Laplacian operator, can be seen in Figure 1.

$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$

Mean

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

Laplacian

Figure 1 Convolution kernel for a mean filter and one form of the discrete Laplacian.

In contrast to the frequency domain, it is possible to implement non-linear filters in the spatial domain. In this case, the summations in the convolution function are replaced with some kind of non-linear operator:

$$g(i,j) = O_{m,n}[h(m,n)\ f(i-m, j-n)]$$

For most non-linear filters the elements of $h(i,j)$ are all 1. A commonly used non-linear operator is the median, which returns the `middle' of the input values.

Image Transforms

Image transforms are the bases of image processing and analysis. Image transforms are used in image enhancement, restoration, reconstruction, encoding and description.

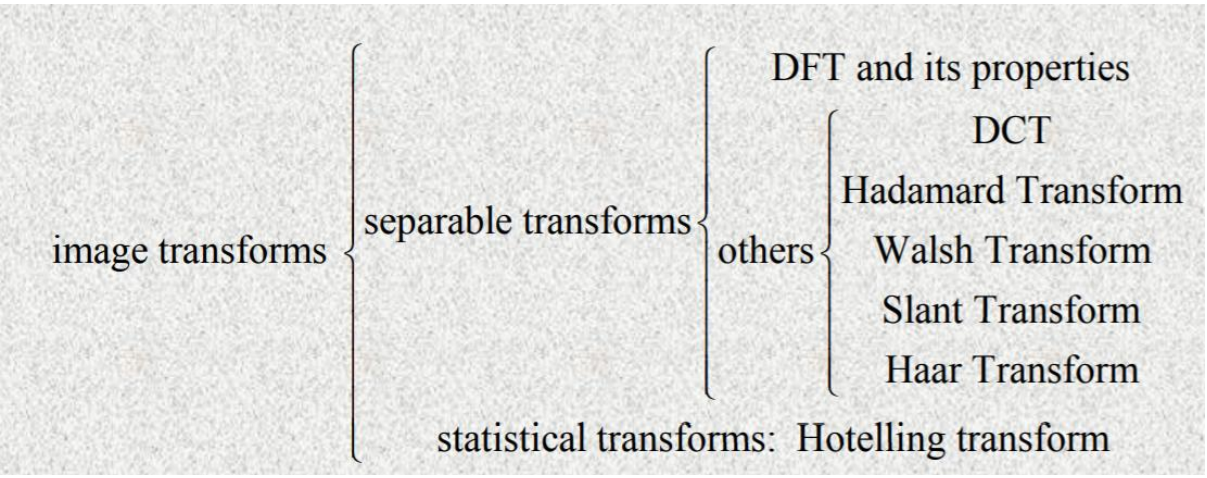


Figure 2.classification of image transform

2D-DFT

The Two-Dimensional Discrete Fourier Transform and its Inverse

2D Discrete Fourier Transform

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

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

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

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

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

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

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

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

The above is equivalent to:

Properties

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

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

WALSH TRANSFORM:

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

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

The transform kernel values are obtained from:

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

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

1-D Inverse Walsh Transform

The above is again equivalent to

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

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

2-D Walsh Transform

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

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

•The above is equivalent to:

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

Inverse Walsh Transform

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

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

•The above is equivalent to:

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

HADAMARD TRANSFORM:

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

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

The above is equivalent to:

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

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

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

The above is equivalent to:

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

DISCRETE COSINE TRANSFORM (DCT) :

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

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

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

and the corresponding *inverse* 1D DCT transform is simple $F^{-1}(u)$, i.e.:where The general equation for a 2D (*N* by *M* image) DCT is defined by the following equation:

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

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

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

where

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

The basic operation of the DCT is as follows:

- The input image is N by M;
- f(i,j) is the intensity of the pixel in row i and column j;
- F(u,v) is the DCT coefficient in row k1 and column k2 of the DCT matrix.
- For most images, much of the signal energy lies at low frequencies; these appear in the upper left corner of the DCT.
- Compression is achieved since the lower right values represent higher frequencies, andare often small - small enough to be neglected with little visible distortion.
- The DCT input is an 8 by 8 array of integers. This array contains each pixel's gray scale level;
- 8 bit pixels have levels from 0 to 255.

DISCRETE WAVELET TRANSFORM (DWT):

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

Haar Wavelet Transform

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

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

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

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

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

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

Two-Dimensional Wavelets

The two-dimensional wavelet transform is separable, which means we can apply a one-dimensional wavelet transform to an image. We apply one-dimensional DWT to all rows and then one-dimensional DWTs to all columns of the result. This is called the standard decomposition and it is illustrated in Figure 2.8.

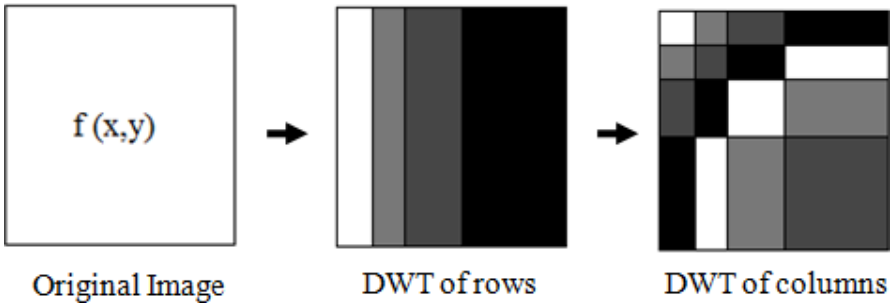


Figure The standard decomposition of the two-dimensional DWT.

We can also apply a wavelet transform differently. Suppose we apply a wavelet transformto an image by rows, then by columns, but using our transform at one scale only. This technique will produce a result in four quarters: the top left will be a half-sized version of the image and theother quarter's high-pass filtered images. These quarters will contain horizontal, vertical, and diagonal edges of the image. We then apply a one-scale DWT to the top-left quarter, creating smaller images, and so on. This is called the nonstandard decomposition, and is illustrated inFigure 3.

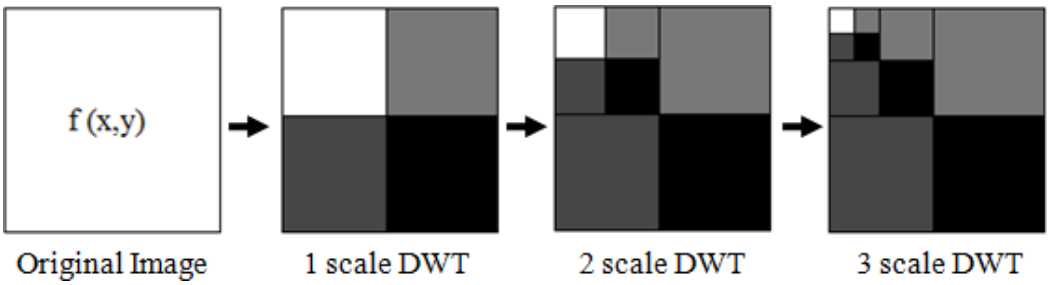


Figure 3 The nonstandard decomposition of the two-dimensional DWT.

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

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

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

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

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

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

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

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

1. The low-pass / low-pass image (LL), the result of step 2,
2. The low-pass / high-pass image (LH), the result of step 3,
3. The high-pass / low-pass image (HL), the result of step 5, and
4. The high-pass / high-pass image (HH), the result of step 6

These images can be placed into a single image grid as shown in the Figure 4.

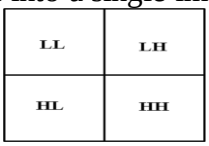


Figure 4 the one-scale wavelet transforms in terms of filters.

Figure 5 describes the basic dwt decomposition steps for an image in a block diagram form. The two-dimensional DWT leads to a decomposition of image into four components CA, CH, CV and CD, where CA are approximation and CH, CV, CD are details in three orientations (horizontal, vertical, and diagonal), these are same as LL, LH, HL, and HH. In these coefficients the watermark can be embedded.

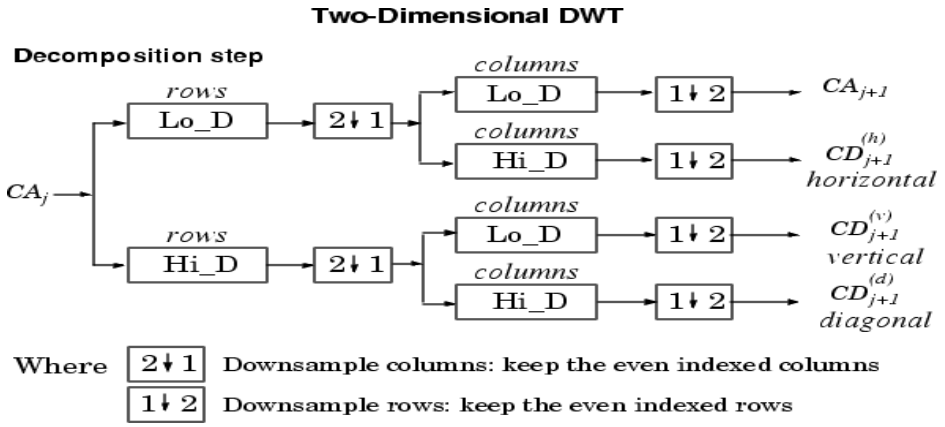


Figure 5. DWT decomposition steps for an image.

Figure 6.Original image and DWT decomposed image.

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

IMAGE ENHANCEMENT

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

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

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

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

utilizes only the pixels in the area of the image spanned by the neighborhood.

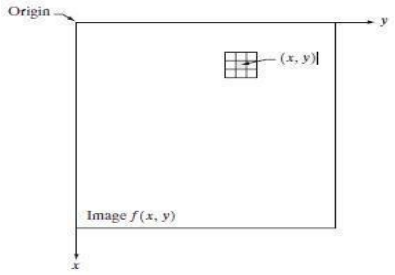


Fig.: 7 .3x3 neighborhood about a point (x,y) in an image.

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

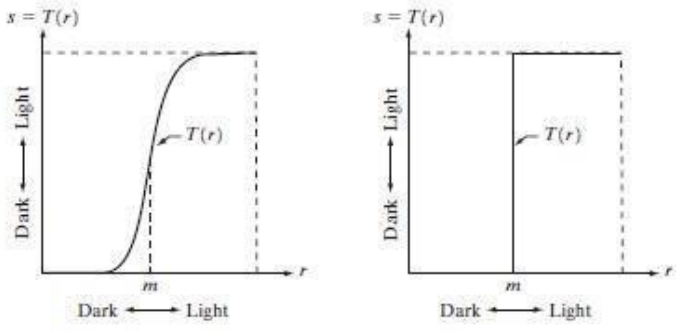
$$s = T (r)$$

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

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

In the limiting case shown in Fig. 2.2(b), T(r) produces a two-level (binary) image. A mapping of this form is called a thresholding function.

One of the principal approaches in this formulation is based on the use of so-called masks (also referred to as filters, kernels, templates, or windows). Basically, a mask is a small (say, 3*3) 2-D array, such as the one shown in Fig. 2.1, in which the values of the mask coefficients determine the nature of the process, such as image sharpening. Enhancement techniques basedon this type



of approach often are referred to as mask processing or filtering.

Fig. 8. Gray level transformation functions for contrast enhancement.

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

BASIC GRAY LEVEL TRANSFORMATIONS:

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

LINEAR TRANSFORMATION:

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

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

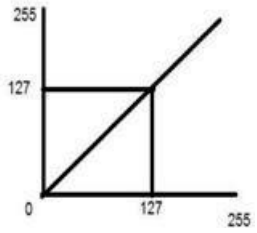


Fig. 9Linear transformation between input and output.

NEGATIVE TRANSFORMATION:

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

IMAGENEGATIVE: The imagenegativewithgraylevelvalue inthe rangeof[0, L-1] is obtained by negative transformation given by S = T(r) or

$$S = L - 1 - r$$

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

L is the largest gray level consists in the image

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

The overall graph of these transitions has been shown below.

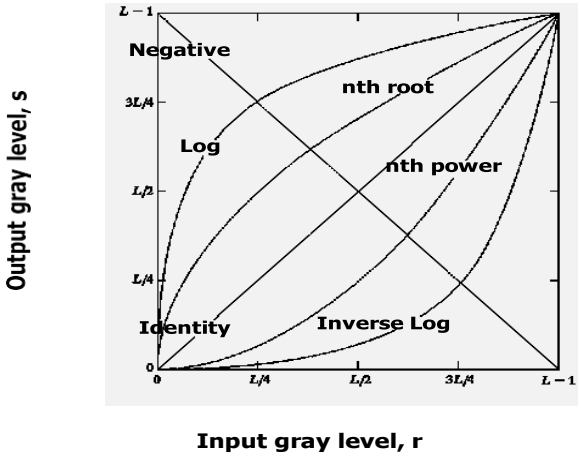


Fig. 10.Some basic gray-level transformation functions used for image enhancement.

In this case the following transition has been done.

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

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

$$S = 255 - r$$

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

It has been shown in the graph below.

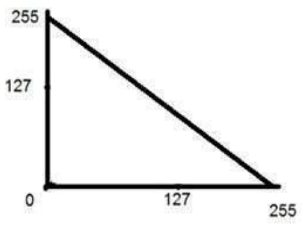


Fig. 11.Negative transformations.

LOGARITHMIC TRANSFORMATIONS:

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

LOG TRANSFORMATIONS:

The log transformations can be defined by this formula

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

Where s and r are the pixel values of the output and the input image and c is a constant. The value 1 is added to each of the pixel value of the input image because if there is a pixel intensityof 0 in the image, then log (0) is equal to infinity. So 1 is added, to make the minimum value at least 1. During log transformation, the dark pixels in an image are expanded as compare to the higher pixel values. The higher pixel values are kind of compressed in log transformation. This result in following image enhancement.

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

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

- Here C is constant and $r \geq 0$.
- The shape of the curve shows that histransformation maps then arrow range of low graylevel values in the input image into a wider range of output image.
- The opposite is true for high level values of input image.

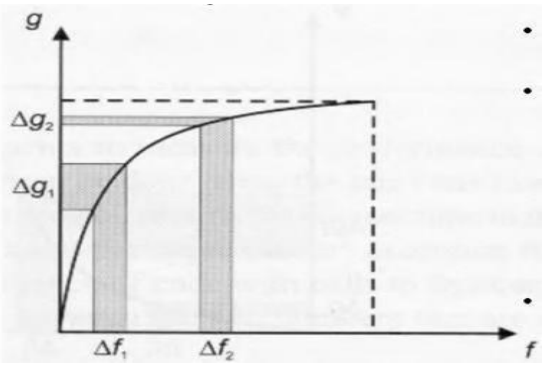


Fig. 12.log transformation curve input vs output

POWER – LAW TRANSFORMATIONS:

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

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

This symbol γ is called gamma, due to which this transformation is also known as gamma transformation. Variation in the value of γ varies the enhancement of the images. Different display devices / monitors have their own gamma correction, that's why they display their image at different intensity.

where c and g are positive constants. Sometimes Eq. (6) is written as $S = C (r + \varepsilon)^{\gamma}$ to account for an offset (that is, a measurable output when the input is zero). Plots of s versus r for various values of γ are shown in Fig. 2.10. As in the case of the log transformation, power-law curves with fractional values of γ map a narrow range of dark input values into a wider range of output values, with the opposite being true for higher values of input levels. Unlike the log function, however, we notice here a family of possible transformation curves obtained simply by varying γ.

In Fig 13.that curves generated with values of γ>1 have exactly The opposite effect as those generated with values of γ<1. Finally, the equation reduces to the identity transformationwhen c=γ=1.

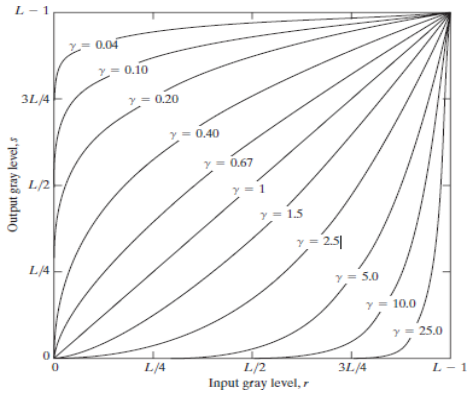


Fig. 13. Plot of the equation $S = cr^{\gamma}$ for various values of γ (c =1 in all cases).

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

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

Here C and γ are positive constants. Plot of S versus r for various values of γ isγ > 1 compresses dark values

Expands bright values. γ <1 (similar to log transformation)Expands dark values Compresses bright values.

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

CORRECTING GAMMA:

$s=cr^{\gamma}$

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

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

Piecewise-Linear Transformation Functions:

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

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

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

$$S = T(r)$$

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

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

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

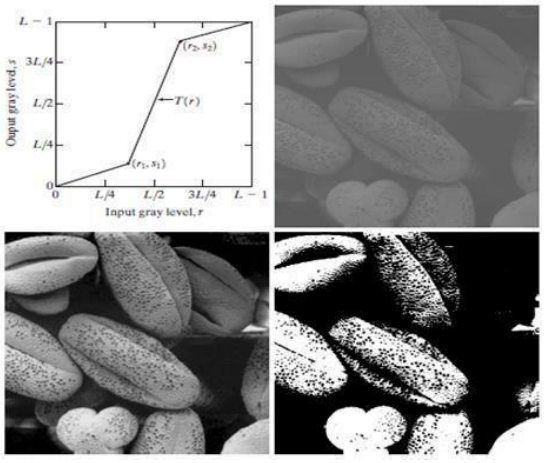


Fig. 15. Contrast stretching. (a) Form of transformation function. (b) A low-contrast stretching. (c) Result of high contrast stretching. (d) Result of thresholding (original image courtesy of Dr.Roger Heady, Research School of Biological Sciences, Australian National University Canberra

Australia.

Figure 15(b) shows an 8-bit image with low contrast. Fig. 15(c) shows the result of contrast stretching, obtained by setting (r1, s1)=(rmin, 0) and (r2, s2)=(r_{max},L-1) where r_{min} and r_{max} denote the minimum and maximum gray levels in the image, respectively.Thus, the transformation function stretched the levels linearly from their original range to the full range [0, L-1]. Finally, Fig. 15(d) shows the result of using the thresholding function defined previously, with r1=r2=m, the mean gray level in the image. The original image on which these results are based is a scanning electron microscope image of pollen, magnified approximately 700 times.

Gray-level slicing:

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

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

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

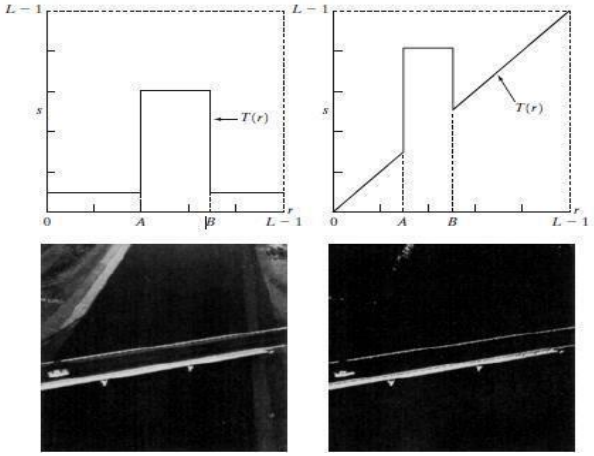


Fig. 16 (a)This transformation highlights range [A,B] of gray levels and reduces all others to a constant level (b) This transformation highlights range [A,B] but preserves all other levels. (c) An image . (d) Result of using the transformation in (a).

BIT-PLANE SLICING:

Instead of highlighting gray-level ranges, highlighting the contribution made to total image appearance by specific bits might be desired. Suppose that each pixel in an image is represented by 8 bits. Imagine that the image is composed of eight 1-bit planes, ranging from bit-plane 0 for the least significant bit to bit plane 7 for the most significant bit. In terms of 8-bit bytes, plane 0 contains all the lowest order bits in the bytes comprising the pixels in the image and plane 7 contains all the high-order bits.

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

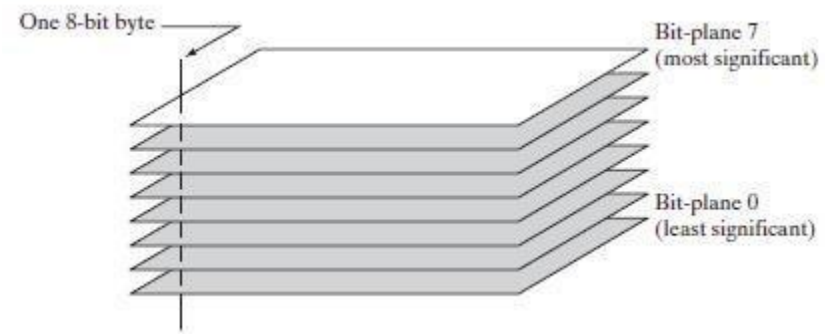


FIGURE Bit-plane representation of an 8-bit image.

Fig. 17 shows the various bit planes for an image .

In terms of bit-plane extraction for an 8-bit image, it is not difficult to show that the (binary) image for bit-plane 7 can be obtained by processing the input image with a thresholding gray-level transformation function that (1) maps all levels in the image between 0 and 127 to onelevel (for example, 0); and (2) maps all levels between 129 and 255 to another (for example, 255).The binary image for bit-plane 7 in Fig. 3.14 was obtained in just this manner. It is left as an exercise (Problem 3.3) to obtain the gray-level transformation functions that would yield the other bit planes.

Histogram Processing:

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

$$H(r_k)=n_k$$

where r_k is the k^{th} gray level and n_k is the number of pixels in the image having the level r_k..
A normalized histogram is given by the equation

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

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

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

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

In the dark image the components of the histogram are concentrated on the low (dark) side of the gray scale. In case of bright image the histogram components are baised towards the high side of the gray scale. The histogram of a low contrast image will be narrow and will be centered towards the middle of the gray scale.

The components of the histogram in the high contrast image cover a broad range of the gray scale. The net effect of this will be an image that shows a great deal of gray levels details and has high dynamic range.

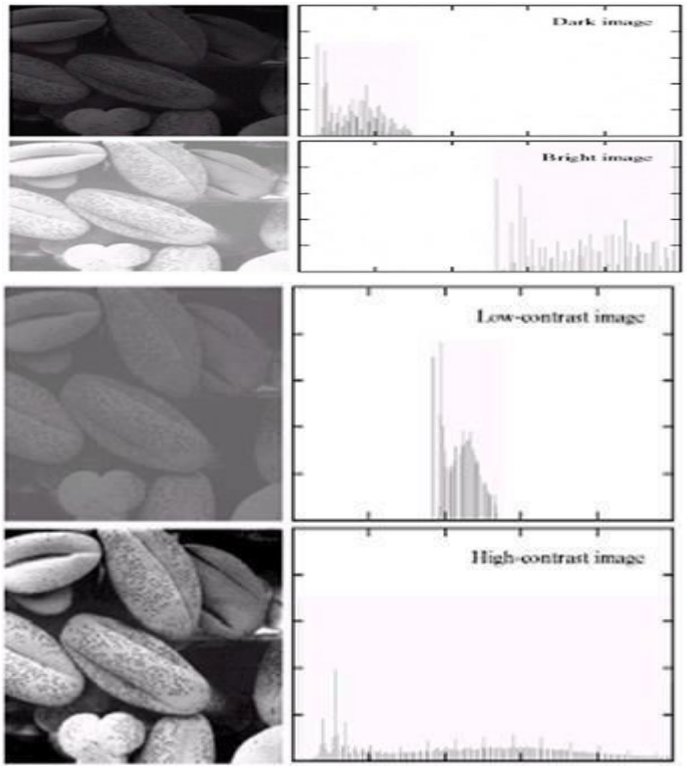


Fig 18. Histogram plots

Histogram Equalization:

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

skewed towards the lower end of the grey scale and all the image detail are compressed into the dark end of the histogram. If we could 'stretch out' the grey levels at the dark end to produce a more uniformly distributed histogram then the image would become much clearer.

Let there be a continuous function with r being gray levels of the image to be enhanced. The range of r is [0, 1] with r=0 representing black and r=1 representing white. The transformation function is of the form

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

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

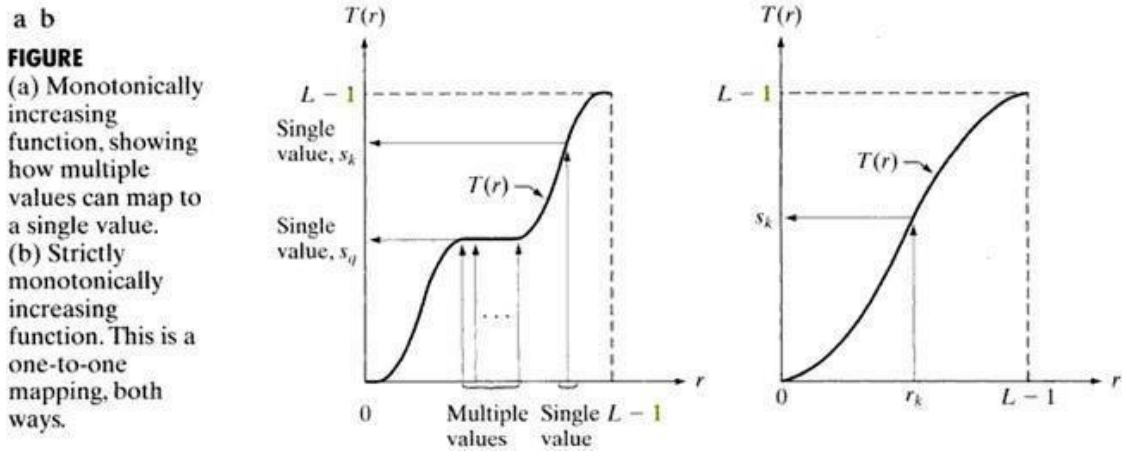


Fig 19. The transformation function is assumed to fulfill two condition T(r) is single valued and monotonically increasing in the internal 0<T(r)<1 for 0<r<1.The transformation function should be single valued so that the inverse transformations should exist. Monotonically increasing condition preserves the increasing order from black to white in the output image. The second conditions guarantee that the output gray levels will be in the same range as the input levels. The gray levels of the image may be viewed as random variables in the interval [0.1]. The most fundamental descriptor of a random variable is its probability density function (PDF) Pr(r) and Ps(s) denote the probability density functions of random variablesr and s respectively. Basic results from an elementary probability theory states that if Pr(r) and Tr are known and T-1(s) satisfies conditions (a), then the probability density function Ps(s) of the transformed variable is given by the formula

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

Thus the PDF of the transformed variable s is the determined by the gray levels PDF of the input image and by the chosen transformations function.

A transformation function of a particular importance in image processing

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

This is the cumulative distribution function of r.

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

IMAGE ENHANCEMENT IN FREQUENCY DOMAIN

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

IDEAL LOW-PASS FILTER:

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

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

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

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

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

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

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

VISUALIZATION: IDEAL LOW PASS FILTER:

Aa shown in fig.below

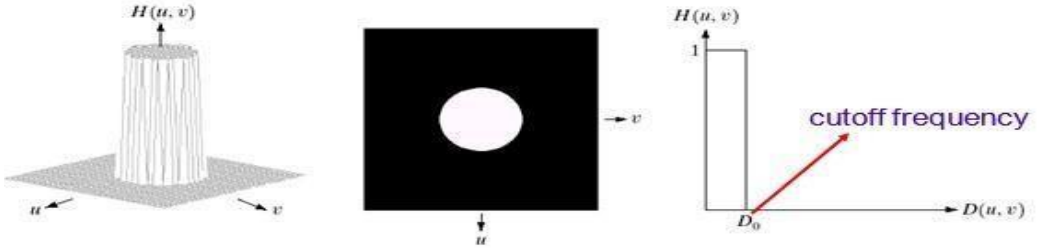


Fig: 20.ideal low pass filter 3-D view and 2-D view and line graph.

EFFECT OF DIFFERENT CUT OFF FREQUENCIES:

Fig.21below(a) Test pattern of size 688x688 pixels, and (b) its Fourier spectrum. The spectrum is double the image size due to padding but is shown in half size so that it fits in the page. The superimposed circles have radii equal to 10, 30, 60, 160 and 460 with respect to the full-size spectrum image. These radii enclose 87.0, 93.1, 95.7, 97.8 and 99.2% of the padded image power respectively.

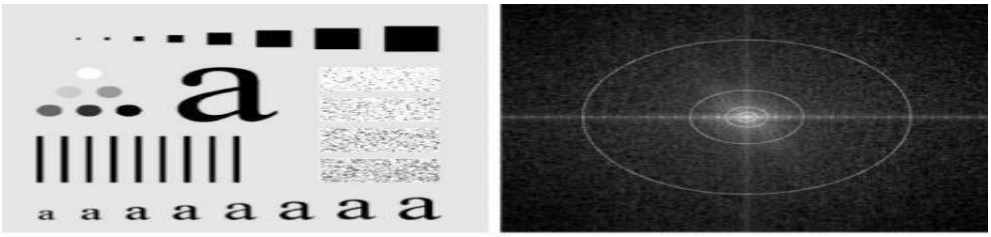


Fig: 21.(a) Test patter of size 688x688 pixels (b) its Fourier spectrum

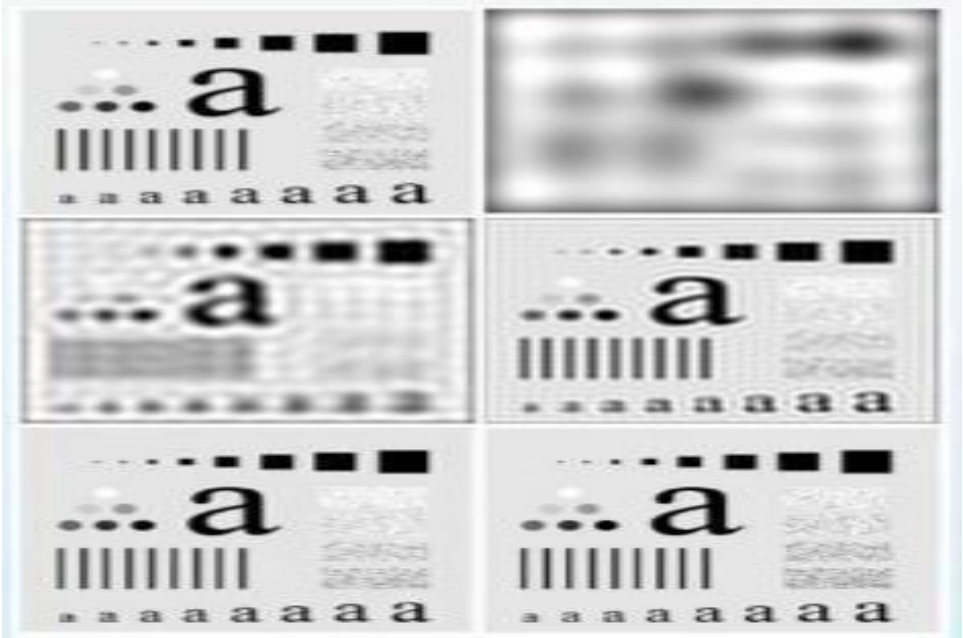


Fig:22. (a) original image, (b)-(f) Results of filtering using ILPFs with cutoff frequencies setat radii values 10, 30, 60, 160 and 460, as shown in fig.2.2.2(b). The power removed by these

filters was 13, 6.9, 4.3, 2.2 and 0.8% of the total, respectively.

As the cutoff frequency decreases,

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

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

Ideal low-pass filter function is a rectangular function.The inverse Fourier transform of a rectangular function is a sinc function.

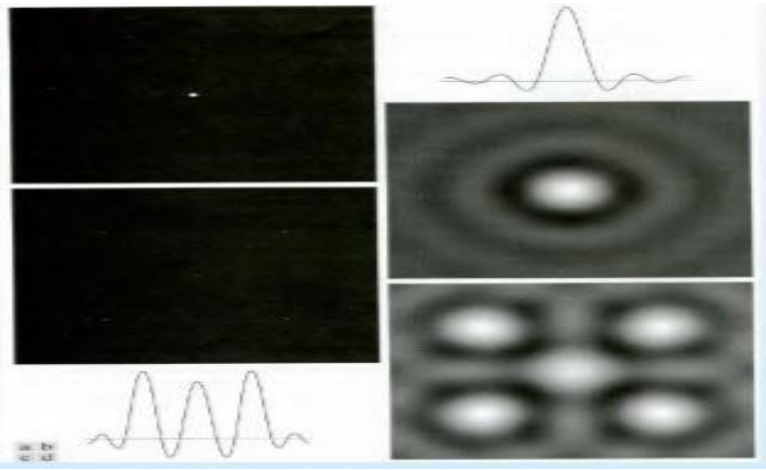


Fig. 23.Spatial representation of ILPFs of order 1 and 20 and corresponding intensity profiles through the center of the filters(the size of all cases is 1000x1000 and the cutoff frequency is 5), observe how ringing increases as a function of filter order.

BUTTERWORTH LOW-PASS FILTER:

Transfor funtion of a Butterworth lowpass filter (BLPF) of order n, and with cutoff frequency at a distance D_0 from the origin, is defined as

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

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

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

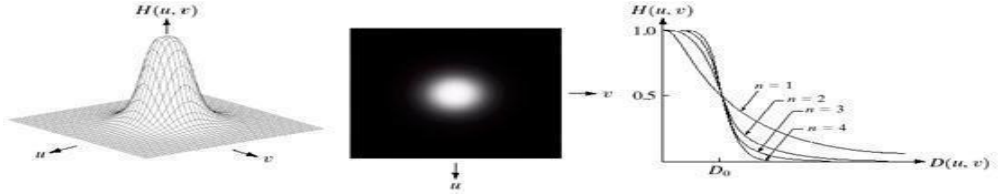


Fig. 24. perspective plot of a Butterworth lowpass-filter transfer function. (b) Filter displayed as an image. (c)Filter radial cross sections of order 1 through 4.

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

BUTTERWORTH LOW-PASS FILTERS OF DIFFEREN T FREQUENCIES:

Fig.25. (a) Original image.(b)-(f) Results of filtering using BLPFs of order 2, with cutoff frequencies at the radii. Fig. 25 shows the results of applying the BLPF of eq. to fig.(a), with n=2 and D0 equal tothe five radii in fig.(b) for the ILPF, we note here a smooth transition in blurring as a function of increasing cutoff frequency. Moreover, no ringing is visible in any of the images processed with this particular BLPF, a fact attributed to the filter's smooth transition between low and high frequencies.

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

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

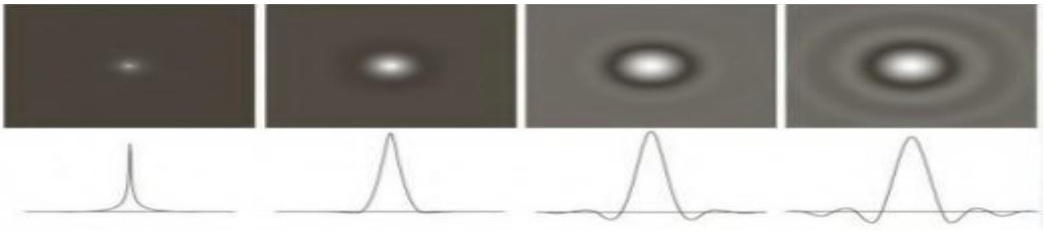


Fig.26. (a)-(d) Spatial representation of BLPFs of order 1, 2, 5 and 20 and corresponding intensity profiles through the center of the filters (the size in all cases is 1000 x 1000 and the cutoff frequency is 5) Observe how ringing increases as a function of filter order.

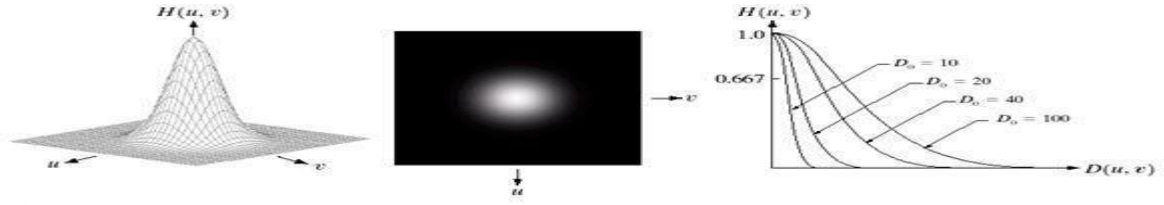
GAUSSIAN LOWPASS FILTERS:

The form of these filters in two dimensions is given by

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

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

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



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

Fig. 27.(a) Perspective plot of a GLPF transfer function. (b) Filter displayed as an image.(c). Filter radial cross sections for various values of D_0 frequencies atthe radii shown fig 27

Fig. 27.(a) Original image. (b)-(f) Results of filtering using GLPFs with cutoff

Fig.28. (a) Original image (784x 732 pixels). (b) Result of filtering using a GLPF with D0 = 100. (c) Result of filtering using a GLPF with D0 = 80. Note the reduction in fine skin lines in the magnified sections in (b) and (c).

Fig. 28.shows an application of lowpass filtering for producing a smoother, softer-looking result from a sharp original. For human faces, the typical objective is to reduce the sharpness of fine skin lines and small blemished.

IMAGE SHARPENING USING FREQUENCY DOMAIN FILTERS:

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

The filter function H(u,v) are understood to be discrete functions of size PxQ; that is the

discrete frequency variables are in the range u = 0,1,2,…….P-1 and v = 0,1,2,…….Q-1.

The meaning of sharpening is

- Edges and fine detail characterized by sharp transitions in image intensity
- Such transitions contribute significantly to high frequency components of Fourier transform.
- Intuitively, attenuating certain low frequency components and preserving high frequency components result in sharpening.

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

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

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

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

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

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

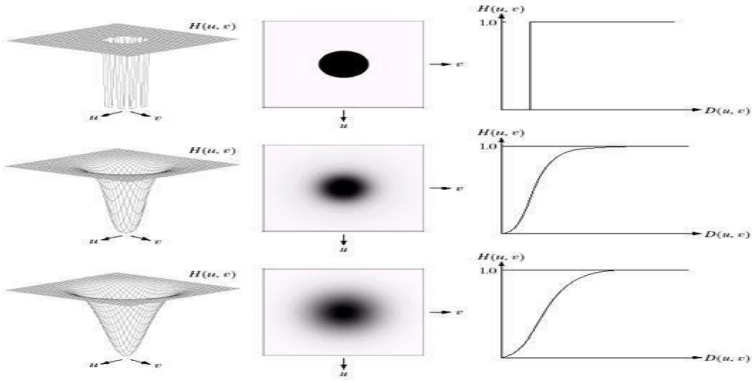


Fig: 29.Top row: Perspective plot, image representation, and cross section of a typical ideal high-pass filter. Middle and bottom rows: The same sequence for typical butter-worth and Gaussian high-pass filters.

IDEAL HIGH-PASS FILTER:

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

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

Where D_0 is the cutoff frequency and D(u,v) is given by eq. As intended, the IHPF is theopposite of the ILPF in the sense that it sets to zero all frequencies inside a circle of radius D_0 while passing, without attenuation, all frequencies outside the circle. As in case of the ILPF, the IHPF is not physically realizable.

SPATIAL REPRESENTATION OF HIGHPASS FILTERS:

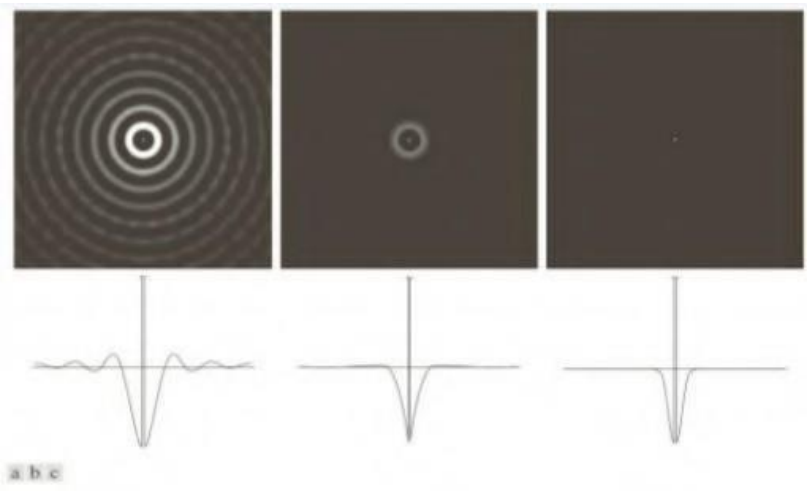


Fig.30. Spatial representation of typical (a) ideal (b) Butter-worth and (c) Gaussian frequency domain high-pass filters, and corresponding intensity profiles through their centers.

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

FILTERED RESULTS: IHPF:

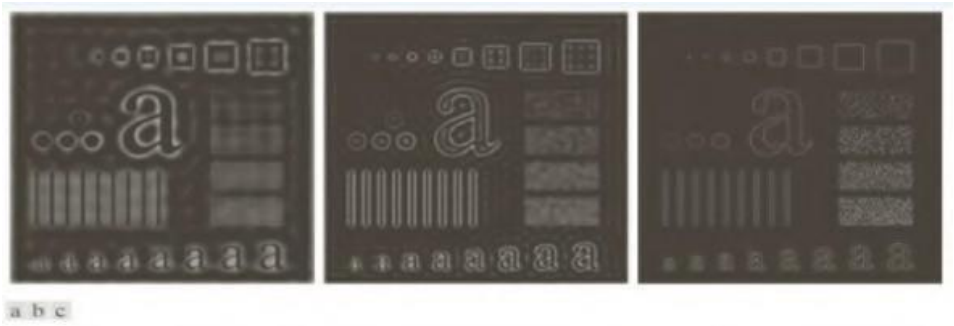


Fig.31. Results of high-pass filtering the image in Fig.(a) using an IHPF with D_0 = 30, 60, and 160.

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

BUTTER-WORTH HIGH-PASS FILTERS:

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

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

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

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

FILTERED RESULTS: BHPF:

Fig. 32.Results of high-pass filtering the image in Fig.2.2.2(a) using a BHPF of order 2

with D_0 = 30, 60, and 160 corresponding to the circles in Fig.2.2.2(b). These results are much smoother than those obtained with an IHPF.

GAUSSIAN HIGH-PASS FILTERS:

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

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

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

FILTERED RESULTS:GHPF:

a b c

Fig.33. Results of high-pass filtering the image in fig.(a) using a GHPF with D0 = 30, 60and 160, corresponding to the circles in Fig.(b).

IMAGE RESTORATION:

Restoration improves image in some predefined sense. It is an objective process. Restoration attempts to reconstruct an image that has been degraded by using a priori knowledge of the degradation phenomenon. These techniques are oriented toward modeling the degradation and then applying the inverse process in order to recover the original image. Restoration techniques are based on mathematical or probabilistic models of image processing. Enhancement, on the other hand is based on human subjective preferences regarding what constitutes a “good” enhancement result. Image Restoration refers to a class of methods that aim to remove or reduce the degradations that have occurred while the digital

image was being obtained. All natural images when displayed have gone through some sort of degradation:

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

Degradation Model:

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

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

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

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

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

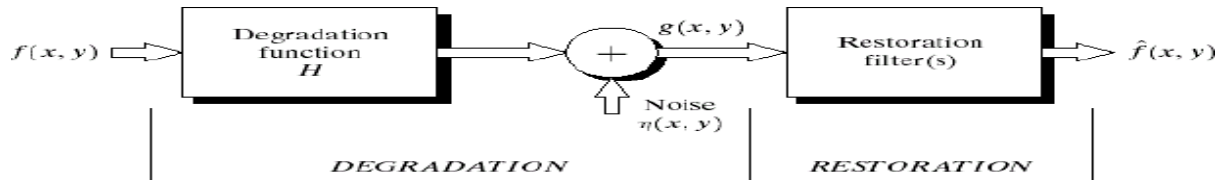


Fig: 34. model of the image Degradation / Restoration process

Noise Models:

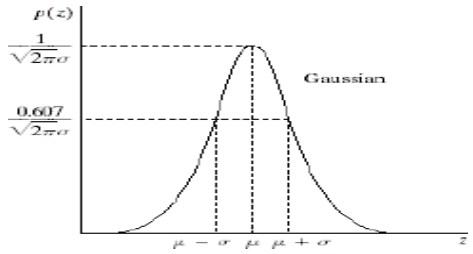
The principal source of noise in digital images arises during image acquisition and /or transmission. The performance of imaging sensors is affected by a variety of factors, such as environmental conditions during image acquisition and by the quality of the sensing elements themselves. Images are corrupted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

Gaussian Noise:

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

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

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



Rayleigh Noise:

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

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

The mean and variance of this density is

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

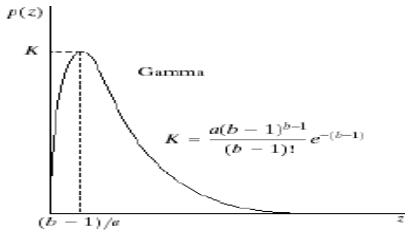
(iii) Gamma Noise:

The PDF of Erlang noise is given by

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

The mean and variance of this density are given by

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



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

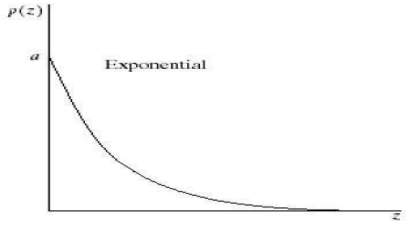
(iv) Exponential Noise:

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

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

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

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



(v) Uniform Noise:

The PDF of uniform noise is given by

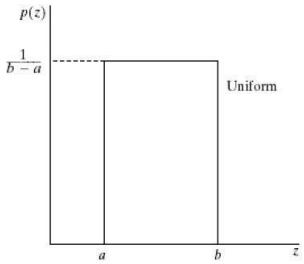
$$p_z(z) = \begin{cases} \frac{1}{b-a} & \text{if } a \le z \le b \\ 0 & \text{otherwise} \end{cases}$$

The mean and variance of this noise is

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

(vi) Impulse (salt & pepper) Noise:

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

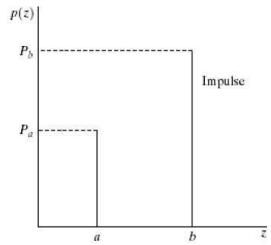


The PDF of bipolar (impulse) noise is given by

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

$$b > a$$

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



Restoration in the presence of Noise only- Spatial filtering:

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

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

or

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

The noise terms are unknown so subtracting them from g(x,y) or G(u,v) is not a realistic approach. In the case of periodic noise it is possible to estimate N(u,v) from the spectrum G(u,v).

So N(u,v) can be subtracted from G(u,v) to obtain an estimate of original image. Spatial

filtering can be done when only additive noise is present. The following techniques can be used to reduce the noise effect:

i) Mean Filter:

ii) (a)Arithmetic Mean filter:

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

This operation can be using a convolution mask in which all coefficients have value1/mn A

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

mean filter smoothes local variations in image Noise is reduced as a result of blurring. For every pixel in the image, the pixel value is replaced by the mean value of its neighboring pixels with a weight .This will resulted in a smoothing effect in the image.

(b)Geometric Mean filter:

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

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

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

(c) Harmonic Mean filter:

The harmonic mean filtering operation is given by the expression

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

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

(d) Order statistics filter:

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

(e) Median filter:

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

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

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

(e) Max and Min filter:

Using the l00th percentile of ranked set of numbers is called the max filter and is given by the equationIt is used for finding the brightest point in an image. Pepper noise in the image

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

has very low values, it is reduced by max filter using the max selection process in the sublimated area sky. The 0th percentile filter is min filter.

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

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

(f) Midpoint filter:

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

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

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

Periodic Noise by Frequency domain filtering:

These types of filters are used for this purpose-

Band Reject Filters:

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

Ideal Band reject Filter:

An ideal band reject filter is given by the expression

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

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

W- the width of the band

Do- the radial center of the frequency rectangle.

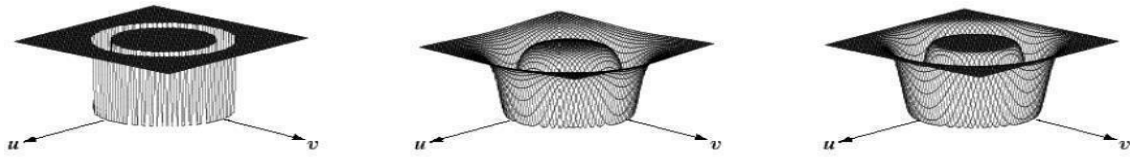
Butterworth Band reject Filter:

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

Gaussian Band reject Filter:

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

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



a b c

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

Band pass Filter:

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

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

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

Notch Filters:

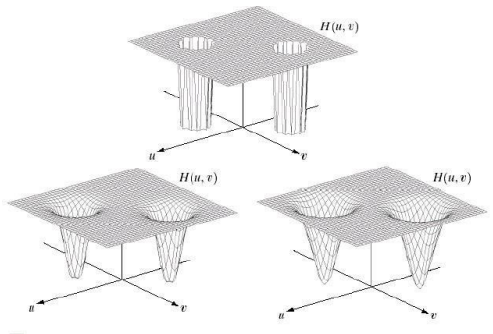
A notch filter rejects (or passes) frequencies in predefined neighborhoods about a center frequency.

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

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

$$D_1(u,v) = \sqrt{(u - M/2 - u_0)^2 + (v - N/2 - v_0)^2}$$
$$D_2(u,v) = \sqrt{(u - M/2 + u_0)^2 + (v - N/2 + v_0)^2}$$

Ideal, butterworth, Gaussian notch filters



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

INVERSE FILTERING:

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

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

We know that

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

Therefore

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

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

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

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

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

Minimum mean Square Error (Wiener) filtering:

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

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

The error measure is given by

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

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

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

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

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

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

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

Where H(u,v)= degradation function

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

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

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

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

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

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

Where K is a specified constant.

Constrained least squares filtering:

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

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

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

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

The minimum of a criterion function C defined as

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

Subject to the constraint

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

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

The frequency domain solution to this optimization problem is given by

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

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

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

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

IMAGE SEGMENTATION

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

A good segmentation is typically one in which:

- pixels in the same category have similar grey scale of multivariate values and form a connected region,
- neighboring pixels which are in different categories have dissimilar values.

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

Image segmentation is the key behind image understanding. Image segmentation isconsidered as an important basic operation for meaningful analysis and interpretation of image acquired. It is a critical and essential component of an image analysis and/or pattern recognition system, and is one of the most difficult tasks in image processing, which determines the quality of the final segmentation.

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

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

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

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

• In ***thresholding***, pixels are allocated to categories according to the range of values in which a pixel lies. Fig 2.1(a) shows boundaries which were obtained by thresholding the muscle fibers image. Pixels with values less than 128 have been placed in one category, and the rest have been placed in the other category. The boundaries between adjacent pixels in different categories has been superimposed in white on the original image. It can be seen that the threshold has successfully segmented the image into the two predominant fiber types.

• In ***edge-based segmentation***, an edge filter is applied to the image, pixels are classified as edge or non-edge depending on the filter output, and pixels which are not separated by an edge are allocated to the same category. Fig 2.1(b) shows the boundaries of connected regions after applying Prewitt's filter (3.4.2) and eliminating all non-border segments containing fewer than 500 pixels. (More details will be given in 4.2.)

• Finally, ***region-based segmentation algorithms*** operate iteratively by grouping together pixels which are neighbors and have similar values and splitting groups of pixels which are dissimilarin value. Fig 2.1(c) shows the boundaries produced by one such algorithm, based on the concept of watersheds, about which we will give more details in Fig 2.3

Note that none of the three methods illustrated in Fig 2.1 has been completely successful in segmenting the muscle fibers image by placing a boundary between every adjacent pair of fibers. Each method has distinctive faults. For example, in Fig 2.1(a) boundaries are well placed, but others are missing. In Fig 2.1(c), however, more boundaries are present, and they are smooth,but they are not always in exactly the right positions.

The following three sections will consider these three approaches in more detail. Algorithms will be considered which can either be fully automatic or require some manual intervention. The key points of the chapter will be summarized in Fig 2.4.

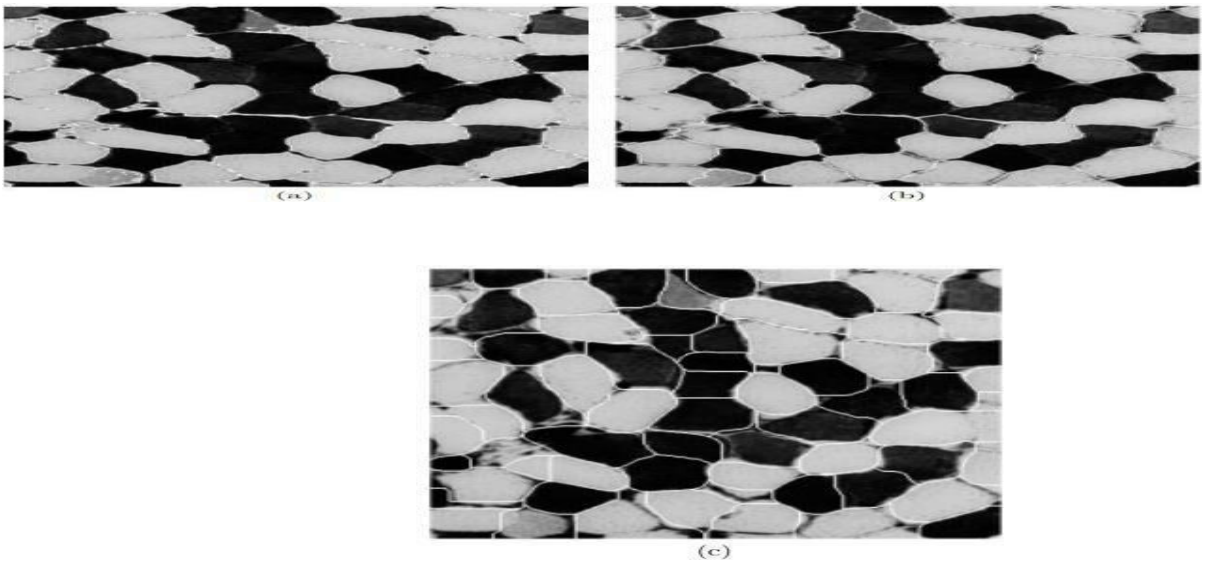


Fig.35 Boundaries produced by three segmentations of the muscle fibers images: (a) by thresholding, (b) connected regions after thresholding the output of Prewitt"s edge filter and removing small regions, (c) result produced by watershed algorithm on output from a variance filter with Gaussian weights (σ^2 = 96).

It is the prime area of research in computer vision.

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

(1) Histogram thresholding,

(2) Clustering (Fuzzy and Hard),

(3) Region growing, region splitting and merging,

(4) Discontinuity-based,

(5) Physical model- based,

(6) Fuzzy approaches, and

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

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

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

THRESHOLDING:

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

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

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

$$f_{ij} \leq t.$$

In many cases t is chosen manually by the scientist, by trying a range of values of t and seeing which one works best at identifying the objects of interest. Fig 36 shows some segmentations of the soil image Thresholds of 7, 10, 13, 20, 29 and 38 were chosen in Figs 36(a)
to (f) respectively, to identify approximately 10, 20, 30, 40, 50 and 60% of the pixels as being pores.

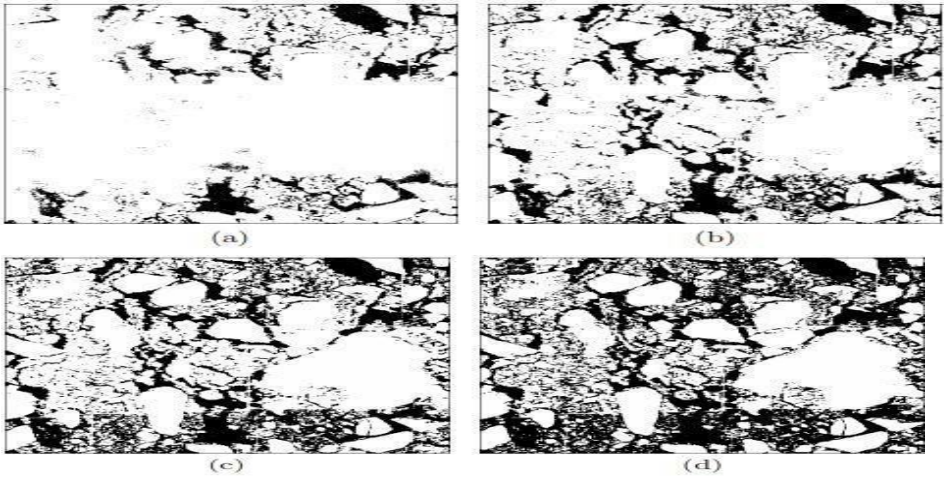


Fig 36(d), with a threshold of 20, looks best because most of the connected pore network evident.

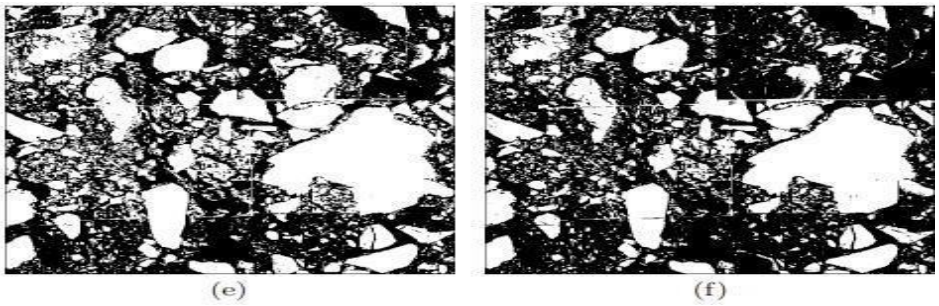


Fig.37 : six segmentations of the soil image, obtained using manually-selected thresholds of (a)7, (b) 10, (c) 13, (d) 20, (e) 29 and (f) 38. These correspond to approximately 10%, 20%,….60%, respectively, of the image being displayed as black.

HISTOGRAM-BASED THRESHOLDING

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

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

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

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

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

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

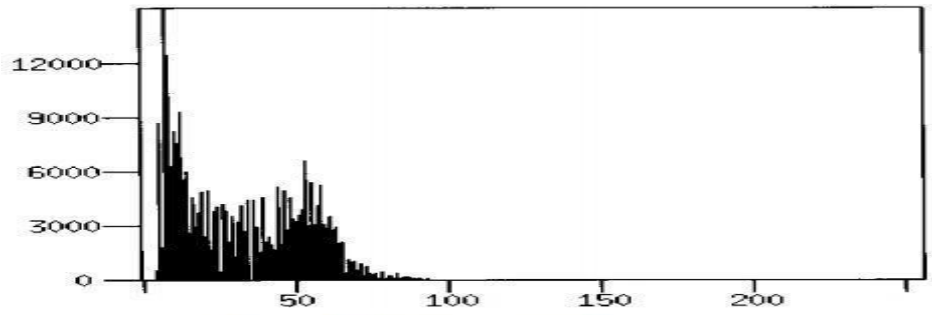


Figure 4.3: Histogram of soil image.

2. Calculate the mean pixel value in each category. For values less than or equal to t, thisis given by:

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

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

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

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

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

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

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

Fig 2.3 shows the histogram of the soil image. From an initial value of t = 28 (the median pixel value), the algorithm changed t to 31, 32, and 33 on the first three iterations, and then t remained unchanged. The pixel means in the two categories are 15.4 and 52.3. Fig 2.4(a) shows the result of using this threshold. Note that this value of t is considerably higher than the threshold value of20 which we favored in the manual approach.

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

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

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

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

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

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

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

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

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

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

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

There are three cases to consider:

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

$$2kB \geq C.$$

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

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

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

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

The criteria for category 1 are

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

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

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

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

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

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

Mathematically:

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

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

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

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

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

3. Re-estimate t by

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

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

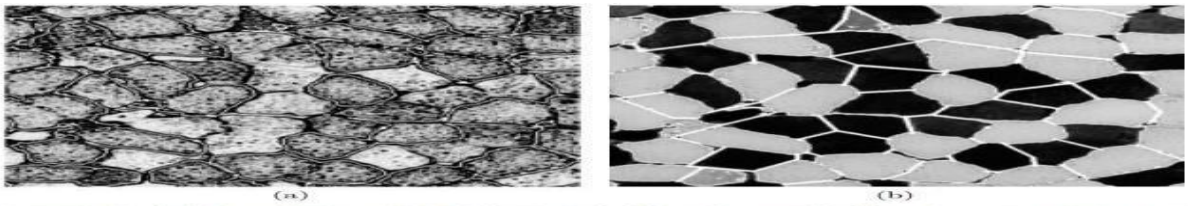


Figure 4.10: **(a)** Boundaries obtained by thresholding the muscle fibres image, superimposed on output for Prewitt's filter, with values between 0 and 5 displayed in progressively darker shades of grey and values in excess of 5 displayed as black. **(b)** Manual segmentation of the image by addition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

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

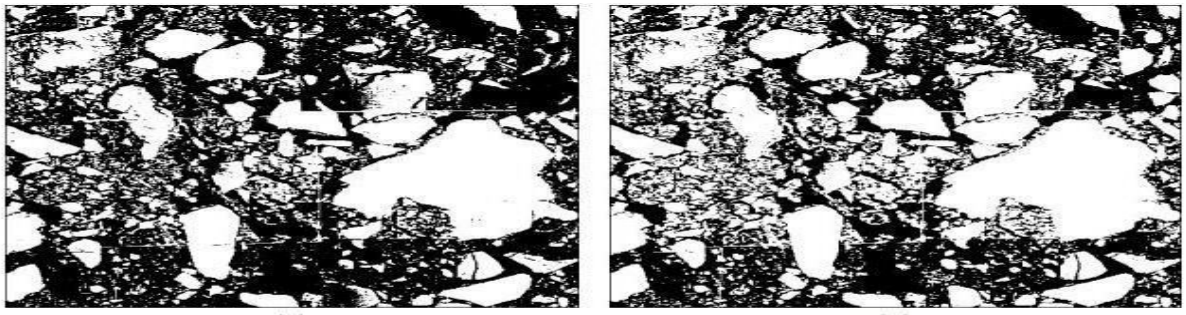


Figure 4.4: Segmentations of the soil image, obtained by thresholding at: **(a)** 33, selected by the intermeans algorithm, **(b)** 24, selected by the minimum-error algorithm.

The algorithm has also taken account of the air pixels being less variable in value than those for the soil matrix ($\sigma_1^2 = 30$, whereas $\sigma_2^2 = 186$). This is in accord with the left- most peak in the histogram plot (Fig 2.3) being quite narrow.

EDGE-BASED SEGMENTATION

As we have seen, the results of threshold-based segmentation are usually less than perfect. Often, a scientist will have to make changes to the results of automatic segmentation. One simple way of doing this is by using a computer mouse to control a screen cursor and draw boundary lines between regions. Fig 2.10(a) shows the boundaries obtained by thresholding the muscle fibres image (as already displayed in Fig 2.1(a)), superimposed on the output from Prewitt's edge filter, with the contrast stretched so that values between 0 and 5 are displayed as shades of grey ranging from white to black and values exceeding 5 are all displayed as black. This display can be used as an aid to determine where extra boundaries need to be inserted to fully segment all muscle fibres. Fig 2.10(b) shows the result after manually adding 71 straight lines.

Algorithms are available for semi-automatically drawing edges, whereby the scientist's rough lines are smoothed and perturbed to maximise some criterion of match with the image (see, for example, Samadani and Han, 1993). Alternatively, edge finding can be made fully automatic, although not necessarily fully successful. Fig 2.11(a) shows the result of applying Prewitt's edge filter to the muscle fibre image. In this display, the filter output has been thresholded at a value of 5: all pixels exceeding 5 are labelled as edge pixels and displayed as black. Connected chains of edge pixels divide the image into regions. Segmentation can be achieved by allocating to a single category all non-edge pixels which are not separated by an edge.

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

- Initialize the count of the number of categories by setting K = 0.
- Consider each pixel (i, j) in turn in a raster scan, proceeding row by row (i = 1,...,n), and for each value of i taking j = 1,...,n.
- One of four possibilities apply to pixel (i, j):

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

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

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

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

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

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

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

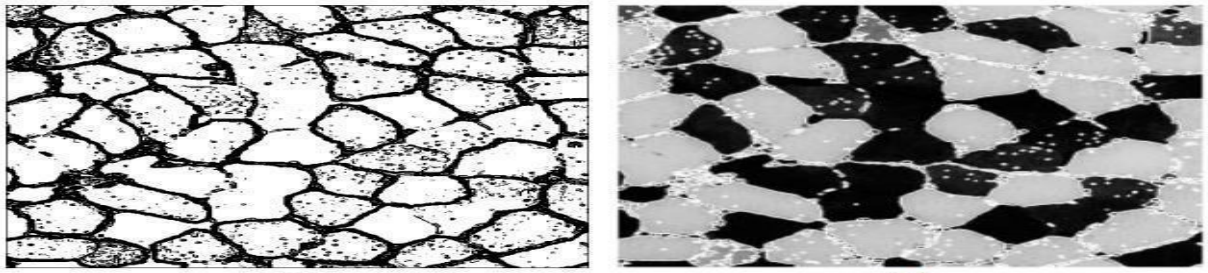
and if the neighbors have labels which have not been marked as equivalent, i.e. gi−1,j 6= gi,j−1 , then this needs to be done (because they are connected at pixel (i, j)). The equivalence is recorded by changing the entries in $h_1,...,h_K$, as follows: – Set l1 = min(gi−1,j ,gi,j−1) and l2 = max(gi−1,j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

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

gij → hgij for i, j = 1, . . . , n

After application of the labeling algorithm, superfluous edge pixels — that is, those which do not separate classes — can be removed: any edge-pixel which has neighbors only of one category is assigned to that category. Fig 2.11(b) shows the result of applying the labeling algorithm with edges as shown in Fig 2.11(a), and removing superfluous edge pixels. The white boundaries have been superimposed on the original image.

Similarly, small segments (say less than 500 pixels in size) which do not touch the borders of the image can be removed, leading to the previously displayed Fig 2.1(b). The segmentation has done better than simple thresholding, but has failed to separate all fibers because of gaps in output from Prewitt's edge filter. Martello (1976), among others, has proposed algorithms for bridging these gaps.



(a) (b)

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

REGION-BASED SEGMENTATION

Segmentation may be regarded as spatial clustering:

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

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

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

The distinction between edge-based and region-based methods is a little arbitrary.

Seeded region growing is a semi-automatic method of the merge type. Fig 36.(a) shows a set of seeds, white discs of radius 3, which have been placed inside all the muscle fibres, using an on-screen cursor controlled by a computer mouse. Fig 36(b) shows again the output from Prewitt's edge filter. Superimposed on it in white are the seeds and the boundaries of a segmentation produced by a form of watershed algorithm. The boundaries are also shown superimposed on the original muscle fibres image in Fig 36.(c). The watershed algorithm operates as follows (we will explain the name later).

For each of a sequence of increasing values of a threshold, all pixels with edge strength less than this threshold which form a connected region with one of the seeds are allocated to the corresponding fibre. When a threshold is reached for which two seeds become connected, the pixels are used to label the boundary. A mathematical representation of the algorithm is too complicated to be given here.

- The use of discs of radius 3 pixels, rather than single points, as seeds make the watershed results less sensitive to fluctuations in Prewitt's filter output in the middle of fibres.

- The results produced by this semi-automatic segmentation algorithm are almost as good as those shown in Fig 36.(b), but the effort required in positioning seeds inside muscle fibres is far less than that required to draw boundaries.

The watershed algorithm, in its standard use, is fully automatic. Again, we will demonstrate this by illustration. Fig 37. shows the output produced by a variance filter with Gaussian weights ($\sigma^2 = 96$) applied to the muscle fibers image after histogram-equalization. The white seeds overlie all the local minima of the filter output, that is, pixels whose neighbors all have larger values and so are shaded lighter. Note that it is necessary to use a large value of σ^2 to ensure that the filter output does not have many more local minima. The boundaries produced by the watershed algorithm have been added to Fig 37. An intuitive way of viewing the watershed algorithm is by considering the output from the variance filter as an elevation map: light areas are high ridges and dark areas are valleys. Each local minimum can be thought of as the point to which any water falling on the region drains, and the segments are the catchments for them. Hence, the boundaries, or watersheds, lie along tops of ridges. The previously mentioned Fig 2.1(c) shows this segmentation superimposed on the original image.

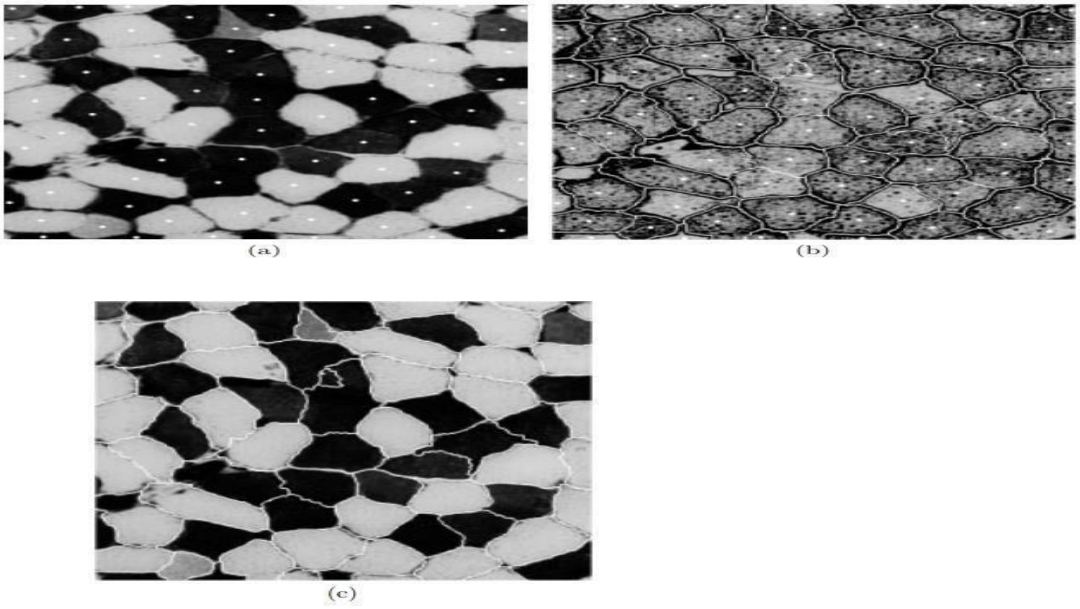


Fig.36: Manual segmentation of muscle fibres image by use of watersheds algorithm (a) manually positioned 'seeds' in centers of all fibres, (b) output from Prewitt's edge filter together with watershed boundaries, (c) watershed boundaries superimposed on the image.

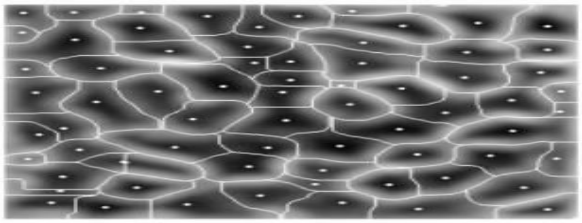


Figure 37: Output of variance filter with Gaussian weights ($\sigma 2 = 96$) applied to muscle fibres image, together with seeds indicating all local minima and boundaries produced by watershed algorithm.

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

The first stage is the splitting one. Initially, the variance of the whole image is calculated. If this variance exceeds the specified limit, then the image is subdivided into four quadrants. Similarly, if the variance in any of these four quadrants exceeds the limit it is further subdivided into four. This continues until the whole image consists of a set of squares of varying sizes, all of which have variances below the limit. (Note that the algorithm must be capable of achieving this because at the finest resolution of each square consisting of a single pixel the variances are taken to be zero.)

Fig 38.(a) shows the resulting boundaries in white, superimposed on the log- transformed SAR image, with the variance limit set at 0.60. Note that:

- Squares are smaller in non-uniform parts of the image.

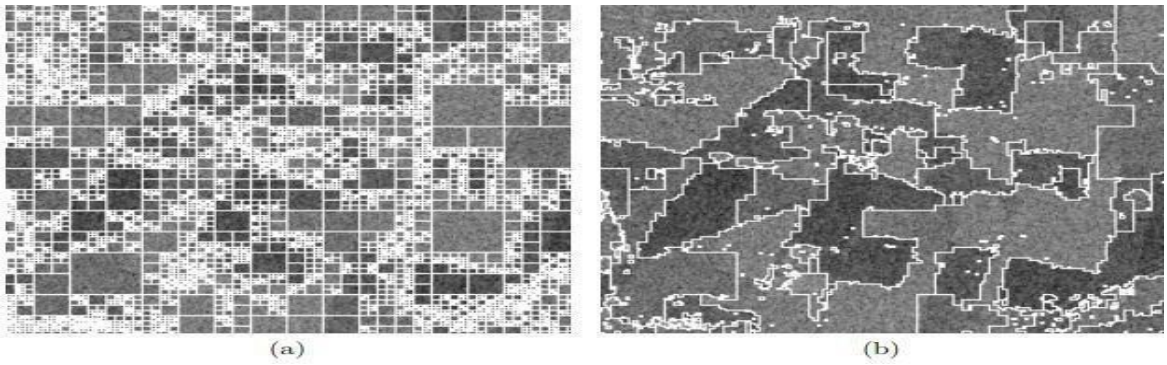


Figure 38: Region-growing segmentation of log-transformed SAR image: (a) division of image into squares with variance less than 0.60, obtained as first step in algorithm, (b) final segmentation, after amalgamation of squares, subject to variance limit of 0.60.

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

• The algorithm requires the image dimension, n, to be a power of 2. Therefore, the 250 × 250 SAR image was filled out to 256 × 256 by adding borders of width 3.

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

Fig 38(b) shows the boundaries produced by the algorithm, superimposed on the SAR image. Dark and light fields appear to have been successfully distinguished between, althoughthe boundaries are rough and retain some of the artefacts of the squares in Fig 38(a).
One possibility for improving segmentation results is to use an algorithm which over- segments an image, and then apply a rule for amalgamating these regions. This requires 'high-level' knowledge, which falls into the domain of artificial intelligence. (All that we have considered in this chapter may be termed 'low-level'.) For applications of these ideas in the area of remote sensing, see Tailor, Cross, Hogg and Mason (1986) and Ton, Sticklen and Jain (1991). It is possible that such domain-specific knowledge could be used to improve the automatic segmentations of the SAR and muscle fibres images, for example by constraining boundaries to be straight in the SAR image and by looking only for convex regions of specified size for the muscle fibres.

Basic Formulation

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

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

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

(C) Ri ∩ Rj = ϕ for all i and j, i = j

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

(e) P(Ri ∪ Rj) = FALSE for i ≠ j

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

symbols ∪ and ∩ represent set union and intersection, respectively.
The two regions Ri and Rj are said to be adjacent if their union forms a connected set.

Condition (a) indicates that the segmentation must be complete; that is every pixel must be in a region. Condition (b) requires that points in a region be connected in some predefined sense.Condition (c) indicates that the regions must be disjoint.
Condition (d) deals with the properties that must be satisfied by the pixels in a segmented region. For ex: P(Ri) = TRUE if all pixels in Ri have the same intensity level. Finally, condition(e) indicates that two adjacent regions Ri and Rj must be different in the sense of predicate P.

Point Detection

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

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

Figure 2. A mask for point detection

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

Where R is the response of the mask at any point in the image and T is non-negative threshold value. It means that isolated point is detected at the corresponding value (x, y). This formulation serves to measures the weighted differences between the center point and its neighbors since the gray level of an isolated point will be very different from that of its neighbors [ ]. The result of point detection mask is as shown in figure 39

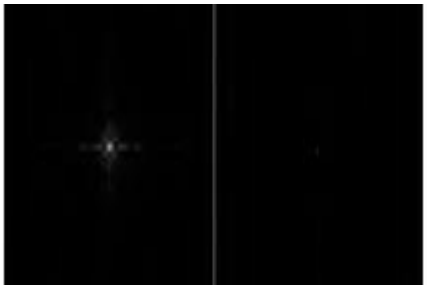


Figure 39. (a) Gray-scale image with a nearly invisible isolated black point (b) Image showing the detected point

Line Detection

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

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

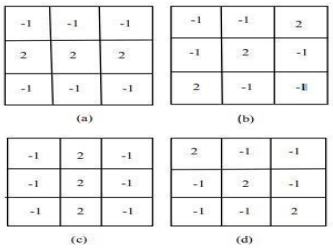


Figure 40. Line Detector masks in (a) Horizontal direction (b) 45° direction (c) Vertical direction (d) -45° direction The greatest response calculation from these matrices will yield the direction of the given pixel []. The result of line detection mask is as shown in figure 41

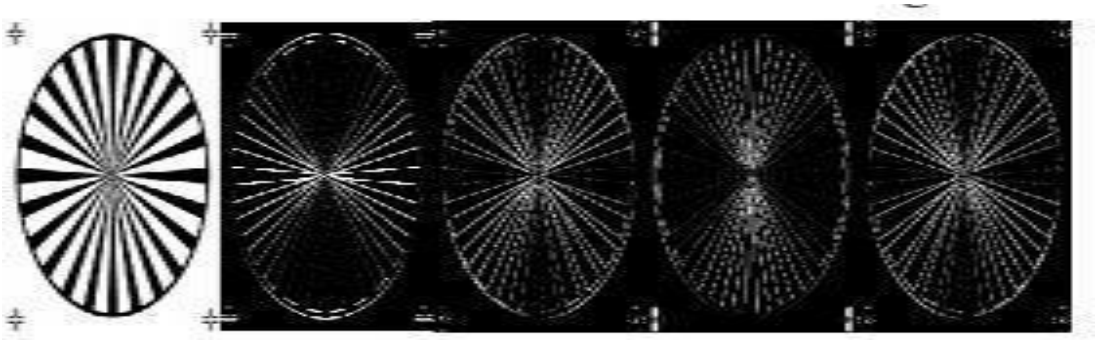


Figure 41. (a) Original Image (b) result showing with horizontal detector (c) with 45° detector (d) with vertical detector (e) with -45° detector

With the help of lines detector masks, we can detect the lines in a specified direction. For example, we are interesting in finding all the lines that are one pixel thick, oriented at -45°. For that, we take a digitized (binary portion of a wire-bond mask for an electronics circuit. The results are shown as in figure 6.

Edge detection

Since isolated points and lines of unitary pixel thickness are infrequent in most practical application, edge detection is the most common approach in gray level discontinuity segmentation. An edge is a boundary between two regions having distinct intensity level. It is very useful in detecting of discontinuity in an image. When the image changes from dark to white or vice-versa. The changes of intensity, first-order derivative and second-order derivative are shown in figure 42.

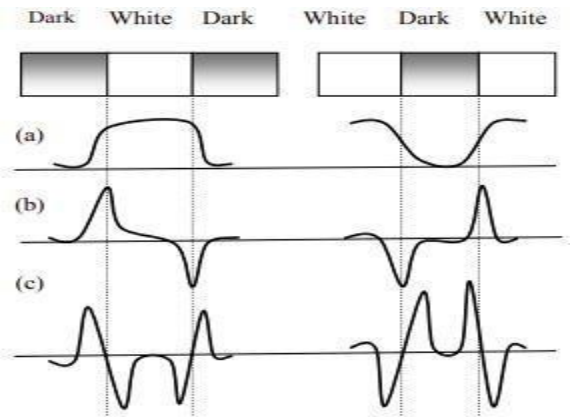


Figure 42. (a) Intensity profile (b) First-order derivatives (c) Second-order derivatives

First-order derivatives. First-order derivatives responds whenever there is discontinuity in intensity level. It is positive at the leading edge and negative at the trailing edge.

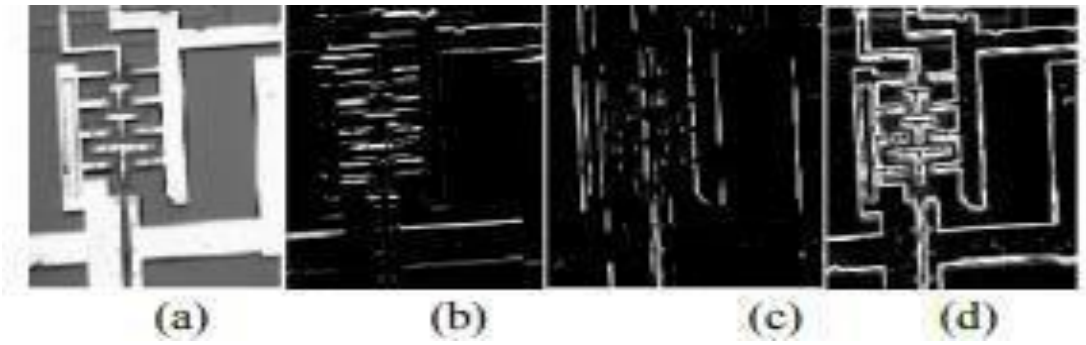


Figure 43 (a) Original Image (b) $\|Gx\|$component of the gradient along x-direction (c) ‖ Gy ‖component of the gradient along y-direction (d) Gradient Image $\|Gx\|+\|Gy\|$
There is several ways to calculate the image gradient:

Prewitt Edge operator

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

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

Figure 44. Masks used for Prewitt Edge operator

The mask finds the horizontal edges is equivalent to gradient in the vertical direction and the mask compute the vertical edges is equivalent to gradient in the horizontal direction. Using these two masks passing to the intensity image, we can find out and component at different location in an image. So, we

can find out the strength and direction of edge at that particular location (x, y).

Sobel Edge operator

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

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

Figure 45. Masks used for Sobel Edge operator

It gives the averaging effect over an image. It considers the effect due to the spurious noise in the image. It is preferable over prewitt edge operator because it gives the smoothing effect and by which we can reduce spurious edge which are generated because of noise present inthe image.

Second-order derivatives

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

Zero-crossing: It is useful to identify the exact location of the edge where there is gradual transition of intensity from dark to bright region and vice-versa. There are several second-order derivative operators: 46. Laplacian operator. The Laplacian mask

Laplacian operator. The Laplacian mask is given by:

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

Figure 46. Masks used for Laplacian operator

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

If we consider the diagonal elements:

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

1	1	1
1	8	1
1	1	1

Figure 47. Masks used for Laplacian operator using 8-connectivity

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

LoG operator

The LoG mask is given by

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

Figure 48. Masks used for LoG operator

The Gaussian operator is given by:

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

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

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

Canny operator

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

Image Pyramid

In low frequency represents smooth details, shapes, and colour in the image, and high frequency represents fine details and noise in the image.

The introducing of frequency domain make more possibility for image processing. For example, we

can just discard the high frequency responses to remove most noises from the image! Sounds great except that the beautiful object edge also gone with the noises. A major problem of frequency domain is that it will mix all frequency from every object in the image together, formally called "loss of locality", and most of the time we don't want to mix our faces' beautiful details with the detail of the grass in background.

Images that we can see in everyday life are represented using so-called Spatial domain, which is actually the things that most people will think of when talk about images. This spatial domain represents images by the luminance values of each image's location.

There are many processing operations that can be done with the spatial domain, such as basic low-pass filter and high-pass filter, adaptive filter, median filter, and many other things end with filter. Normally most of these filters work by using local information from each location. However, since images mostly contain complex informations, sometimes they are hard works to process in spatial domain.

There is another popular type of image representation known as the frequency domain. If we perform the Fast Fourier Transform to the image, we will get the totally different image called frequency spectrum. For those who didn't know about the frequency domain, let's simply explain like this: From the signal theory, every signal (including image, of course) can be express as a linear combination of a set of various-frequency sine and cosine signals. The process to transform from normal signal to frequency domain is called Fourier Transformation, and the transformed result – in this case, frequency spectrum- is actually a set of coefficients of each sine and cosine signals. In our case, low frequency represents smooth details, shapes, and colour in the image, and high frequency represents fine details and noise in the image.

The introducing of frequency domain make more possibility for image processing. For example, we can just discard the high frequency responses to remove most noises from the image! Sounds great except that the beautiful object edge also gone with the noises. A major problem of frequency domain is that it will mix all frequency from every object in the image together, formally called "loss of locality", and most of the time we don't want to mix our faces' beautiful details with the detail of the grass in background.

In order to get the advantages of both spatial domain and frequency domain, the new representation is invented and called spatial-frequency domain. This new domain give us the ability to deal with separate frequency easily as well as preserve the locality of the information. Our image pyramid is also in this domain.

The image pyramid is actually a representation of the image by a set of the different frequency-band images . For example, if we put our original Lena image to construct her pyramid, we may get, for simplicity, 3 layers of pyramid. The first image will represent low frequency band (smooth detail), the second image will represent middle frequency band (some detail), and the last image will represent high frequency band (finest detail). Image Pyramid is also called "multi-scale" and "multi-resolution" because of this characteristic.

Image pyramids types

- Gaussian pyramid
- Laplacian pyramid

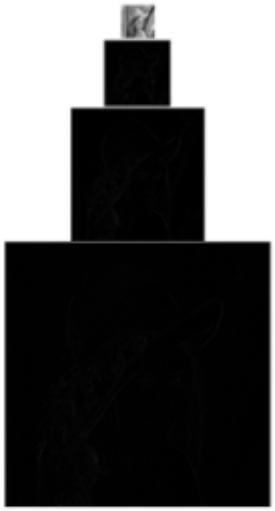


Fig 49.Example of Gaussian Pyramid

Fig 50. Example of Laplacian Pyramid

The Gaussian pyramid

• Smooth with Gaussians, because

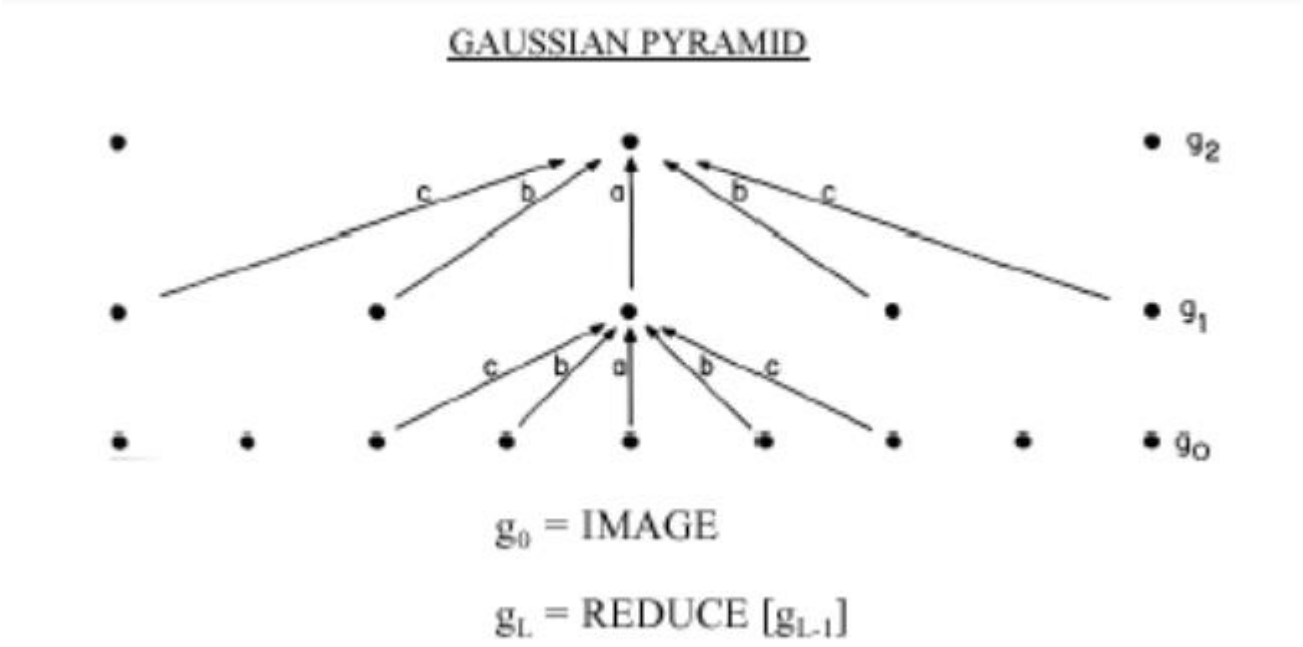
– A Gaussian*Gaussian = another Gaussian

• Gaussians are low pass filters, so representation is redundant.

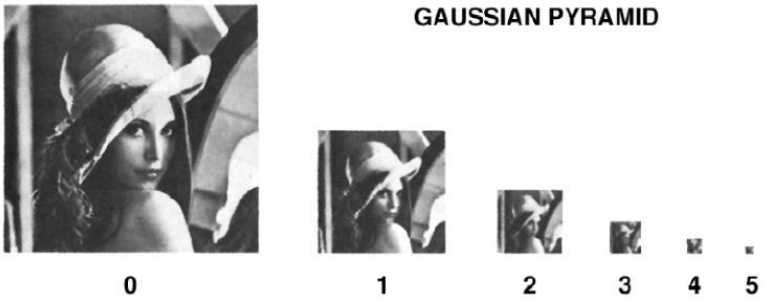
• Gaussian pyramid creates versions of the input image at multiple resolutions.

• This is useful for analysis across different spatial scales, but doesn't separate the image into different frequency bands.

The computational advantage of pyramids



The Gaussian Pyramid



Gaussian pyramids used for

• up- or down- sampling images.

• Multi-resolution image analysis

– Look for an object over various spatial scales

– Coarse-to-fine image processing: form blur estimate or the motion analysis on very low-resolution image, upsample and repeat. Often a successful strategy for avoiding local minima in complicated estimation tasks.

Template Matching with Image Pyramids

Input: Image, Template

1. Match template at current scale

2. Downsample image

3. Repeat 1-2 until image is very small

4. Take responses above some threshold, perhaps with nonmaxima suppression

Coarse-to-fine Image Registration

1. Compute Gaussian pyramid

2. Align with coarse pyramid

3. Successively align with finer pyramids

– Search smaller range

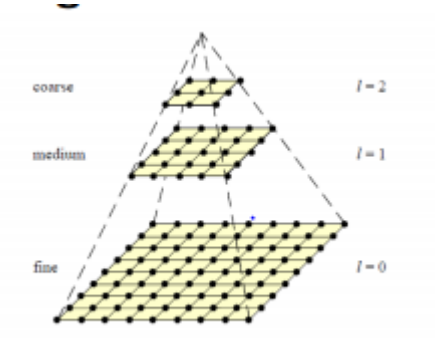


Fig 51.**Coarse-to-fine Image Registration**

The Laplacian Pyramid

• Synthesis

– Compute the difference between upsampled Gaussian pyramid level and Gaussian pyramid level.

– band pass filter - each level represents spatial frequencies (largely) unrepresented at other level.

• Laplacian pyramid provides an extra level of analysis as compared to Gaussian pyramid by breaking the image into different isotropic spatial frequency bands.

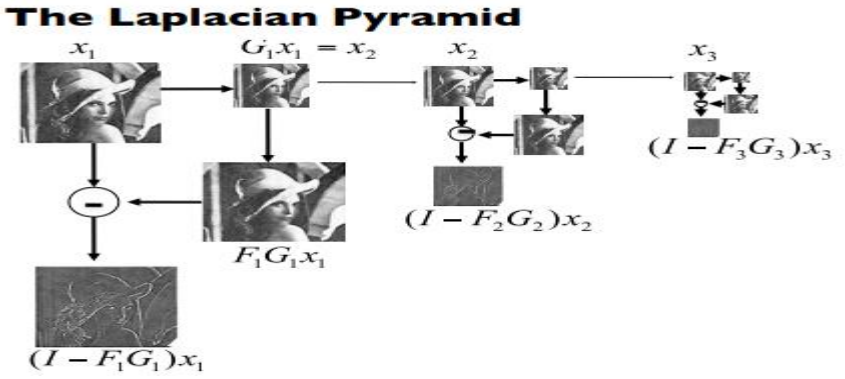


Fig.52. The Laplacian Pyramid

Laplacian pyramid reconstruction algorithm:

recover x1 from L1, L2, L3 and x4

G# is the blur-and-downsample operator at pyramid level

F# is the blur-and-upsample operator at pyramid level

Laplacian pyramid elements:

L1 = (I – F1 G1) x1

L2 = (I – F2 G2) x2

L3 = (I – F3 G3) x3

x2 = G1 x1

x3 = G2 x2

x4 = G3 x3

Reconstruction of original image (x1) from Laplacian pyramid elements:

x3 = L3 + F3 x4

x2 = L2 + F2 x3

x1 = L1 + F1 x2

Laplacian pyramid applications

• Texture synthesis

• Image compression

• Noise removal

Optical flow

Optical flow or optic flow is the pattern of apparent motion of objects, surfaces, and edges in a visual scene caused by the relative motion between an observer and a scene. Optical flow can also be defined as the distribution of apparent velocities of movement of brightness pattern in an image.

The basic assumption used in most optic flow algorithms is that when a point x in the image at time t moves to point x+dx in the image at time t+dt its luminance does not change (*the constant luminance assumption*):

$$f(x+dx,t+dt)=f(x,t)f(x+dx,t+dt)=f(x,t)$$

Note that dx=vdt dx=vdt where vv is the optic flow vector: the velocity vector at point xx at time tt:

$$f(x+vdt,t+dt)=f(x,t)f(x+vdt,t+dt)=f(x,t)$$

because dtdt is assumed to be infinitessimally small we may approximate the above equations in first order as:

$$f(x,t)+(\nabla f)(x,t)\top vdt+ft(x,t)dt=f(x,t)f(x,t)+(\nabla f)(x,t)\top vdt+ft(x,t)dt=f(x,t)$$

or equivalently:

$$(\nabla f)(x,t)\top v+ft(x,t)=0(\nabla f)(x,t)\top v+ft(x,t)=0$$

Note that given a video sequence $f:(x,t)\mapsto f(x,t)f:(x,t)\mapsto f(x,t)$ we can approximate the spatial gradient $\nabla f\nabla f$ and the temporal derivative ftft, but then we are left with just *one* equation and *two* unknowns: the two elements of the optic flow vector vv.

The impossibility to calculate the optic flow vector in a point is often called the *aperture problem*.

Solving the aperture problem

Basic idea: impose additional constraints

• most common is to assume that the flow field is smooth locally

• one method: pretend the pixel's neighbors have the same (u,v)

• If we use a 5x5 window, that gives us 25 equations per pixel

$$0 = I_t(\mathbf{p_i}) + \nabla I(\mathbf{p_i}) \cdot [u\ v]$$

$$\underset{25\times2}{\begin{bmatrix} I_x(\mathbf{p_1}) & I_y(\mathbf{p_1}) \\ I_x(\mathbf{p_2}) & I_y(\mathbf{p_2}) \\ \vdots & \vdots \\ I_x(\mathbf{p_{25}}) & I_y(\mathbf{p_{25}}) \end{bmatrix}} \underset{2\times1}{\begin{bmatrix} u \\ v \end{bmatrix}} = - \underset{25\times1}{\begin{bmatrix} I_t(\mathbf{p_1}) \\ I_t(\mathbf{p_2}) \\ \vdots \\ I_t(\mathbf{p_{25}}) \end{bmatrix}}$$

$$\underset{25\times2}{A} \qquad \underset{2\times1}{d} \qquad \underset{25\times1}{b}$$

Stereo Vision

Computer stereo vision is the extraction of 3D information from digital images, such as those obtained by a CCD camera. By comparing information about a scene from two vantage points, 3D information can be extracted by examining the relative positions of objects in the two panels.

Stereo vision is an imaging technique that can provide full field of view 3D measurements in an unstructured and dynamic environment. The foundation of stereo vision is similar to 3D perception in human vision and is based on triangulation of rays from multiple viewpoints. Each pixel in a digital camera image collects light that reaches the camera along a 3D ray. If a feature in the world can be identified as a pixel location in an image, we know that this feature lies on the 3D ray associated with that pixel. If we use multiple cameras, we can obtain multiple rays. The intersection of these rays is the 3D location of the feature.

In practice, the key problems to solve in stereo vision are:

- Identify which pixels in multiple images match the same world feature. This is known as the *correspondence* problem.
- Identify for each pixel in the image the corresponding ray in 3D space. This is known as the *calibration* problem, and requires accurate calibration of the camera optical parameters and physical location.

    - The correspondence problem is solved through image processing software. Depending on the application, the algorithm used may solve correspondences for only a sparse set of features in the image (feature-based algorithms), or attempt to find correspondences for every pixel in the image (dense stereo algorithms).

    - Camera calibration is done before using the stereo rig. For dense stereo algorithms, typically the images from the cameras must be remapped to an image that fits a pin-hole camera model. This remapped image is called the *rectified* image. For lenses with *barrel distortion*, straight lines in the world will appear curved in the image. In the rectified image, however, barrel distortion is removed and straight lines will appear straight.

The accuracy of the 3D results of stereo matching depends upon many factors such as image texture, image resolution, lens focal length and the separation between cameras. Increase in camera separation or narrower

field-of-view lenses improve the accuracy at long range. Higher image resolution increases the accuracy of the results but also may increase the processing time.

Template Matching

Template matching is a technique in computer vision **used for finding a subimage of a target image which matches a template image**. This technique is widely used in object detection fields such as surveillance [1], vehicle tracking [2], robotics [3], medical imaging [4], and manufacturing [5].

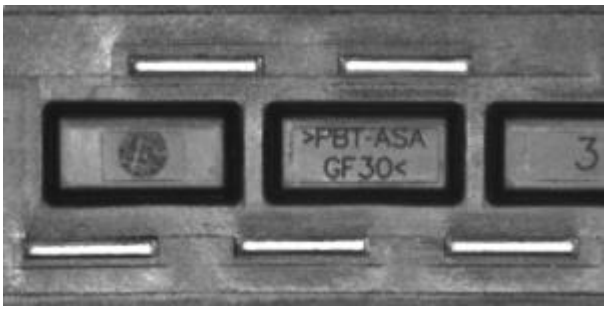
Template Matching is a high-level machine vision technique that identifies the parts on an image that match a predefined template. Advanced template matching algorithms allow to find occurrences of the template regardless of their orientation and local brightness.

Template Matching techniques are flexible and relatively straightforward to use, which makes them one of the most popular methods of object localization. Their applicability is limited mostly by the available computational power, as identification of big and complex templates can be time-consuming.

Template Matching techniques are expected to address the following need: provided a reference image of an object (the *template image*) and an image to be inspected (the *input image*) we want to identify all *input image* locations at which the object from the *template image* is present. Depending on the specific problem at hand, we may (or may not) want to identify the rotated or scaled occurrences.

Naive Template Matching

Imagine that we are going to inspect an image of a plug and our goal is to find its pins. We are provided with a *template image* representing the reference object we are looking for and the *input image* to be inspected.



Template image *Input image*

Perform the actual search in a rather straightforward way –position the *template* over the image at every possible location, and each time we will compute some numeric measure of similarity between the template and the image segment it currently overlaps with. Finally identify the positions that yield the best similarity measures as the probable template occurrences.

Image Correlation

One of the subproblems that occur in the specification above is calculating the *similarity measure* of the aligned template image and the overlapped segment of the input image, which is equivalent to calculating a similarity measure of two images of equal dimensions. This is a classical task, and a numeric measure of image similarity is usually called *image correlation*.

Cross-Correlation

Image1	Image2	Cross-Correlation
		19404780
		23316890
		24715810

The fundamental method of calculating the image correlation is so called *cross-correlation*, which essentially is a simple sum of pairwise multiplications of corresponding pixel values of the images.

Though we may notice that the correlation value indeed seems to reflect the similarity of the images being compared, cross-correlation method is far from being robust. Its main drawback is that it is biased by changes in global brightness of the images - brightening of an image may sky-rocket its cross-correlation with another image, even if the second image is not at all similar.

$$\text{Cross-Correlation}(\text{Image1}, \text{Image2}) = \sum_{x,y} \text{Image1}(x, y) \times \text{Image2}(x, y)$$

Normalized Cross-Correlation

Image1	Image2	NCC
		-0.417
		0.553
		0.844

Normalized cross-correlation is an enhanced version of the classic *cross-correlation* method that introduces two improvements over the original one:

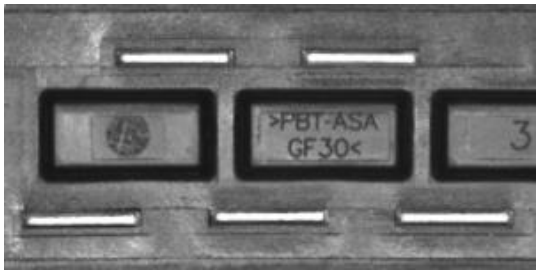
- The results are invariant to the global brightness changes, i.e. consistent brightening or darkening of either image has no effect on the result (this is accomplished by subtracting the mean image brightness from each pixel value).
- The final correlation value is scaled to [-1, 1] range, so that NCC of two identical images equals 1.0, while NCC of an image and its negation equals -1.0.

$$\mathrm{NCC}(\mathrm{Image1}, \mathrm{Image2}) = \frac{1}{N\sigma_1\sigma_2} \sum_{x,y} (\mathrm{Image1}(x,y) - \overline{\mathrm{Image1}}) \times (\mathrm{Image2}(x,y) - \overline{\mathrm{Image2}})$$

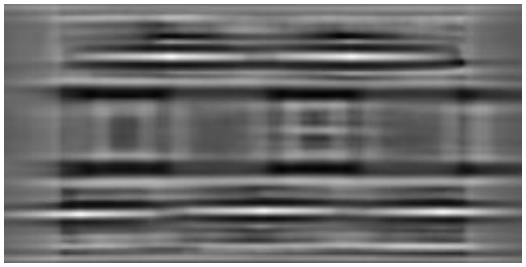
Template Correlation Image

Having introduced the Normalized Cross-Correlation - robust measure of image similarity - we are now able to determine how well the template fits in each of the possible positions. Represent the results in a form of an image, where brightness of each pixels represents the NCC value of the template positioned over this pixel (black color representing the minimal correlation of -1.0, white color representing the maximal correlation of 1.0).

Template image



Input image



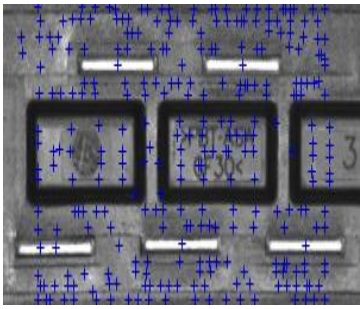
Template correlation image

Point Matching

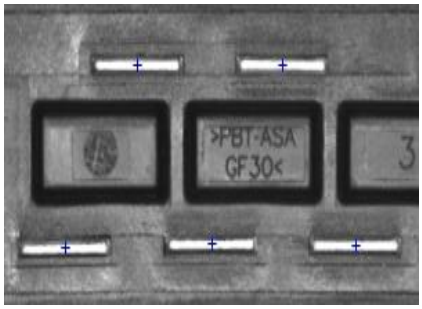
In image processing, point feature matching is **an effective method to detect a specified target in a cluttered scene**. This method detects single objects rather than multiple objects. For instance, by using this method, one can recognize one specific person in a cluttered scene, but not any other person. All that needs to be done at this point is to decide which points of the *template correlation image* are good enough to be considered actual matches. Usually we identify as matches the positions that (simultaneously) represent the template correlation:

- stronger that some predefined threshold value (i.e stronger that 0.5)
- locally maximal (stronger that the template correlation in the neighboring pixels)

Areas of template correlation above 0.75



Points of locally maximal template correlation



Points of locally maximal template correlation above 0.75

TEXT / REFERENCE BOOKS

1. Donald D Hearn, M. Pauline Baker, Computer Graphics C version, Pearson Education.
2. Dave Shreiner, Mason Woo, Jackie Neider, Tom Davis, OpenGL Programming Guide: The Official Guide to Learning OpenGL, (2013).
3. James D. Foley, Andries van Dam, Steven K. Feiner, John F. Hughes, Computer Graphics: Principles & Practice in C, Addison Wesley Longman.
4. Zhigang Xiang, Roy A Plastock, Computer Graphics, Schaums Outline, TMH.

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

SCHOOL OF ELECTRICAL AND ELECTRONICS

DEPARTMENT OF ELECTRONICS AND COMMMUNICATIONENGINEERING

UNIT - III

SCSA3007- INTRODUCTION TO VISUAL COMPUTING

UNIT 3 DIGITAL IMAGE GENERATION

The Graphics Pipeline- Lighting and Reflection Models- Shading-Texture Analysis and Texture Mapping-Aliasing- Global Illumination-Radiosity-Ray Tracing-Graphics Systems-APIs-3D Graphics Hardware.

Graphics pipeline

A graphics pipeline can be divided into three major steps: application, geometry and rasterization.

Application: The application step is executed in software, so it cannot be divided into individual steps that are executed in the form of a pipeline. However, it is possible to parallelize it on multi-core processors or multiprocessor systems. In the application step, changes are made to the scene as required, for example, due to user interaction with input devices or in the case of an animation. The new scene with all its primitives – mostly triangles, lines and points - is then forwarded to the next step of the pipeline.

Examples of tasks typically performed by the application step include collision detection, animation, morphing, and data management. The latter include, for example, acceleration techniques using spatial subdivision

schemes (Quadtree, Octree) that optimize the data currently stored in memory. The "world" and its textures of a modern-day computer game are much bigger than could be loaded into the available RAM or graphics memory.

Geometry: The geometry step, which is responsible for the majority of operations with polygons and their vertices, can be divided into five tasks:

1. Modelling Transformations: In addition to the objects, the scene also defines a virtual camera or viewer that indicates the position and viewing direction from which the scene is to be rendered. In order to simplify later projection and clipping, the scene is transformed so that the camera is at its origin, facing along the Z axis. The resulting coordinate system is called the camera coordinate system and the transformation is called the view transformation.

2. Illumination (Shading): A scene often contains light sources placed at different positions to make the lighting of the objects appear more realistic. In this case, a texture enhancement factor is calculated for each vertex based on the light sources and the material properties

associated with the corresponding triangle. In the subsequent screening step, the corner point values of a triangle are interpolated over its surface. General lighting (ambient light) is applied to all surfaces. It is the diffuse and thus direction independent brightness of the scene. The sun is a directional light source that can be assumed to be infinitely distant. The lighting effect of the sun on a surface is determined by the formation of the scalar product of the directional vector from the sun and the normal vector of the surface. If the value is negative, the surface is facing the sun.

3. Viewing Transformation (Perspective / Orthographic): This step transforms the visible volume into a cube with corner point coordinates (-1, -1, -1, -1) and (1,1,1,1); occasionally other target volumes are also used. This step is called projection, although it transforms a volume into another volume, because the resulting Z coordinates are not stored in the image, but only used for z-buffering in the subsequent rasterizing step. A central projection is used for a perspective image. In order to limit the number of displayed objects, two additional clipping planes are used; the visible volume is a pyramid stump (Frustum). For example, parallel or orthogonal projection is used for technical representations, because it has the advantage that all parallels in object space are also parallel in the image space and surfaces and volumes are the same size regardless of the distance to the viewer. For efficiency reasons, the camera and projection matrix are usually combined in a transformation matrix, so that the camera coordinate system is ignored. The resulting matrix is usually consistent for a single image, while the world matrix looks different for each object. In practice, therefore, view and projection are pre-calculated, so that only the World-Matrix has to be adjusted during display. However, more complex transformations such as vertex blending are possible. Freely programmable geometry shaders that change the geometry can also be executed. In the actual rendering step, the model matrix * camera (view) matrix * projection matrix is then calculated and finally applied to each individual point. The possible combination of matrices is illustrated in Figure 1. This transfers the points of all objects directly into the screen coordinate system (at least nearly, the value ranges of the axes are still -1... 1 for the visible area).

Clipping: Only the primitives that are located within the visible volume must actually be rasterized. Primitives that are completely out of sight are discarded; this is called frustum culling. Further culling procedures such as backface pulling, which reduce the number of primitives to be considered, can the oretically be performed in any step of the graphics pipeline. Primitives that are only partially inside the cube must be clipped against the cube. The advantage of the previous projection step is that clipping always takes place against the same cube. Only the - possibly clipped - primitives that are within the visible volume are

forwarded to the next step. 5. Projection (to Screen Space): To output the image to any viewport on the screen, a further transformation, the Window Viewport Transformation, must be applied. This is a shift, followed by scaling. The resulting coordinates are the device coordinates of the output device. The viewport contains 6 values: Height and width of the window in pixels, the upper left-hand corner of the window in window coordinates (usually 0.0) and the minimum and maximum values for Z (usually 0 and 1). On modern hardware, most of the geometry calculation steps are performed in the Vertex Shader. This is in principle freely programmable, but as a rule it takes over at least the transformation of the points and the lighting calculation. For the programming interface DirectX from version 10 onwards and OpenGL version 4, a user-defined vertex shader is unavoidable, whereas older versions have provided a standard shader.

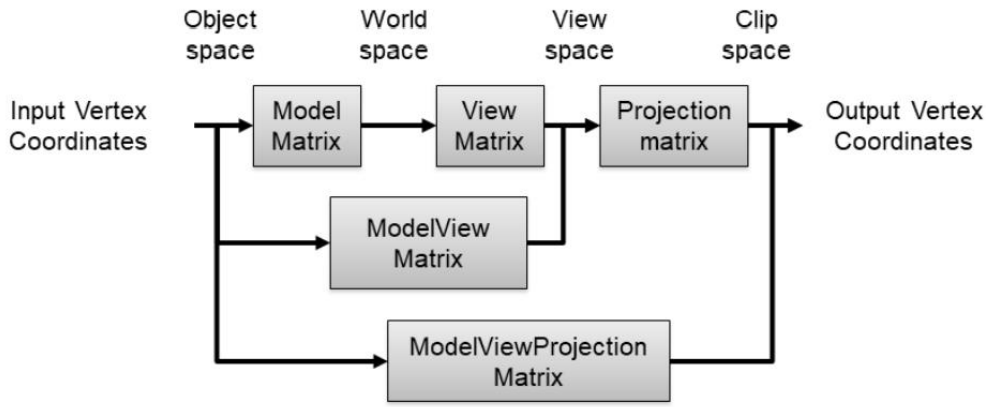


Figure 1: Possible pre-combinations of transformation matrices and their common names.

It depends on the respective implementation how these tasks are organized as actual pipeline steps that are executed in parallel. Rasterization: The geometry steps are followed by rasterization, i.e. the sampling of primitives into pixels on screen. In this step, all primitives are rasterized, i.e. discrete fragments are created from continuous surfaces. In this stage of the graphic pipeline, the raster points are also called fragments, i.e. each fragment corre sponds to one pixel in the frame buffer and this corresponds to one pixel of the screen. These can then be coloured (illuminated if necessary). Furthermore, it is necessary to determine the visible, i.e. closer to the viewer, of overlapping polygons. A z-buffer is usually used for this so-called masking calcula tion. The colour of a fragment depends on the illumination, texture, and other material properties of the visible primitive and is often interpolated using the triangular corner points. Where available, a fragment shader is

executed after the rasterization step for each fragment of the object. If a fragment is visible, it can be mixed with existing color values in the image if transparency is simulated or multi-sampling is used. In this step, one or more fragments become a pixel. To prevent the user from seeing the gradual screening of the primitives, double buffering is used. The rasterization takes place in a special memory area. As soon as the image has been completely rastered, it is copied into the visible area of the image memory (frame buffer).

Illumination model, also known as Shading model or Lightning model, is used to calculate the intensity of light that is reflected at a given point on surface. There are three factors on which lightning effect depends on:

1. **Light Source :**

Light source is the light emitting source. There are three types of light sources:

1. **Point Sources –** The source that emit rays in all directions (A bulb in a room).
2. **Parallel Sources –** Can be considered as a point source which is far from the surface (The sun).
3. **Distributed Sources –** Rays originate from a finite area (A tubelight).

Their position, electromagnetic spectrum and shape determine the lightning effect.

2. **Surface :**

When light falls on a surface part of it is reflected and part of it is absorbed. Now the surface structure decides the amount of reflection and absorption of light. The position of the surface and positions of all the nearby surfaces also determine the lightning effect.

3. **Observer :**

The observer's position and sensor spectrum sensitivities also affect the lightning effect.

1. Ambient Illumination :

Assume you are standing on a road, facing a building with glass exterior and sun rays are falling on that building reflecting back from it and the falling on the object under observation. This would be **Ambient Illumination**. In simple words, Ambient Illumination is the one where source of light is indirect.

The reflected intensity I_{amb} of any point on the surface is:

$$I_{amb} = K_a I_a$$

Where, I_a : ambient light intensity

K_a : surface ambient reflectivity, value of K_a varies from 0 to 1

2. Diffuse Reflection :

Diffuse reflection occurs on the surfaces which are rough or grainy. In this reflection the brightness of a point depends upon the angle made by the light source and the surface.
The reflected intensity I_{diff} of a point on the surface is:

$$I_{diff} = K_d I_p \cos(\theta) = K_d I_p (N \cdot L)$$

Where, I_p : the point light intensity
K_d : the surface diffuse reflectivity, value of K_d varies from 0 to 1
N : the surface normal
L : the light direction

3. Specular Reflection :

When light falls on any shiny or glossy surface most of it is reflected back, such reflection is known as Specular Reflection.
Phong Model is an empirical model for Specular Reflection which provides us with the formula for calculation the reflected intensity I_{spec}:

$$I_{spec} = W(\theta) I_l \cos^n(\Phi)$$

where, $W(\theta)$: K_s
L : direction of light source
N : normal to the surface
R : direction of reflected ray
V : direction of observer
Θ : Angle between L and R
Φ : angle between R and V

The Phong Lighting Model

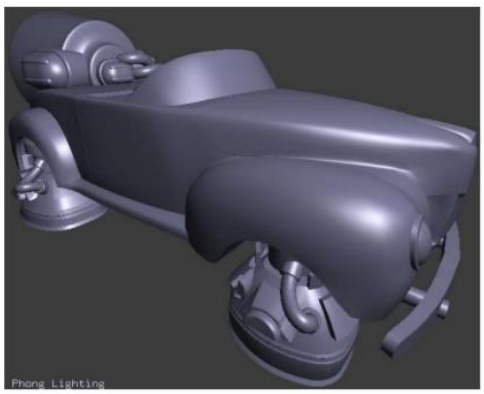
Phong lighting is the other major lighting model that is used in real time rendering, especially after the advent of programmable pipelines. Phong did not change any of Lamberts assumptions, and hence the cosine of the angle between the incident light vector and the surface normal is still used to calculate the diffuse component of the surface. Phong

did however create Phong interpolation that involves the interpolation of the vectors across the faces of the polygons as opposed to colors. The largest benefit of interpolating vectors across the polygon is that accurate specular highlights can be reproduced. The major drawback of calculating the lighting equation at every pixel using these interpolated vectors is of course that a lot more computational power is used.

The Phong Lighting Equation:

$$I = k_a + k_d f_{att}(N \cdot L) + k_s f_{att}(E \cdot R)^n$$

Two other advantages of Phong interpolation are that it is easy to add bump mapping to surfaces, and the visual appearance of the surface is significantly better than per-vertex interpolated lighting. Objects that are lit by Phong lighting tend to look like various types of plastic (depending on the specular exponent).



Screenshot of Phong lighting

Shading model

Shading model is used to compute the intensities and colors to display the surface. The shading model has two primary ingredients: properties of the surface and properties of the illumination falling on it. The principal surface property is its reflectance, which determines how much of the incident light is reflected.

We often use polygons to simulate curved surfaces. If these cases we want the colours of the polygons to flow smoothly into each other.

- flat shading
- goraud shading (color interpolation shading)

- phong shading (normal interpolation shading)

Flat Shading

- each entire polygon is drawn with the same colour
- need to know one normal for the entire polygon
- fast
- lighting equation used once per polygon

Given a single normal to the plane the lighting equations and the material properties are used to generate a single colour. The polygon is filled with that colour.

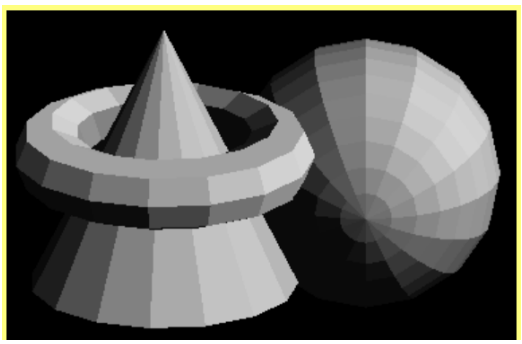


Fig 1.flat shaded scene

Goraud Shading

- colours are interpolated across the polygon
- need to know a normal for each **vertex** of the polygon
- slower than flat shading
- lighting equation used at each vertex

Given a normal at each vertex of the polygon, the colour at each vertex is determined from the lighting equations and the material properties. Linear interpolation of the colour values at each vertex are used to generate colour values for each pixel on the edges. Linear interpolation across each scan line is used to then fill in the colour of the polygon.

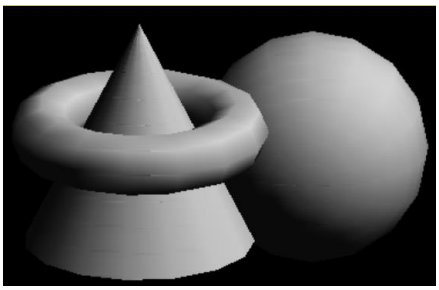


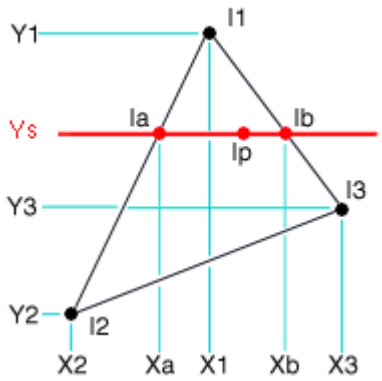
Fig 2.Goraud Shading

Phong Shading

- **normals** are interpolated across the polygon
- need to know a normal for each vertex of the polygon
- better at dealing with highlights than Goraud shading
- slower than Goraud shading
- lighting equation used at each pixel

Where Goraud shading uses normals at the vertices and then interpolates the resulting colours across the polygon, Phong shading goes further and interpolates tha normals. Linear interpolation of the normal values at each vertex are used to generate normal values for the pixels on the edges. Linear interpolation across each scan line is used to then generate normals at each pixel across the scan line.

Whether we are interpolating normals or colours the procedure is the same:



To find the intensity of Ip, we need to know the intensity of Ia and Ib. To find the intensity of Ia we need to know the intensity of I1 and I2. To find the intensity of Ib we need to know the intensity of I1 and I3.

Ia = (Ys - Y2) / (Y1 - Y2) * I1 + (Y1 - Ys) / (Y1 - Y2) * I2
Ib = (Ys - Y3) / (Y1 - Y3) * I1 + (Y1 - Ys) / (Y1 - Y3) * I3
Ip = (Xb - Xp) / (Xb - Xa) * Ia + (Xp - Xa) / (Xb - Xa) * Ib

Texture Analysis

Texture is a feature used to partition images into regions of interest and to classify those regions. Texture provides information in the spatial arrangement of colours or intensities in an image. Texture is characterized by the spatial distribution of intensity levels in a neighborhood. Texture is a repeating pattern of local variations in image intensity. Texture cannot be defined for a point.

Texture consists of texture primitives or texture elements, sometimes called texels. Texture can be described as fine, coarse, grained, smooth, etc. Such features are found in the tone and structure of a texture. Tone is based on pixel intensity properties in the texel, whilst structure represents the spatial relationship between texels. If texels are small and tonal differences between texels are large a fine texture results. If texels are large and consist of several pixels, a coarse texture results. There are two primary issues in texture analysis:

1. texture classification
2. texture segmentation

• Texture segmentation is concerned with automatically determining the boundaries between various texture regions in an image. Texture classification is concerned with identifying a given textured region from a given set of texture classes. Each of these regions has unique texture characteristics. Statistical methods are extensively used. e.g. GLCM, contrast, entropy, homogeneity.

There are three approaches to defining exactly what texture is:

 Structural: texture is a set of primitive texels in some regular or repeated relationship.

 Statistical: texture is a quantitative measure of the arrangement of intensities in a region. This set of measurements is called a feature vector.

 Modelling: texture modelling techniques involve constructing models to specify textures.

- Statistical methods are particularly usefulwhen the texture primitives are small, resulting in microtextures.
- When the size of the texture primitive is large,first determine the shape and properties of the basic primitive and the rules which governthe placement of these primitives, forming macrotextures.
- One of the simplest of the texture operators is the range or difference between maximum andminimum intensity values in a neighborhood.

The range operator converts theoriginal image to one in which brightness represents texture.

- Another estimator of texture isthe variance in neighborhood regions.

This is the sum of the squares of the differences between the intensity of the central pixel and itsneighbours.

Numeric quantities or statistics that describea texture can be calculated from the intensities (or colours) themselves

The statistical measures described so far areeasy to calculate, but do not provide any information about the repeating nature of texture.

A gray level co-occurrence matrix (GLCM)contains information about the positions ofpixels having similar gray level values.

- A co-occurrence matrix is a two-dimensionalarray, **P**, in which both the rows and the columns represent a set of possible image values.

Entropy is a measure of information content. It measures the randomness of intensity distribution. Such a matrix corresponds to an image in which there are no preferred graylevel pairs for the distance vector d. Entropy is highest when all entries in P[i,j] are of similar magnitude, and small when the entries in P[i,j] are unequal.

$$C_e = -\sum_i \sum_j P_d[i,j] \ln P_d[i,j]$$

Correlation is a measure of image linearity

$$C_c = \frac{\sum_i \sum_j [ijP_d[i,j]] - \mu_i \mu_j}{\sigma_i \sigma_j}$$

$$\mu_i = \sum iP_d[i,j], \quad \sigma_i^2 = \sum i^2 P_d[i,j] - \mu_i^2$$

Correlation will be high if an image contains a considerable amount of linear structure.

Texture Mapping

A texture map is a two-dimensional image file that can be applied to the surface of a 3D model to add color, texture, or other surface detail like glossiness, reflectivity, or transparency. Texture maps are developed to directly correspond to the UV coordinates of an unwrapped 3D model and are either devised from real-life photos or hand-painted in a graphics application like Photoshop or Corel Painter.

Texture maps are usually painted directly on top of the model's UV layout, which can be exported as a square bitmap image from any 3D software package. Texture artists usually work in layered files, with the UV coordinates on a semi-transparent layer that the artist will use as a guide for where to place specific details.

Color (Or Diffuse) Maps

As the name would imply, the most obvious use for a texture map is to add color or texture to the surface of a model. This could be as simple as applying a wood grain texture to a table surface, or as complex as a color map for an entire game character (including armor and accessories).

However, the term *texture map*, as it's often used is a bit of a misnomer — surface maps play a huge role in computer graphics beyond just color and texture. In a production setting, a character or environment's color map is usually just one of three maps that will be used for almost every single 3D model.

The other two "essential" map types are specular maps and bump/displacement, or normal maps.

Specular Maps

Specular maps (also known as gloss maps). A specular map tells the software which parts of a model should be shiny or glossy, and also the magnitude of the glossiness. Specular maps are named for the fact that shiny surfaces, like metals, ceramics, and some plastics show a strong specular highlight (a direct reflection from a strong light source). If you're

unsure about specular highlights, look for the white reflection on the rim of your coffee mug. Another common example of specular reflection is the tiny white glimmer in someone's eye, just above the pupil.

A specular map is typically a greyscale image and is absolutely essential for surfaces that aren't uniformly glossy. An armored vehicle, for example, requires a specular map in order for scratches, dents, and imperfections in the armor to come across convincingly. Similarly, a game character made of multiple materials would need a specular map to convey the different levels of glossiness between the character's skin, metal belt buckle, and clothing material.

Bump, Displacement, or Normal Map

A bit more complex than either of the two previous examples, bump maps are a type of texture map that can help give a more realistic indication of bumps or depressions on the surface of a model.

Consider a brick wall: An image of a brick wall could be mapped to a flat polygon plane and called finished, but chances are it wouldn't look very convincing in a final render. This is because a flat plane doesn't react to light the same way a brick wall would, with its cracks and coarseness.

To increase the impression of realism, a bump or normal map would be added to more accurately recreate the coarse, grainy surface of bricks, and heighten the illusion that the cracks between bricks are actually receding in space. Of course, it would be possible to achieve the same effect by modeling each and every brick by hand, but a normal mapped plane is much more computationally efficient. It's impossible to overstate the importance of normal mapping in the modern game industry — games simply could not look the way they do today without normal maps.

Bump, displacement, and normal maps are a discussion in their own right and are absolutely essential for achieving photo-realism in a render. Be on the lookout for an article covering them in depth.

Other Map Types to Know

Aside from these three map types, there are one or two others you'll see relatively often:

- **Reflection Map:** Tells the software which portions of the 3D model should be reflective. If a model's entire surface is reflective, or if the level of reflectivity is uniform a reflection map is usually omitted. Reflection maps are grayscale images, with black indicating 0% reflectivity and pure white indicating a 100% reflective surface.
- **Transparency Map:** Exactly like a reflection map, except it tells the software which portions of the model should be transparent. A common use for a transparency map would be a surface that would otherwise be very difficult, or too computationally expensive to duplicate, like a chain-link fence. Using a transparency, instead of modeling the links individually, can be quite convincing as long as the model doesn't feature too close to the foreground, and uses far fewer polygons.

Aliasing

Aliasing A problem with high resolution texturing is aliasing, which occurs when adjacent pixels in a rendered image are sampled from pixels that are far apart in a texture image. By down-sampling— reducing the size of a texture—aliasing can be reduced for far away or small objects, but then textured objects look blurry when close to the viewer. What we really want is a high resolution texture for nearby viewing, and down-sampled textures for distant viewing. A technique called mipmapping gives us this by prerendering a texture image at several different scales. For example, a 256x256 image might be down-sampled to 128x128, 64x64, 32x32, 16x16, and so on. Then it is up to the renderer to select the correct mipmap to reduce aliasing artifacts at the scale of the rendered texture.

Basic Ray Tracing

So far, we have considered only local models of illumination; they only account for incident light coming directly from the light sources. Global models include incident light that arrives from other surfaces, and lighting effects that account for global scene geometry. Such effects include: – Shadows – Secondary illumination (such as color bleeding) – Reflections of other objects, in mirrors, for example • Ray Tracing was developed as one approach to modeling the properties of global illumination. • The basic idea is as follows: For each pixel: – Cast a ray from the eye of the camera through the pixel, and find the first surface hit by the ray. – Determine the surface radiance at the surface intersection with a combination of local and global models. – To estimate the global component, cast rays from the surface point to possible incident directions to determine how much light comes from each direction. This leads to a recursive form for tracing paths of light backwards from the surface to the light sources. Aside: Basic Ray Tracing is also sometimes called Whitted Ray Tracing, after its inventor, Turner Whitted.

Computational Issues
• Form rays.
• Find ray intersections with objects.
• Find closest object intersections.
• Find surface normals at object intersection.
• Evaluate reflectance models at the intersection.

Local vs. Global Illumination

Local Illumination Models
■ e.g. Phong
■ Model source from a light reflected once off a surface towards the eye
■ Indirect light is included with an ad hoc "ambient" term which is normally constant across the scene.
Global Illumination Models
■ e.g. ray tracing or radiosity (both are incomplete)
■ Try to measure light propagation in the scene
■ Model interaction between objects and other objects and objects and their environment

Radiosity is a global illumination algorithm used in 3D computer graphics rendering. Unlike direct illumination algorithms (such as Ray tracing), which tend to simulate light reflecting only once off each surface, global illumination algorithms such as Radiosity simulate the many reflections of light around a scene, generally resulting in softer, more natural shadows and reflections.

Radiosity as a rendering method was introduced in 1984 by researchers at Cornell (C. Goral, K. E. Torrance, D. P. Greenberg and B. Battaile) in their paper "Modeling the interaction of light between diffuse surfaces". The theory had been in use in engineering, to solve problems in radiative heat transfer, since about 1950.

The Radiosity Method

The computation of lighting via radiosity is unlike many traditional computer graphics lighting computation because it is view independent. The intensity of surfaces in the model are computed before any view calculations are made. This difference can be thought of as the difference between demand-driven and data-driven lighting computation.

Demand Driven Lighting Calculation

This is what is typically used with a z-buffer, raytrace, or painter's algorithm rendering system. The renderer computes the location of a polygon, or what polygon is present at a

particular pixel and then needs to know the intensity and/or color to draw that polygon/pixel. The lighting calculation is done as the last step and is driven by the demand to know what color/intensity to display on the screen.

Data Driven Lighting Calculation

Radiosity uses a data driven approach to lighting calculation. Instead of computing the lighting as the last step in the rendering process, it is done beforehand. Certain surfaces in the scene are given initial intensities, and the effect they have on other surfaces in the scene is computed in an iterative manner. This is done independently of the location of the viewer. The presence of these lit surfaces in the model is what drives the computation of the system.

The Radiosity Model

The description of the radiosity model that follows is based on the original radiosity system developed by Goral et al. [GOR84].

The radiosity method is based upon a simple model of energy transfer. At each surface in a model the amount of energy that is given off is comprised of the energy that the surface emits internally, plus the amount of energy that is reflected off the surface. The amount of energy that is reflected off the surface can be further characterized by the product of the amount of energy incident on the surface and a reflectivity constant of the surface.

$$B_j = \rho_j H_j + E_j$$

B_j – Radiosity of surface j

ρ_j – Reflectivity of surface j

H_j – Energy incident on surface j

E_j – Energy emitted by surface j

The radiosity of a surface is the energy that is given off. This is what is used to determine the intensity of the surface and is what is being solved for. The amount of light emitted from a surface must be specified as a parameter in the model, just as in traditional lighting methods where the location and intensity of light sources must be specified. The reflectivity of the surface must also be specified in the model, just as in traditional lighting methods. The only unknown in the equation is the amount of incident light hitting the surface. This can be found by summing for all other surfaces the amount of energy that they contribute to this surface.

$$H_j = \sum_{i=1}^{N} B_i F_{ij}, \quad j = 1..N$$

H_j – Energy incident on surface j
B_i – Radiosity of surface i
F_{ij} – Form factor ij

The form factor in the above equation is defined to be the fraction of energy that leaves surface i and lands on surface j, and is therefore a number in the range (0..1). This form factor can be computed via analytical means, or through a geometric analog. See [COH85] for more information on form factor computation.

The radiosity equation now looks like this:

$$B_j = E_j + \rho_j \sum_{i=1}^{N} B_i F_{ij}, \quad j = 1..N \qquad (1)$$

The Radiosity Matrix
The derived radiosity equations (1) form a set of N linear equations in N unknowns. This leads nicely to a matrix solution:

$$\begin{bmatrix} 1-\rho_1 F_{11} & -\rho_1 F_{12} & \cdots & -\rho_1 F_{1N} \\ -\rho_2 F_{21} & 1-\rho_2 F_{22} & \cdots & -\rho_2 F_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ -\rho_N F_{N1} & -\rho_N F_{N2} & \cdots & 1-\rho_N F_{NN} \end{bmatrix} \begin{bmatrix} B_1 \\ B_2 \\ \vdots \\ B_N \end{bmatrix} = \begin{bmatrix} E_1 \\ E_2 \\ \vdots \\ E_N \end{bmatrix}$$

This matrix has two interesting properties: it is diagonally dominant and is therefore guaranteed to converge when using Gauss Seidel iteration, and the upper right of the matrix is computable from the lower left. Alternate methods for computing the solution of this matrix have been proposed by Cohen et al. Their alterations allow for a faster convergence towards the correct solution than simple Gauss Seidel iteration.

A BASIC GRAPHICS SYSTEM

Let us consider the organization of a typical graphics system we might use . As our initial emphasis will be on how the applications programmer sees the system , we shall omit detaails of the hardware. A blockdiagram of our system is shown in fig17-1. There are four key types of elements in our system:

. A processor
. Memory
. Output devices
. Input devices

The model is general enough to include workstations , personal computers , terminals attached to a central time-shared computer ,and sophisticated image-generation systems. In most ways , this block diagram is that of a standard computer. How each element is specialized for computer graphics will characterize this diagram as one of a graphics system , rather than one of a general-purpose computer.

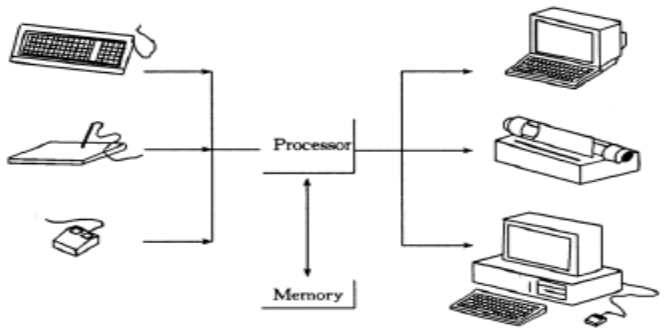


Fig.17-1 A Graphics System

1. The Processor

Within the processor box , two types of processing take place. The first is picture formation processing. In this stage , the user program or commands are processed. The picture is formed from the elements (lines , text)available in the system using the desired attributes. such as line color and text font. The user interface is a part of this processing.

The picture can be specified in a number of ways , such as through an interactive menu-controlled painting program or via a C program using a graphics library. The physical processor used in this stage is often the processor in the workstation or host computer.

The second kind of processing is concerned with the display of the picture. In a raster system , the specified primitives must be scan converted. The screen must be refreshed to avoid flicker. Input from the user might require objects to be repositioned on the display. The kind of processor best suited for these jobs is not the standard type of processor found

in most computers. Instead , special boards and chips are often used. As we have already noted , one of the elements that distinguishes real-time graphics systems is their use of display processors. Since we have agreed to stay at the block-diagram level for now , however , we shall not explore these architectures in any detail until later.

3. Memory

There are often two distinct types of memory employed in graphics systems. For the processing of the user program , the memory is similar to that of a standard computer , as the picture is formed by a standard type of arithmetic processing. Display processing , however , requires high-speed display memory that can be accessed by the display processor , and , in raster systems , memory for the frame buffer. This display memory usually is different in both its physical characteristics and its organization from what is used by the picture processor. At this point , we need not consider details of how memory can be organized. You should be aware that the way the internals of our processor and memory boxes are organized distinguishes a slow system from a real-time picture-generating system , such as a flight simulator. However , from our present perspective , we shall emphasize that all implementations have to do the same kinds of tasks to produce output.

4. Output Devices

Our basic system has one or more output devices. As raster displays are the dominant type , we shall assume there is a raster-scan CRT on our system. We shall consider the frame buffer to be part of the display memory. In a self-contained system such as a workstation , the display is an integral part of the system , so the transfer of information from the processor to the display will happen rapidly. When the display is separate , such as with a graphics terminal , the speed of the connection is much slower. Terminals with raster displays usually must have their own frame buffers , so the displays can be refreshed locally. In our simple system , we might also have other displays , such as a plotter , to allow us to produce hardcopy.

5. Input Devices

A simple system may have only a keyboard to provide whatever input is necessary. Keyboards provide digital codes corresponding to sequences of keystrokes by a user. These sequences are usually interpreted as codes for characters. If individual keystrokes or groups of keystrokes are interpreted as graphical input , the keyborad can be used as a complex input device. For example , the "arrow" keys available on most keyboards can be used to direct the movement of a cursor on the screen. Most graphics systems will provide at least

one other input device. The most common are the mouse , the lightpen, the joystick , and the data tablet. Each can provide positional information to the system and each usually is equipped with one or more buttons to provide signals to the processor.

From the programmer's perspective , there are numerous important issues with regard to the imput and output devices. We must consider how the program can communicate with these devices. We must decide what kinds of input and output can be produced. We will be interested in how to control multiple devices , so that we can choose a particular device for our input , and can direct our output to some group of the available output devices.

How the Interactive Graphics Display Works

The modern graphic display is very simple in construction. It consists of the three components shown in figure 1.2 below.

(1) Frame Buffer

(2) Monitor like a TV set without the tuning and receiving electronics.

(3) Display Controller It passes the contents of the frame buffer to the monitor.

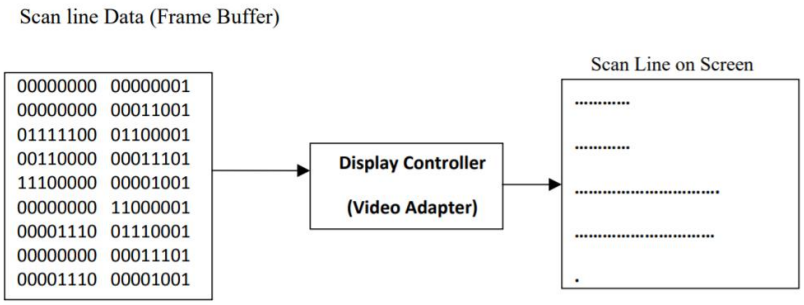


Fig 4.The modern graphic display

Inside the frame buffer the image is stored as a pattern of binary digital numbers, which represent a array of picture elements, or pixels. In the simplest case, where you want to store only black and white images, you can represent black pixels by —1‘s‖ and white pixels by —0‘s‖ in the frame buffer. Therefore, a array of black and white pixels of 16X16 could be represented by 32 bytes, stored in frame buffer. The display controller reads each successive byte of data from the frame buffer and converts its 0‘s and 1‘s into corresponding video signals. This signal is then fed to the monitor, producing a black and white image on the

screen. The display controller repeats this operation 30 times a second to maintain a steady picture on the monitor. If you want to change the image, then you need to modify the frame buffer‘s contexts to represent the new pattern of pixels. 5 Display Devices The principle of producing images as collections of discrete points set to appropriate colours is now widespread throughout all fields of image production. The most common graphics output device is the video monitor which is based on the standard cathode ray tube(CRT) design, but several other technologies exist and solid state monitors may eventually predominate. Basic Operation of CRT The phosphor then emits a small spot of light at each position contacted by the electron beam. Because the light emitted by the phosphor fades very rapidly, some method is needed for maintaining the screen picture. One Way to keep the phosphor glowing is to redraw the picture repeatedly by quickly directing the electron beam back over the same points. This type of display is called a refresh CRT. Beam passes between two pairs of metal plates, one vertical and other horizontal. A voltage difference is applied to each pair of plates according to the amount that the beam is to be deflected in each direction. As the electron beam passes between each pair of plates, it is bent towards the plate with the higher positive voltage. To get the proper deflection, adjust the current through coils placed around the outside of the CRT loop. The primary components of an electron gun in a CRT are the heated metal cathode and a control grid (Fig. 2.2). Heat is supplied to the cathode by directing a current through a coil of wire, called the filament, inside the cylindrical cathode structure. This causes electrons to be "boiled off" the hot cathode surface. In the vacuum inside the CRT envelope, the free, negatively charged electrons are then accelerated toward the phosphor coating by a high positive voltage. The accelerating voltage can be generated with a positively charged metal coating on the in- side of the CRT envelope near the phosphor screen, or an accelerating anode can be used, as in Fig. . Sometimes the electron gun is built to contain the accelerating anode and focusing system within the same unit.

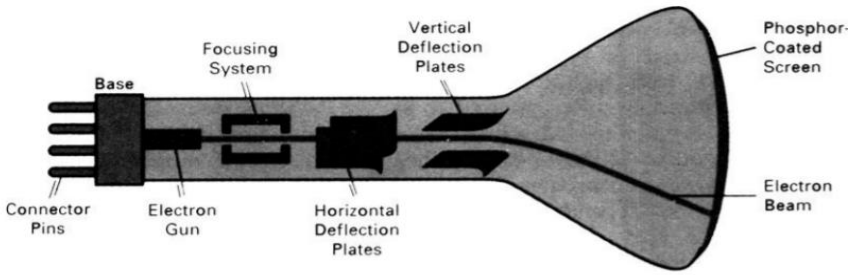


Fig. 5.Cathode ray tube(CRT)

The focusing system in a CRT is needed to force the electron beam to converge into a small spot as it strikes the phosphor. Otherwise, the electrons would repel each other, and the beam would spread out as it approaches the screen. Focusing is accomplished with either electric or magnetic fields. Electrostatic focusing is commonly used in television and computer graphics monitors. With electrostatic focusing, the electron beam passes through a positively charged metal cylinder that forms an electrostatic lens, as shown in Fig. 2.3. Similar lens focusing effects can be accomplished with a magnetic field set up by a coil mounted around the outside of the CRT envelope. Magnetic lens focusing produces the smallest spot size on the screen and is used in special-purpose devices.

The CRT envelope, as illustrated in Fig. 2.1. Two pairs of coils are used, with the coils in each pair mounted on opposite sides of the neck of the CRT envelope. One pair is mounted on the top and bottom of the neck, and the other pair is mounted on opposite sides of the neck. The magnetic field produced by each pair of coils results in a transverse deflection force that is perpendicular both to the direction of the magnetic field and to the direction of travel of the electron beam. Horizontal deflection is accomplished with one pair of coils, and vertical deflection by the other pair. The proper deflection amounts are attained by adjusting the current through the coils. When electrostatic deflection is used, two pairs of parallel plates are mounted inside the CRT envelope. Spots of light are produced on the screen by the transfer of the CRT beam energy to the phosphor. When the electrons in the beam collide with the phosphor coating, they are stopped and their kinetic energy is absorbed by the phophor. Part of the beam energy is converted by friction into heat energy, and the remainder causes electrons in the phosphor atoms to move up to higher quanturn-energy levels. After a short time, the "excited" phosphor electrons begin dropping back to their stable ground state, giving up their extra energy as small quantums of light energy. What we see on the screen is the combined effect of all the electron light emissions: a glowing spot that quickly fades after all the excited phosphor electrons have returned to their ground energy level. The frequency (or color) of the light emitted by the phosphor is proportional to the energy difference between the excited quantum state and the ground state.

Figure 5 shows the intensity distribution of a spot on the screen. The intensity is greatest at the center of the spot, and decreases with a Gaussian distribution out to the edges of the spot. This distribution corresponds to the cross-sectional electron density distribution of the CRT beam.

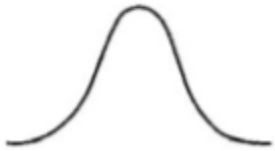


Fig.6. Intensity distribution of an illuminated phosphor spot on a CRT screen

Resolution The maximum number of points that can be displayed without overlap on a CRT is referred to as the resolution. A more precise definition of resolution is the number of points per centimeter that can be plotted horizontally and vertically, although it is often simply stated as the total number of points in each direction. This depends on the type of phosphor used and the focusing and deflection system.

Aspect Ratio Another property of video monitors is aspect ratio. This number gives the ratio of vertical points to horizontal points necessary to produce equal-length lines in both directions on the screen. (Sometimes aspect ratio is stated in terms of the ratio of horizontal to vertical points.) An aspect ratio of 3/4 means that a vertical line plotted with three points has the same length as a horizontal line plotted with four points. Random-Scan and Raster Scan Monitor.

Random-Scan/Calligraphic displays Random scan system uses an electron beam which operates like a pencil to create a line image on the CRT. The image is constructed out of a sequence of straight line segments. Each line segment is drawn on the screen by directing the beam to move from one point on screen to the next, where each point is defined by its x and y coordinates. After drawing the picture, the system cycles back to the first line and design all the lines of the picture 30 to 60 time each second. When operated as a random-scan display unit, a CRT has the electron beam directed only to the parts of the screen where a picture is to be drawn. Random-scan monitors draw a picture one line at a time and for this reason are also referred to as vector displays (or stroke- writing or calligraphic displays)

Refresh rate on a random-scan system depends on the number of lines to be displayed. Picture definition is now stored as a set of line-drawing commands in an area of memory referred to as the refresh display file. Random-scan systems are designed for line-drawing applications and can-not display realistic shaded scenes. Since picture definition is stored as a set of line-drawing instructions and not as a set of intensity values for all screen points, vector displays generally have higher resolution than raster systems. Also, vector displays produce smooth line drawings because the CRT beam directly follows the line path.

Raster-Scan Displays

In raster scan approach, the viewing screen is divided into a large number of discrete phosphor picture elements, called pixels. The matrix of pixels constitutes the raster. The number of separate pixels in the raster display might typically range from 256X256 to 1024X 1024. Each pixel on the screen can be made to glow with a different brightness. Colour screen provide for the pixels to have different colours as well as brightness. In a raster-scan system, the electron beam is swept across the screen, one row at a time from top to bottom. As the electron beam moves across each row, the beam intensity is turned on and off to create a pattern of illuminated spots. Picture definition is stored in a memory area called the refresh buffer or frame buffer. This memory area holds the set of intensity values for all the screen points. Stored intensity values are then retrieved from the refresh buffer and "painted" on the screen one row (scan line) at a time (Fig. 2.6). Each screen point is referred to as a pixel or pel (shortened forms of picture element). The capability of a raster-scan system to 9 store intensity information for each screen point makes it well suited for the realistic display of scenes containing subtle shading and color patterns. Home television sets and printers are examples of other systems using raster-scan methods.

Intensity range for pixel positions depends on the capability of the raster system. In a simple black-and-white system, each screen point is either on or off, so only one bit per pixel is needed to control the intensity of screen positions. Interlacing of the scan lines in this way allows us to see the entire screen displayed in one-half the time it would have taken to sweep across all the lines at once from top to bottom. Interlacing is primarily used with slower refreshing rates. On an older, 30 frame- per-second, noninterlaced display, for instance, some flicker is noticeable. But with interlacing, each of the two passes can be accomplished in l/60th of a second, which brings the refresh rate nearer to 60 frames per second. This is an effective technique for avoiding flicker, providing that adjacent scan lines contain similar display information.

APIs

In application programming interface (API) is a connection between computers or between computer programs. It is a type of software interface, offering a service to other pieces of software.[1] A document or standard that describes how to build such a connection or interface is called an API specification. A computer system that meets this standard is said to implement or expose an API. The term API may refer either to the specification or to the implementation.

In contrast to a user interface, which connects a computer to a person, an application programming interface connects computers or pieces of software to each other. It is not intended to be used directly by a person (the end user) other than a computer programmer who is incorporating it into software. An API is often made up of different parts which act as tools or services that are available to the programmer. A program or a programmer that uses one of these parts is said to call that portion of the API. The calls that make up the API are also known as subroutines, methods, requests, or endpoints. An API specification defines these calls, meaning that it explains how to use or implement them.

One purpose of APIs is to hide the internal details of how a system works, exposing only those parts a programmer will find useful and keeping them consistent even if the internal details later change. An API may be custom-built for a particular pair of systems, or it may be a shared standard allowing interoperability among many systems.

TEXT / REFERENCE BOOKS

1. Donald D Hearn, M. Pauline Baker, Computer Graphics C version, Pearson Education.
2. Dave Shreiner, Mason Woo, Jackie Neider, Tom Davis, OpenGL Programming Guide: The Official Guide to Learning OpenGL, (2013).
3. James D. Foley, Andries van Dam, Steven K. Feiner, John F. Hughes, Computer Graphics: Principles & Practice in C, Addison Wesley Longman.
4. Zhigang Xiang, Roy A Plastock, Computer Graphics, Schaums Outline, TMH.

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

SCHOOL OF ELECTRICAL AND ELECTRONICS

DEPARTMENT OF ELECTRONICS AND COMMMUNICATIONENGINEERING

UNIT – IV

SCSA3007- INTRODUCTION TO VISUAL COMPUTING

UNIT 4 REPRESENTATION OF GEOMETRY

Parametric Curves-Bézier Curves-B-Splines-NURBS- Tensor Product Surfaces, Triangle Meshes-Subdivision Methods-Shape Models, linear (Gaussian) Diffusion.

In computer graphics, we often need to draw different types of objects onto the screen. Objects are not flat all the time and we need to draw curves many times to draw an object.

Types of Curves

A curve is an infinitely large set of points. Each point has two neighbors except endpoints. Curves can be broadly classified into three categories − explicit, implicit, and parametric curves.

3 basic representation strategies:
Explicit: $y = mx + b$
Implicit: $ax + by + c = 0$
Parametric: $P = P0 + t\ (P1 - P0)$

Implicit Curves

Implicit curve representations define the set of points on a curve by employing a procedure that can test to see if a point in on the curve. Usually, an implicit curve is defined by an implicit function of the form −

Fx+y+,y = 0

It can represent multivalued curves multiple y values for an x value multiple y values for an x value. A common example is the circle, whose implicit representation is

$$\mathbf{x^2 + y^2 - R^2 = 0}$$

Explicit Curves

A mathematical function y = fxx can be plotted as a curve. Such a function is the explicit representation of the curve. The explicit representation is not general, since it cannot represent vertical lines and is also single-valued. For each value of x, only a single value of y is normally computed by the function.

Advantages of parametric forms

- More degrees of freedom
- Directly transformable
- Dimension independent
- No infinite slope problems
- Separates dependent and independent variables
- Inherently bounded
- Easy to express in vector and matrix form
- Common form for many curves and surfaces

Algebraic Representation

- All of these curves are just parametric algebraic polynomials expressed in different bases
- Parametric linear curve (in E^3)

$$\mathbf{p}(u) = \mathbf{a}u + \mathbf{b}$$

$$x = a_x u + b_x$$
$$y = a_y u + b_y$$
$$z = a_z u + b_z$$

- Parametric cubic curve (in E^3)

$$\mathbf{p}(u) = \mathbf{a}u^3 + \mathbf{b}u^2 + \mathbf{c}u + \mathbf{d}$$

$$x = a_x u^3 + b_x u^2 + c_x u + d_x$$
$$y = a_y u^3 + b_y u^2 + c_y u + d_y$$
$$z = a_z u^3 + b_z u^2 + c_z u + d_z$$

- Basis (monomial or power)

$$\begin{bmatrix} u & 1 \end{bmatrix}$$
$$\begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix}$$

BEZIER CURVES

Bezier curve is discovered by the French engineer **Pierre Bézier**. These curves can be generated under the control of other points. Approximate tangents by using control points are used to generate curve. The Bezier curve can be represented mathematically as −

$$\sum_{k=0}^{n} P_i B_i^n(t)$$

Where p_i is the set of points and $B_i^n(t)$ represents the Bernstein polynomials which are given by −

$$B_i^n(t) = \binom{n}{i}(1-t)^{n-i}t^i$$

Where **n** is the polynomial degree, **i** is the index, and **t** is the variable.

The simplest Bézier curve is the straight line from the point P0P0 to P1P1. A quadratic Bezier curve is determined by three control points. A cubic Bezier curve is determined by four control points.

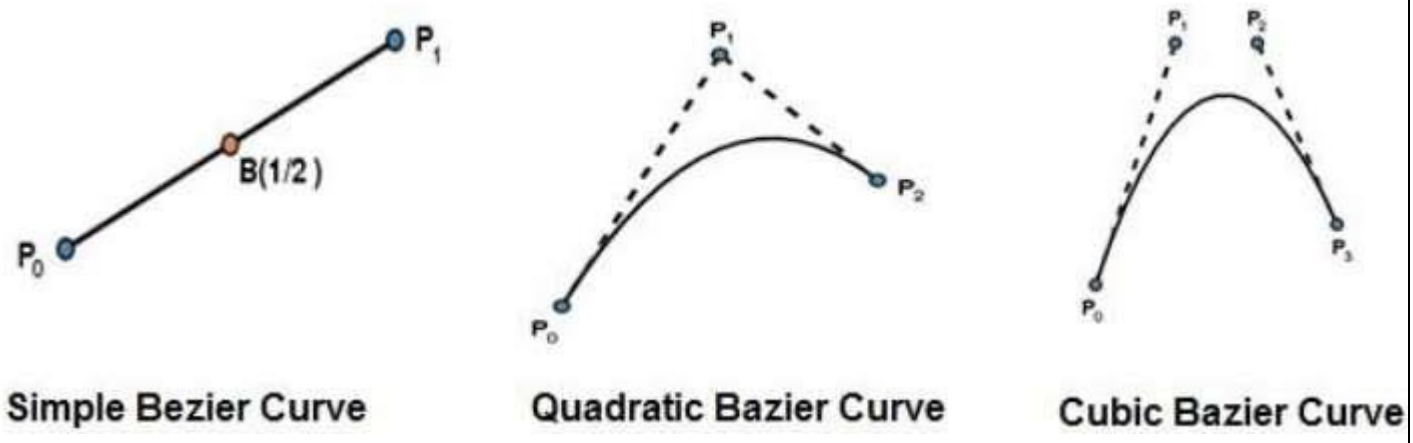


Fig 4.1 Types of Bezier Curves

Properties of Bezier Curves

Bezier curves have the following properties −

- They generally follow the shape of the control polygon, which consists of the segments joining the control points.
- They always pass through the first and last control points.
- They are contained in the convex hull of their defining control points.
- The degree of the polynomial defining the curve segment is one less that the number of defining polygon point. Therefore, for 4 control points, the degree of the polynomial is 3, i.e. cubic polynomial.
- A Bezier curve generally follows the shape of the defining polygon.

- The direction of the tangent vector at the end points is same as that of the vector determined by first and last segments.
- The convex hull property for a Bezier curve ensures that the polynomial smoothly follows the control points.
- No straight line intersects a Bezier curve more times than it intersects its control polygon.
- They are invariant under an affine transformation.
- Bezier curves exhibit global control means moving a control point alters the shape of the whole curve.
- A given Bezier curve can be subdivided at a point t=t0 into two Bezier segments which join together at the point corresponding to the parameter value t=t0.

B-Spline Curves

The Bezier-curve produced by the Bernstein basis function has limited flexibility.

- First, the number of specified polygon vertices fixes the order of the resulting polynomial which defines the curve.
- The second limiting characteristic is that the value of the blending function is nonzero for all parameter values over the entire curve.

The B-spline basis contains the Bernstein basis as the special case. The B-spline basis is non-global.

A B-spline curve is defined as a linear combination of control points Pi and B-spline basis function $N_{i,}$ k t given by

$$C(t) = \sum_{i=0}^{n} P_i N_{i,k}(t), \qquad n \geq k-1, \qquad t\,\epsilon\,[tk-1, tn+1]$$

Where,

- { p_i : i=0, 1, 2....n} are the control points
- k is the order of the polynomial segments of the B-spline curve. Order k means that the curve is made up of piecewise polynomial segments of degree k - 1,
- the $N_{i,k}(t)$ are the "normalized B-spline blending functions". They are described by the order k and by a non-decreasing sequence of real numbers normally called the "knot sequence".

$$t_i : i = 0, \ldots n + K$$

Properties of B-spline Curve

B-spline curves have the following properties −

- The sum of the B-spline basis functions for any parameter value is 1.
- Each basis function is positive or zero for all parameter values.
- Each basis function has precisely one maximum value, except for k=1.
- The maximum order of the curve is equal to the number of vertices of defining polygon.
- The degree of B-spline polynomial is independent on the number of vertices of defining polygon.
- B-spline allows the local control over the curve surface because each vertex affects the shape of a curve only over a range of parameter values where its associated basis function is nonzero.
- The curve exhibits the variation diminishing property.
- The curve generally follows the shape of defining polygon.
- Any affine transformation can be applied to the curve by applying it to the vertices of defining polygon.
- The curve line within the convex hull of its defining polygon.

NURBS (NonUniform Rational B-Splines)

NURBS are mathematical representations of 2- or 3-dimensional objects, which can be standard shapes (such as a cone) or free-form shapes (such as a car). NURBS are used in computer graphics and the CAD/CAM industry and have come to be regarded as a standard way to create and represent complex objects. In addition to curves and surfaces, NURBS can also represent hypersurfaces.

Most sophisticated graphic creation tools provide an interface for using NURBS, which are flexible enough to design a wide range of shapes - anything from points to straight lines to conic sections. NURBS are compact expressions that can be evaluated and displayed quickly. NURBS work especially well in 3-D modeling, allowing the designer to easily manipulate control vertices, called ISO curves, and control curvature and the smoothness of contours. NURBS are defined by both control points and weights. It takes very little data to define a NURB.

A *spline* is a usually curvy pattern used to guide someone shaping something large, such as a boat hull. The *B-spline* is based (the B stands for *basis*) on four local functions or control points that lie outside the curve itself. *Nonuniform* is the idea that some sections of a defined shape (between any two points) can be shortened or elongated relative to other sections in the overall shape. *Rational* describes the

ability to give more weight to some points in the shape than to other points in considering each positions relation to another object. (This is sometimes referred to as a 4th dimensional characteristic.)

Basic idea: four dimensional non-uniform B-splines, followed by normalization via homogeneous coordinates – If Pi is [x, y, z, 1], results are invariant with respect to perspective projection.

To say that a curve is "Rational" means that it maybe described in homogenous coordinate system and this has as a consequence that they may be subject to the usual transformations without loosing their form.

To say that a curve is "Rational" means that it maybe described in homogenous coordinate system and this has as a consequence that they may be subject to the usual transformations without loosing their form.

A nonuniform rational B-spline curve defined by

$$\mathbf{C}(t) = \frac{\sum_{i=0}^{n} N_{i,p}(t)\, w_i\, \mathbf{P}_i}{\sum_{i=0}^{n} N_{i,p}(t)\, w_i},$$

where p is the order, $N_{i,p}$ are the B-spline basis functions, $\mathbf{P}_i$ are control points, and the weight w_i of $\mathbf{P}_i$ is the last ordinate of the homogeneous point $\mathbf{P}_i^w$. These curves are closed under perspective transformations and can represent conic sections exactly.

$$N_{i,p}(u) = \frac{u - u_i}{u_{i+p} - u_i} N_{i,p-1}(u) + \frac{u_{i+p+1} - u}{u_{i+p+1} - u_{i+1}} N_{i-1,p-1}(u)\ .$$

Where

$$N_{i,0}(u) = \begin{cases} 1 & \text{if} \quad u_i \leqslant u < u_i + 1 \\ 0 & \text{else} \end{cases}$$

As a consequence, the basis functions become,

$$R_{i,p}(u) = \frac{N_{i,p}(u) w_i}{\sum_{i=0}^{n} N_{i,p}(u) w_i}\ .$$

and

$$C(u) = \sum_{i=0}^{n} R_{i,p}(u) P_i\ .$$

Ri,p(u) is called NURBS basis function and share some properties of B-Spline and Bezier basis function.

Control points

Three-dimensional NURBS surfaces can have complex, organic shapes. Control points influence the directions the surface takes. A separate square below the control cage delineates the X and Y extents of the surface.

The control points determine the shape of the curve.[8] Typically, each point of the curve is computed by taking a weighted sum of a number of control points. The weight of each point varies according to the governing parameter. For a curve of degree d, the weight of any control point is only nonzero in d+1 intervals of the parameter space. Within those intervals, the weight changes according to a polynomial function (*basis functions*) of degree d. At the boundaries of the intervals, the basis functions go smoothly to zero, the smoothness being determined by the degree of the polynomial.

As an example, the basis function of degree one is a triangle function. It rises from zero to one, then falls to zero again. While it rises, the basis function of the previous control point falls. In that way, the curve interpolates between the two points, and the resulting curve is a polygon, which is continuous, but not differentiable at the interval boundaries, or knots. Higher degree polynomials have correspondingly more continuous derivatives. Note that within the interval the polynomial nature of the basis functions and the linearity of the construction make the curve perfectly smooth, so it is only at the knots that discontinuity can arise.

In many applications the fact that a single control point only influences those intervals where it is active is a highly desirable property, known as local support. In modeling, it allows the changing of one part of a surface while keeping other parts unchanged.

Adding more control points allows better approximation to a given curve, although only a certain class of curves can be represented exactly with a finite number of control points. NURBS curves also feature a scalar weight for each control point. This allows for more control over the shape of the curve without unduly raising the number of control points. In particular, it adds conic sections like circles and ellipses to the set of curves that can be represented exactly. The term *rational* in NURBS refers to these weights.

The control points can have any dimensionality. One-dimensional points just define a scalar function of the parameter. These are typically used in image processing programs to tune the brightness and color curves. Three-dimensional control points are used abundantly in 3D modeling, where they are used in the everyday meaning of the word 'point', a location in 3D space. Multi-dimensional points might be used to control sets of time-driven values, e.g. the different positional and rotational settings of a robot arm. NURBS surfaces are just an application of this. Each control 'point' is actually a full vector of control points, defining a curve. These curves share their degree and the number of control points, and span one dimension of the parameter space. By interpolating these control vectors over the other dimension of the parameter space, a continuous set of curves is obtained, defining the surface.

Knot vector

The knot vector is a sequence of parameter values that determines where and how the control points affect the NURBS curve. The number of knots is always equal to the number of control points plus curve degree plus one (i.e. number of control points plus curve order). The knot vector divides the parametric space in the intervals mentioned before, usually referred to as *knot spans*. Each time the parameter value enters a new knot span, a new control point becomes active, while an old control point is discarded. It follows that the values in the knot vector should be in nondecreasing order, so (0, 0, 1, 2, 3, 3) is valid while (0, 0, 2, 1, 3, 3) is not.

Consecutive knots can have the same value. This then defines a knot span of zero length, which implies that two control points are activated at the same time (and of course two control points become deactivated). This has impact on continuity of the resulting curve or its higher derivatives; for instance, it allows the creation of corners in an otherwise smooth NURBS curve. A number of coinciding knots is sometimes referred to as a

knot with a certain multiplicity. Knots with multiplicity two or three are known as double or triple knots. The multiplicity of a knot is limited to the degree of the curve; since a higher multiplicity would split the curve into disjoint parts and it would leave control points unused. For first-degree NURBS, each knot is paired with a control point.

The knot vector usually starts with a knot that has multiplicity equal to the order. This makes sense, since this activates the control points that have influence on the first knot span. Similarly, the knot vector usually ends with a knot of that multiplicity. Curves with such knot vectors start and end in a control point.

The values of the knots control the mapping between the input parameter and the corresponding NURBS value. For example, if a NURBS describes a path through space over time, the knots control the time that the function proceeds past the control points. For the purposes of representing shapes, however, only the ratios of the difference between the knot values matter; in that case, the knot vectors (0, 0, 1, 2, 3, 3) and (0, 0, 2, 4, 6, 6) produce the same curve. The positions of the knot values influence the mapping of parameter space to curve space. Rendering a NURBS curve is usually done by stepping with a fixed stride through the parameter range. By changing the knot span lengths, more sample points can be used in regions where the curvature is high. Another use is in situations where the parameter value has some physical significance, for instance if the parameter is time and the curve describes the motion of a robot arm. The knot span lengths then translate into velocity and acceleration, which are essential to get right to prevent damage to the robot arm or its environment. This flexibility in the mapping is what the phrase *non uniform* in NURBS refers to.

Necessary only for internal calculations, knots are usually not helpful to the users of modeling software. Therefore, many modeling applications do not make the knots editable or even visible. It's usually possible to establish reasonable knot vectors by looking at the variation in the control points. More recent versions of NURBS software (e.g., Autodesk Maya and Rhinoceros 3D) allow for interactive editing of knot positions, but this is significantly less intuitive than the editing of control points.

Tensor Product Surfaces

Tensor product surfaces are one of the most common surfaces encountered in CAGD. Some simple versions can be computed with matrices. This section only gives an overview, leaving the details with regard to some important special cases for subsequent sections.

Consider a curve

$$\mathbf{p}(u) = \sum_{i=0}^{m} \mathbf{f}_i(u)\mathbf{p}_i,$$

where the $\mathbf{f_i(u)}$ are basis functions, and we treat the pi as a 1-parameter family of vector-valued functions

$$\mathbf{p}_i(v) = \sum_{j=0}^{n} \mathbf{g}_j(v)\mathbf{p}_{ij}$$

Tensor product surfaces can be regarded as “curves of curves,” that is, a curve c_1 is moved along a second curve c_2 and all points that are touched when carrying out this movement belong to the surface defined by c_1 and c_2.

In mathematical terms, we denote a piecewise polynomial curve $F(u)$ of degree n by

$$F(u) = \sum_{i=0}^{n} C_i N_i(u) \text{undefined} u \in [0,\ 1]$$

and a second piecewise polynomial curve $G(v)$ of degree m by

$$G(v) = \sum_{j=0}^{m} C_j N_j(v) \text{undefined} v \in [0,\ 1]$$

The tensor product $S(u, v)$ of these two curves is then

$$S(u,v) = \sum_{i=0}^{n} \sum_{j=0}^{m} C_{ij} N_i(u) N_j(v) \text{undefined} u, v \in [0, \text{undefined} 1]$$

It describes a surface over the given control vertices C_{ij}. Equation (7.1) can also be rewritten as

$$S(u,v) = \sum_{i=0}^{n} N_i(u) C_i(v) \quad \text{with} \quad C_i(v) = \sum_{j=0}^{m} N_j(v) C_{ij}$$

which nicely demonstrates the concept of curves of curves. In all these equations, the C_i, C_j, and C_{ij} are the control vertices of the surface, while $N_i(u)$ and $N_j(v)$ are the base function of the polynomials of degree n and m, respectively.

To represent such a surface as a geometric model by means of data structures and algorithms, similar considerations for polygonal models can be made. Since the control vertices describe the surface, they have to be present in the model. While in polygonal models the connectivity between the vertices is denoted explicitly by edges and polygonal faces, for free-form surfaces this connectivity is established via the interpolation scheme being used for computing points on the curve (or surface). Thus, unlike for polygon meshes, part of the model is represented in the form of algorithms, namely, as the definition of the basis functions.

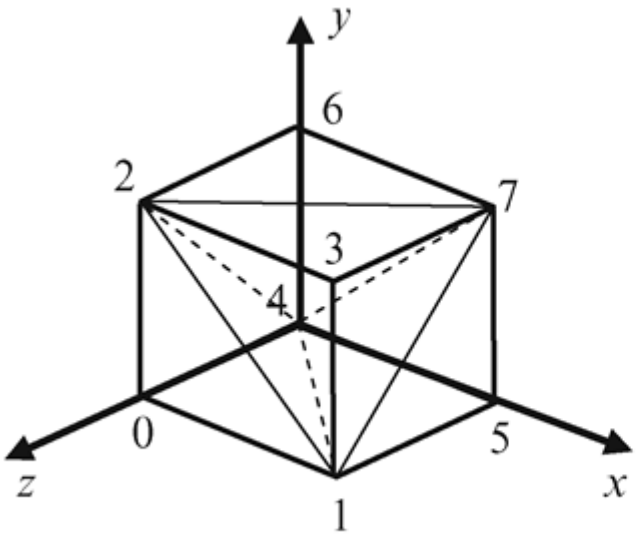
Triangle Meshes

In computer graphics applications, three-dimensional models are almost always represented using polygonal meshes. A mesh in its simplest form consists of a set of vertices, polygons, and optionally a number of additional vertex and polygonal attributes. The complexity of a mesh can vary from low to very high depending on requirements such as rendering quality, speed and resolution. A wide spectrum of mesh processing algorithms is used by graphics and game developers for a variety of applications such as generating, simplifying, smoothing, remapping and transforming meshes. Several types of data structures and file formats are also used to store mesh data.

Mesh Representation

A polygonal mesh is a set of vertices and polygonal elements that collectively define a three-dimensional geometrical shape. The simplest mesh representation thus consists of a vertex list and a polygon list as shown in Fig. 8.1. Polygons are often defined in terms of triangular elements. Since triangles are always both planar and convex, they can be conveniently used in several geometrical computations such as point inclusion tests, area and normal calculations and interpolation of vertex attributes.

The vertex list contains the three-dimensional coordinates of the mesh vertices defined in a suitable coordinate frame, and the polygon list contains integer values that index into the vertex list. An anticlockwise ordering of vertices with respect to the outward face normal direction is commonly used to indicate the front facing side of each polygon. The distinction between the front and the back faces of a polygon becomes important in lighting computations and culling operations. If the polygon list represents a set of connected triangles as in Fig. 4.1, a more efficient and compact data structure called a triangle strip may be used. The first three indices in a triangle strip specify the first triangle. The fourth index along with the previous two indices represents the second triangle. In this fashion, each remaining index represents a triangle that is defined by that index and the previous two indices.



Vertex List		
x	y	z
0.0	0.0	1.0
1.0	0.0	1.0
0.0	1.0	1.0
1.0	1.0	1.0
0.0	0.0	0.0
1.0	0.0	0.0
0.0	1.0	0.0
1.0	1.0	0.0

Triangle List		
i	j	k
0	1	2
1	3	2
2	3	7
2	7	6
1	7	3
1	5	7
6	7	4
7	5	4
0	4	1
1	4	5
2	6	4
0	2	4

Fig. 4.2 A cube and its mesh definition using vertex and polygon lists

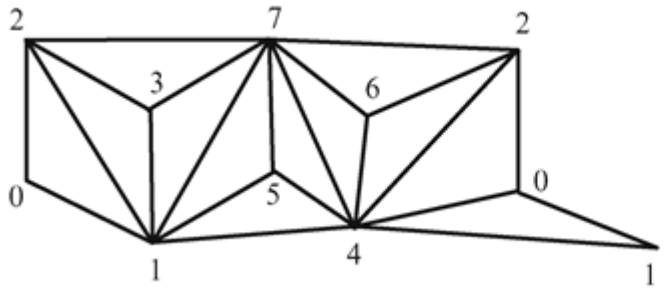


Fig. 4.3 The cut-open view of the cube in

Triangle Strip													
0	1	2	3	7	1	5	4	7	6	2	4	0	1

Fig. 4.4 showing its representation as a triangle strip

The representation of a cube as a triangle strip is given in Fig. 4.4. The triangle strip is decoded as the set of 12 triangles {012, 123, 237, 371, 715, 154, 547, 476, 762, 624, 240, 401 g. Note that the orientation of triangles alternates between clockwise and anticlockwise in this representation. The change of orientation is corrected

by reversing the direction of every alternate triangle in the list, starting from the second triangle. Thus the above list would be correctly interpreted as {012, 213,237,731,715,514, 547,746,762,264, 240, 041}. If the first triangle is defined in the anticlockwise sense, then all triangles in the corrected list will have the same orientation.

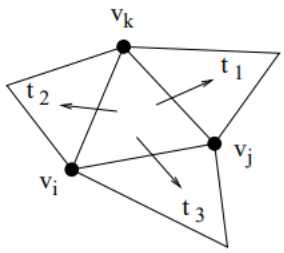
Several file formats are used in graphics applications for storing and sharing mesh data. A number of such file formats represent values in binary and compressed forms for minimizing storage space. In this section, we review some of the popular ASCII file formats that allows easy viewing and editing of mesh data. The Object (.OBJ) format was developed by Wavefront technologies. This format allows the definition of vertices in terms of either three-dimensional Cartesian coordinates or four dimensional homogeneous coordinates. Polynomials can have more than three vertices.

The class TriangleMesh would consist of two arrays, one of Triangle's and the other of Node's. The following diagram indicates that Triangle has (three) pointers to Node.

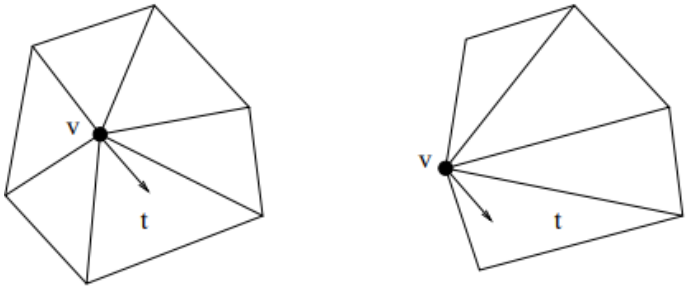
Triangle ⟶ Node

This data structure is fine for simple visualization of the triangles. However, it is not good when we need neighbourhood information.

We might for example wish to estimate normals at the vertices for use in Gouraud shading. This requires finding all triangles which contain a give vertex. The only way to find these triangles is to check all triangles in the triangle array. Thus if there are N triangles, finding one neighbourhood costs O(N) time. Repeating this for each node, and since the number of nodes is about N/2, we end up with an $O(N^2)$ algorithm for estimating all the normals. For N in the region of 10^6 this is not an option. The solution is to enrich the data structure by adding further pointers. One possibility is to add three pointers to the neighbouring triangles in the Triangle class.

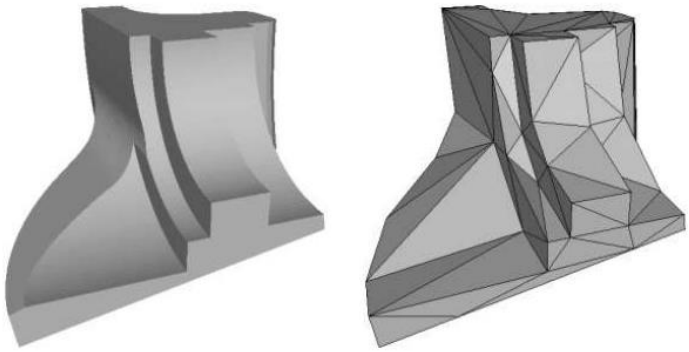


If we traverse the vertices by traversing the triangles, and for each triangle checking which of the three vertices have been visited, we can use the three triangle pointers to find all incident triangles to each vertex. Thus, for example, all normals in the mesh can now be estimated in O(N) time. Note that when building this enhanced data structure from the original, all the triangle pointers can be found in O(N) time. A further help to traversing vertices is to add pointers to the Node class. One can simply add a list of pointers to all incident triangles. Another option which uses less space is to add just one pointer to one of the containing triangles.



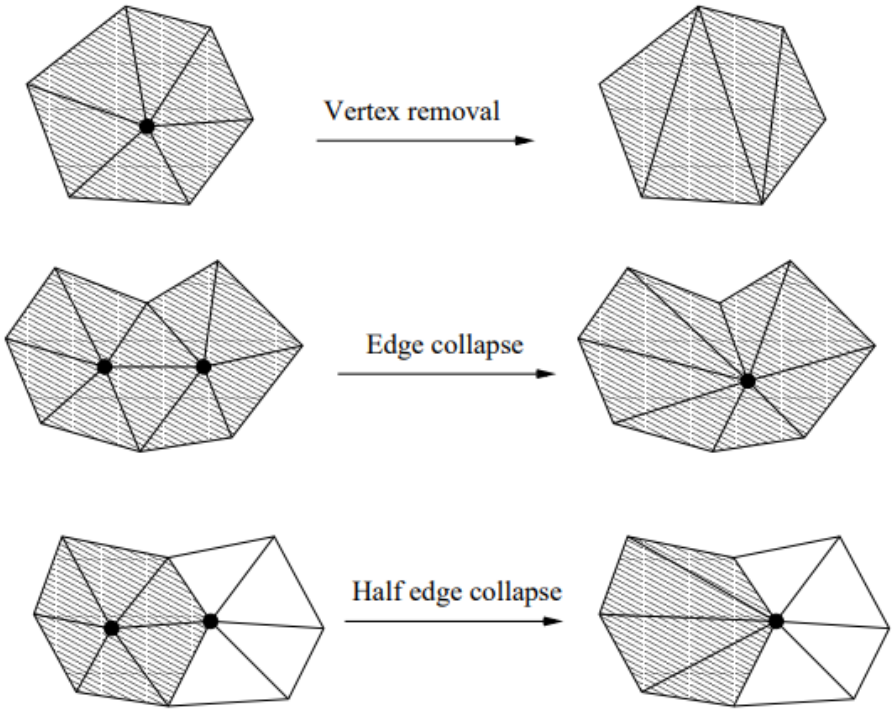
Mesh simplification

Frequently we encounter triangle meshes with very large numbers of triangles and vertices; these data sets often come from scanning millions of points from real objects. It is therefore important to be able to reduce or simplify the mesh if necessary.



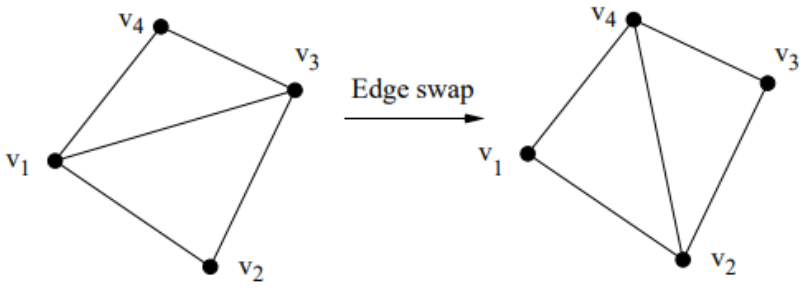
A common way to simplify a mesh is to apply an incremental algorithm, removing one vertex at a time and repairing the hole left by the removal. Ideally, we want to remove as many vertices as we can so that the remaining coarse mesh is still a good enough approximation to the original fine mesh.

There are three common ways to remove a vertex and repair the hole: vertex removal, edge collapse and half-edge collapse.



Mesh optimization

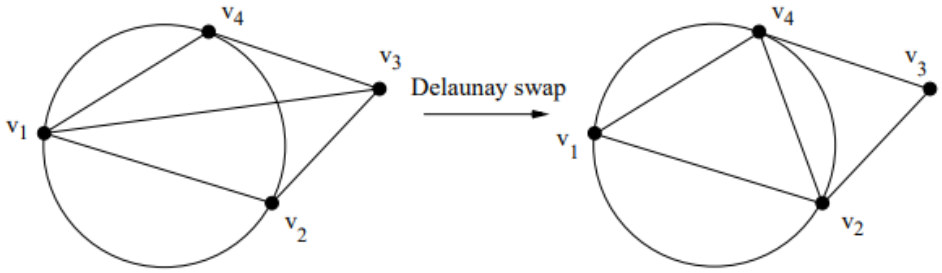
Mesh optimization is conceptually simpler than mesh simplification. The idea is simple. Without changing the vertices, can we change the connectivity of the mesh in order to achieve a better quality surface? The usual approach is edge swapping. Each pair of triangles sharing a common edge form a quadrilateral. The simplest change in connectivity we can make is to swap the given diagonal of a quadrilateral with the other one. In the figure below we swap the edge [v1, v3] with [v2, v4]. This has the effect of replacing the two triangles [v1, v2, v3] and [v1, v3, v4] by [v1, v2, v4] and [v2, v3, v4].



As with vertex removal, care must be taken not to invalidate the mesh by creating a non-simple graph. Thus if v2 and v4 on the left are already connected by an edge outside the quadrilateral, we cannot perform the swap.

By applying several edge swaps we gradually change the original mesh into a better one. We choose some cost function we wish to minimize and the swap criterion is then simply whether the swap decreases the cost function. We keep on swapping edges which result in a decreased cost function, until no further decreases are possible. Usually it is not possible to guarantee that the global minimum can be reached by a sequence of

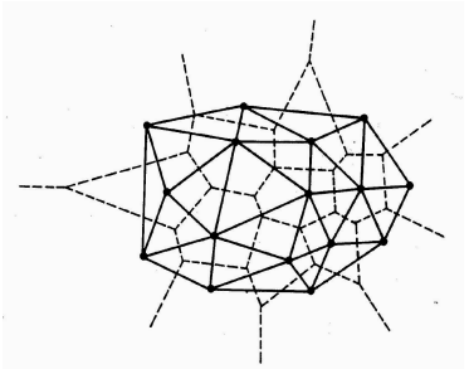
swaps, but the result is often a better mesh anyway. For planar triangle meshes, there is a very well known swap criterion, called the Delaunay criterion. We swap the edge [v1, v3] (of a convex quadrilateral) if the vertex v3 lies outside the circumcircle of the triangle [v1, v2, v4], as is in the following figure.



This swapping procedure was proposed by Lawson and it is equivalent to the max-min angle criterion; we swap if the minimum of the six angles in the two triangles is increased. The beauty of the Delaunay swap criterion is that it always leads to a unique triangulation of the planar points (at least if there are no cocircular points). When no more swaps can be made, the triangle mesh is a Delauanay triangulation of the plane, i.e., a triangulation in which the interior 15 of the circumcircle of each triangle is empty (contains no other vertices of the triangulation). A Delaunay triangulation has the property that the minimum of all the angles of its triangles is maximized. Thus the Delauany swap criterion tends to give 'well-shaped' triangles where possible. Delaunay triangulation have several nice properties. A Delauanay triangulation of a set of planar points is the dual graph of their Voronoi diagram. The Voronoi diagram of a set of planar points p1, . . . , pN is a collection of tiles. There is one tile Vi associated with each point pi . The i-th tile Vi is simply the set of all points in lR 2 that are closer to pi than any other point pj , i.e.,

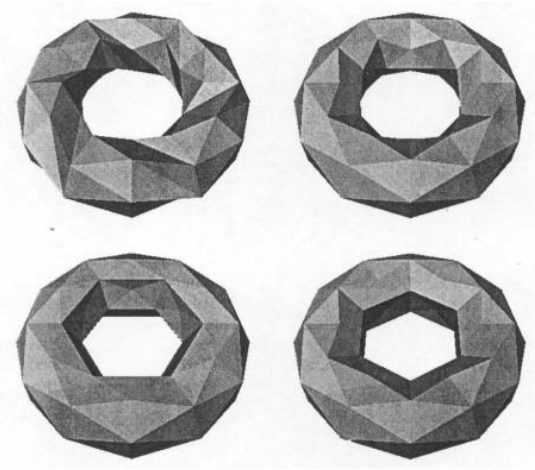
$$V_i = \{x \in \mathbb{R}^2 : \|x - p_i\| \leq \|x - p_j\| \ \forall j \neq i\}.$$

The figure below shows the Delauanay triangulation (solid) of a set of planar points and their Voronoi diagram (dashed).



For triangle meshes in lR ^{3} the circumcircle criterion no longer makes sense, and though the max-min angle criterion could be used, a unique solution is no longer guaranteed. In fact in the lR ^{3} case we often use optimization criteria which reflect the geometry of the surface. For example we might optimize the 'smoothness' of the mesh, by minimizing the angle between the normal directions of adjacent triangles, or by

minimizing some discrete measure of curvature. The figure below show various optimizations of a given toroidal-shaped triangle mesh.



Area Subdivison Algorithm in 3D(Hidden Surface Removal)

Mr. John Warnock proposed an area subdivision algorithm, that's why also known as the Warnock algorithm. This algorithm extensively uses the concept of area coherence in computing the visible surface in the scene, which is closer to the viewing plane, area coherence avoids the computation of the visibility detection of the common surface, that has already been computed in the earlier step, so no need to recompute it. That's all the area coherence does.

The area subdivision algorithm is based on the divide and conquers method where the visible(viewing) area is successively divided into smaller and smaller rectangles until the simplified area is detected. Consider given with the window panel, where the polygon will be projected, is as follows:

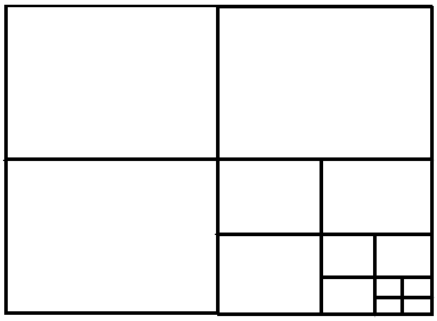
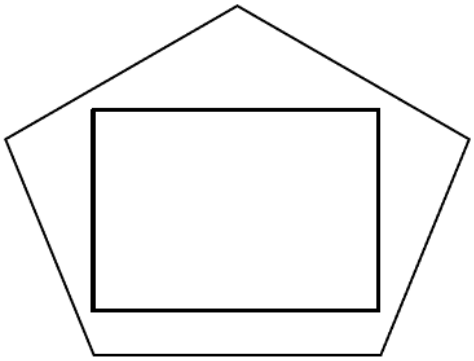


Fig.4.6 subdivide the window pane in rectangles of equal area

When we subdivide the window panel against the polygon we may come through the following cases which are as follows:

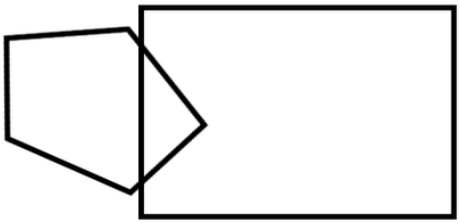
1. Surrounding surface: It's the case in which the viewing polygon surface completely surrounds the whole window panel.



Polygon has circumscribed the window panel

Fig.4.7

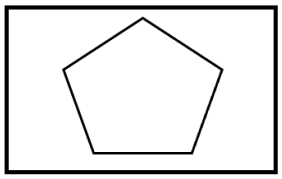
2. Overlapping(Intersecting) surface: It's the case in which the window panel(viewport) and viewing polygon surface both intersect each other.



Both window panel and polygon are overlapping each others surface

Fig.4.8

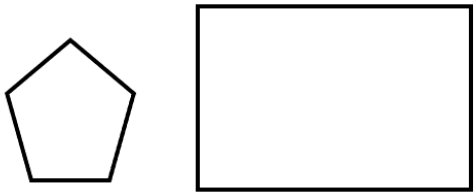
3. Inside(contained) surface: In this case, in which the whole polygon surface is inscribed inside the window panel. This case is just opposite to the first(surrounding surface) case.



Complete polygon has inscribed inside the window panel

Fig.4.9

4. Outside(disjoint) surface: In this case, the whole polygon surface is completely outside the window panel.



Polygon is completely outside of the window panel

Fig.4.10

Algorithm:

1. Initialize the viewing area or window panel dimension.
2. Enlist all the polygon(s) and sort them according to Z_{min} (depth value) with respect to the window panel(view port).
3. Categorize all the polygon(s) according to their corresponding cases in which they are falling.
4. Now, perform the visible surface(hidden surface removal) detection test:

- If we have a polygon, which has completely surrounded the window panel then set the viewing area color to the corresponding polygon color that is stored in the frame buffer.
- If the enlisted polygon(s) are disjoint or completely outside the window panel then, in this case, the background color of the window plane will be done and the polygon(s) will be ignored.
- If the polygon(s) is contained(inscribed) inside the window panel completely then color the polygon from its corresponding color and color the rest of the surface with the background color.
- If polygon(s) and the window panel surfaces are intersecting(overlapping) each other then the following cases need to be considered:
- Fill the overlapped region with the corresponding polygon color that is set in the frame buffer.
- If we are given more than one polygon, with overlapped surfaces with respect to the widow plane then, in this case, we first find out the depth buffer(Z_{min}) in order to find the surface of the polygon which is closer to the window panel and will fill the overlapped region with the color of that polygon that has minimum(Z_{min}).

5. Repeat all the above steps for all given polygons, then exit.

Shape Models

In 3D computer graphics, 3D modeling is the process of developing a mathematical coordinate-based representation of any surface of an object (inanimate or living) in three dimensions via specialized software by manipulating edges, vertices, and polygons in a simulated 3D space.[1][2][3]

Three-dimensional (3D) models represent a physical body using a collection of points in 3D space, connected by various geometric entities such as triangles, lines, curved surfaces, etc.[4] Being a collection of data (points and other information), 3D models can be created manually, algorithmically (procedural modeling), or by scanning.[5][6] Their surfaces may be further defined with texture mapping.

A **2D geometric model** is a geometric model of an object as a two-dimensional figure, usually on the Euclidean or Cartesian plane.

Even though all material objects are three-dimensional, a 2D geometric model is often adequate for certain flat objects, such as paper cut-outs and machine parts made of sheet metal.

2D geometric models are also convenient for describing certain types of artificial images, such as technical diagrams, logos, the glyphs of a font, etc. They are an essential tool of 2D computer graphics and often used as components of 3D geometric models, e.g. to describe the decals to be applied to a car model. Modern architecture practice "digital rendering" which is a technique used to form a perception of a 2-D geometric model as of a 3-D geometric model designed through descriptive geometry and computerized equipment.[1

Elements of Pictures: Computer graphics is all about producing pictures (realistic or stylistic) by computer. Before discussing how to do this, let us first consider the elements that make up images and the devices that produce them. How are graphical images represented? There are four basic types that make up virtually of

computer generated pictures: polylines, filled regions, text, and raster images. Polylines: A polyline (or more properly a polygonal curve is a finite sequence of line segments joined end to end. These line segments are called edges, and the endpoints of the line segments are called vertices. A single line segment is a special case. (An infinite line, which stretches to infinity on both sides, is not usually considered to be a polyline.) A polyline is closed if it ends where it starts. It is simple if it does not self-intersect. Self-intersections include such things as two edge crossing one another, a vertex intersecting in the interior of an edge, or more than two edges sharing a common vertex. A simple, closed polyline is also called a simple polygon. If all its internal angle are at most 180◦, then it is a convex polygon. A polyline in the plane can be represented simply as a sequence of the (x, y) coordinates of its vertices. This is sufficient to encode the geometry of a polyline. In contrast, the way in which the polyline is rendered is determined by a set of properties call graphical attributes. These include elements such as color, line width, and line style (solid, dotted, dashed), how consecutive segments are joined (rounded, mitered or beveled).

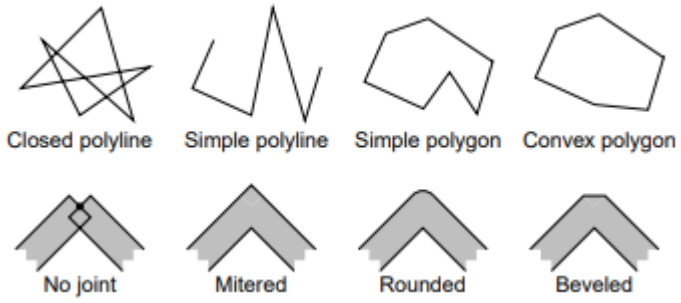


Fig 4.11.Polylines and joint styles.

Many graphics systems support common special cases of curves such as circles, ellipses, circular arcs, and Bezier and B-splines. We should probably include curves as a generalization of polylines. Most graphics drawing systems implement curves by breaking them up into a large number of very small polylines, so this distinction is not very important. Filled regions: Any simple, closed polyline in the plane defines a region consisting of an inside and outside. (This is a typical example of an utterly obvious fact from topology that is notoriously hard to prove. It is called the Jordan curve theorem.) We can fill any such region with a color or repeating pattern. In some instances the bounding polyline itself is also drawn and others the polyline is not drawn.

A polyline with embedded “holes” also naturally defines a region that can be filled. In fact this can be generalized by nesting holes within holes (alternating color with the background color). Even if a polyline is not simple, it is possible to generalize the notion of interior. Given any point, shoot a ray to infinity. If it crosses the boundary an odd number of times it is colored. If it crosses an even number of times, then it is given the background color.

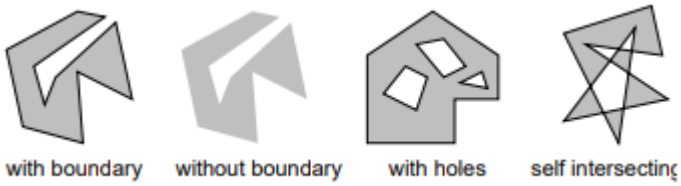


Fig 4.12.Filled regions

Text: Although we do not normally think of text as a graphical output, it occurs frequently within graphical images such as engineering diagrams. Text can be thought of as a sequence of characters in some font. As with polylines there are numerous attributes which affect how the text appears. This includes the font's face (Times-Roman, Helvetica, Courier, for example), its weight (normal, bold, light), its style or slant (normal, italic, oblique, for example), its size, which is usually measured in points, a printer's unit of measure equal to 1/72-inch), and its color

Face (family)	Weight	Style (slant)	Size
Helvetica	Normal	Normal	8 point
Times–Roman	**Bold**	*Italic*	10 point
Courier			12 point

Fig.4.13 : Text font properties.

Linear diffusion filtering

Linear diffusion filtering is probably the best investigated PDE-based method for smoothing images and it has been applied in numerous fields in image processing [11, pp. 11-12]. Its most surprising characteristic is that it is equivalent to Gaussian smoothing. This is very interesting because in some way supports the intuitive idea that Gaussian smoothing "diffuses" (fades out) objects to each other in the image. But at the same time, from this point of view, this method has the same problem as Gaussian filters about loosing edges when passing to coarser levels in a scale-space. However, unlike Gaussian filtering, linear diffusion filtering

Relations to Gaussian smoothing Gaussian smoothing

Let a grey-scale image f be represented by a real-valued mapping $f \in L^1(\mathbb{R}^2)$. A widely-used way to smooth f is by calculating the convolution.

$$(K_\sigma * f)(x) := \int_{\mathbb{R}^2} K_\sigma(x-y)\, f(y)\, dy$$

where Kσ denotes the two-dimensional Gaussian of width (standard deviation) σ>0 :

$$K_\sigma(x) := \frac{1}{2\pi\sigma^2} \cdot \exp\left(-\frac{|x|^2}{2\sigma^2}\right).$$

There are several reasons for the excellent smoothing properties of this method: First we observe that since $K_\sigma \in C^\infty(\mathbb{R}^2)$ we get $K_\sigma * f \in C^\infty(\mathbb{R}^2)$, even if f is only absolutely integrable.

Next, let us investigate the behaviour in the frequency domain. When defining the Fourier transformation $\mathcal{F}$ by

$$(\mathcal{F}f)(\omega) := \int_{\mathbb{R}^2} f(x)\, \exp(-i\langle\omega, x\rangle)\, dx$$

we obtain by the convolution theorem that

$$(\mathcal{F}(K_\sigma * f))(\omega) = (\mathcal{F}K_\sigma)(\omega) \cdot (\mathcal{F}f)(\omega).$$

Since the Fourier transform of a Gaussian is again Gaussian-shaped,

$$(\mathcal{F}K_\sigma)(\omega) = \exp\left(-\frac{|\omega|^2}{2/\sigma^2}\right),$$

a low-pass filter that attenuates high frequencies in a monotone way.

Gaussian derivatives

$$I(x,y,t) = \begin{cases} (K_{\sqrt{2t}} \star f)(x,y) & (t>0) \\ f(x,y) & (t=0) \end{cases}$$

This solution:

(a) Is unique.

(b) Continuously depends on f with respect to $||.||_\infty$.

(c) Fulfills:

$$\inf_{\mathbb{R}^2} f \le I(x,y,t) \le \sup_{\mathbb{R}^2} f \quad \text{on} \quad \mathbb{R}^2 \times [0,\infty)$$

Thus, an image I(x, y) can be regarded as the function f.

$$F_k(I(x,y)) = \begin{cases} I(x,y,k) = (K_{\sqrt{2k}} \star I)(x,y) & (k>0) \\ I(x,y) & (k=0) \end{cases}$$

---------4.1

On the other hand, if you instead would like to solve equation 1 numerically to apply it to an image as a single filter and get a smoothing effect equivalent to a standard deviation σ Gaussian filter effect, you would have to stop the diffusion process at time $T = \sigma^2/2$. The latter does not sound as an attractive idea since expression 4.1 provides an analytical solution, which is suitable to be applied directly, but it gives you an idea of what is happening in terms of the diffusion process in the nonlinear and/or anisotropic case.

$$\frac{\partial I(x,y,t)}{\partial t} = \nabla \cdot (g \nabla I(x,y,t))$$
$$I(x,y,0) = I(x,y)$$

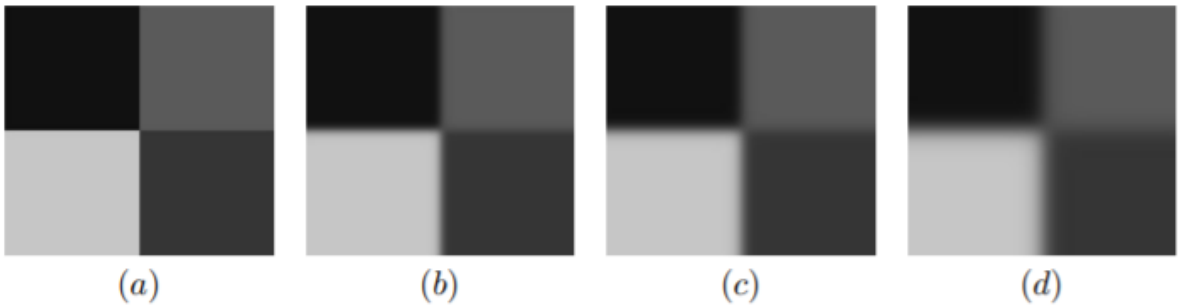


Figure 4.14 : Linear diffusion filtering (σ = 5). (a) Original image, (b) g = 1, (c) g = 2, (d) g = 5. Observe that the blurring level can also be controlled with the diffusivity g.

TEXT / REFERENCE BOOKS

1. Donald D Hearn, M. Pauline Baker, Computer Graphics C version, Pearson Education.
2. Dave Shreiner, Mason Woo, Jackie Neider, Tom Davis, OpenGL Programming Guide: The Official Guide to Learning OpenGL, (2013).
3. James D. Foley, Andries van Dam, Steven K. Feiner, John F. Hughes, Computer Graphics: Principles & Practice in C, Addison Wesley Longman.
4. Zhigang Xiang, Roy A Plastock, Computer Graphics, Schaums Outline, TMH.

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

SCHOOL OF ELECTRICAL AND
ELECTRONICS

DEPARTMENT OF ELECTRONICS AND COMMMUNICATIONENGINEERING

UNIT – V

SCSA3007- INTRODUCTION TO VISUAL COMPUTING

UNIT 5 LEARNING METHODS IN VISION

Classifier Learning-Support Vector Machines-Radial Basis Function Networks-Dimension Reduction: PCA, ICA- Linear Discriminant Analysis-Graphical Models- Markov Random Fields-Maximum Entropy Inference and Bayesian Image Analysis.

Classification

Classification is the process of predicting the class of given data points. Classes are sometimes called as targets/ labels or categories. Classification predictive modeling is the task of approximating a mapping function (f) from input variables (X) to discrete output variables (y).

For example, spam detection in email service providers can be identified as a classification problem. This is s binary classification since there are only 2 classes as spam and not spam. A classifier utilizes some training data to understand how given input variables relate to the class. In this case, known spam and non-spam emails have to be used as the training data. When the classifier is trained accurately, it can be used to detect an unknown email.

Classification belongs to the category of supervised learning where the targets also provided with the input data. There are many applications in classification in many domains such as in credit approval, medical diagnosis, target marketing etc.

There are two types of learners in classification as lazy learners and eager learners.

1. **Lazy learners**

Lazy learners simply store the training data and wait until a testing data appear. When it does, classification is conducted based on the most related data in the stored training data. Compared to eager learners, lazy learners have less training time but more time in predicting.

Ex. k-nearest neighbor, Case-based reasoning

2. Eager learners

Eager learners construct a classification model based on the given training data before receiving data for classification. It must be able to commit to a single hypothesis that covers the entire instance space. Due to the model construction, eager learners take a long time for train and less time to predict.

Ex. Decision Tree, Naive Bayes, Artificial Neural Networks

Classification algorithms

There is a lot of classification algorithms available now but it is not possible to conclude which one is superior to other. It depends on the application and nature of available data set. For example, if the classes are linearly separable, the linear classifiers like Logistic regression, Fisher's linear discriminant can outperform sophisticated models and vice versa.

Supervised vs Unsupervised learning

Two of the most commonly used strategies in machine learning include supervised learning and unsupervised learning.

supervised learning

Supervised learning is when you train a machine learning model using labelled data. It means that you have data that already have the right classification associated with them. One common use of supervised learning is to help you predict values for new data.

With supervised learning, you'll need to rebuild your models as you get new data to make sure that the predictions returned are still accurate.

unsupervised learning: Unsupervised learning is when you train a model with unlabeled data. This means that the model will have to find its own features and make predictions based on how it classifies the data.

An example of unsupervised learning would be giving your model pictures of multiple kinds of food with no labels.

algorithm

An algorithm is just a customizable math function. Most algorithms have things like cost functions, weight values, and parameter functions that you can interchange based on the data you're working with. At its core, machine learning is just a bunch of math equations that need to be solved really fast.

There are so many different algorithms to handle different kinds of data. One particular algorithm is the support vector machine (SVM) and that's what this article is going to cover in detail.

SVM

“Support Vector Machine” (SVM) is a supervised machine learning algorithm that can be used for both classification or regression challenges. ... The SVM classifier is a frontier that best segregates the two classes (hyper-plane/ line).

Support vector machines are a set of supervised learning methods used for classification, regression, and outliers detection. All of these are common tasks in machine learning.

There are specific types of SVMs you can use for particular machine learning problems, like support vector regression (SVR) which is an extension of support vector classification (SVC).

The main thing to keep in mind here is that these are just math equations tuned to give you the most accurate answer possible as quickly as possible.

SVMs are different from other classification algorithms because of the way they choose the decision boundary that maximizes the distance from the nearest data points of all the classes. The decision boundary created by SVMs is called the maximum margin classifier or the maximum margin hyper plane.

SVM working

A simple linear SVM classifier works by making a straight line between two classes. That means all of the data points on one side of the line will represent a category and the data points on the other side of the line will be put into a different category. This means there can be an infinite number of lines to choose from.

Advantage of linear SVM algorithm better than some of the other algorithms, like k-nearest neighbors, is that it chooses the best line to classify your data points. It chooses the line that separates the data and is the furthest away from the closet data points as possible.

A 2-D example helps to make sense of all the machine learning jargon. Basically you have some data points on a grid. Trying to separate these data points by the category they should fit in, but you don't want to have any data in the wrong category. That means you're trying to find the line between the two closest points that keeps the other data points separated.

So the two closest data points give you the support vectors you'll use to find that line. That line is called the decision boundary.

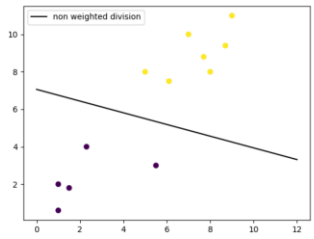


Fig 1.linear SVM

Types of SVMs

There are two different types of SVMs, each used for different things:

- Simple SVM: Typically used for linear regression and classification problems.
- Kernel SVM: Has more flexibility for non-linear data because you can add more features to fit a hyperplane instead of a two-dimensional space.

SVMs are used in applications like handwriting recognition, intrusion detection, face detection, email classification, gene classification, and in web pages. This is one of the reasons we use SVMs in machine learning. It can handle both classification and regression on linear and non-linear data.

Another reason we use SVMs is because they can find complex relationships between your data without you needing to do a lot of transformations on your own. It's a great option when you are working with smaller datasets that have tens to hundreds of thousands of features. They typically find more accurate results when compared to other algorithms because of their ability to handle small, complex datasets.

Here are some of the pros and cons for using SVMs.

Pros

- Effective on datasets with multiple features, like financial or medical data.

- Effective in cases where number of features is greater than the number of data points.
- Uses a subset of training points in the decision function called support vectors which makes it memory efficient.
- Different kernel functions can be specified for the decision function. You can use common kernels, but it's also possible to specify custom kernels.

Cons

- If the number of features is a lot bigger than the number of data points, avoiding over-fitting when choosing kernel functions and regularization term is crucial.
- SVMs don't directly provide probability estimates. Those are calculated using an expensive five-fold cross-validation.
- Works best on small sample sets because of its high training time.

Since SVMs can use any number of kernels, it's important that you know about a few of them.

Kernel functions

Linear

These are commonly recommended for text classification because most of these types of classification problems are linearly separable.

The linear kernel works really well when there are a lot of features, and text classification problems have a lot of features. Linear kernel functions are faster than most of the others and you have fewer parameters to optimize.

Here's the function that defines the linear kernel:

$$f(X) = w^T * X + b$$

In this equation, **w** is the weight vector that you want to minimize, **X** is the data that you're trying to classify, and **b** is the linear coefficient estimated from the training data. This equation defines the decision boundary that the SVM returns.

Polynomial

The polynomial kernel isn't used in practice very often because it isn't as computationally efficient as other kernels and its predictions aren't as accurate.

Here's the function for a polynomial kernel:

$$f(X1, X2) = (a + X1^T * X2) ^ b$$

This is one of the more simple polynomial kernel equations you can use. **f(X1, X2)** represents the polynomial decision boundary that will separate your data. **X1** and **X2** represent your data.

Gaussian Radial Basis Function (RBF)

One of the most powerful and commonly used kernels in SVMs. Usually the choice for non-linear data.

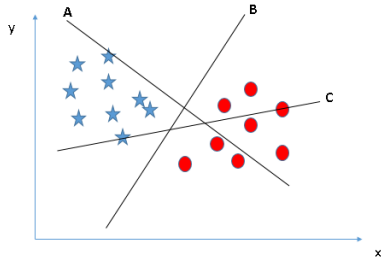
Here's the equation for an RBF kernel:
f(X1, X2) = exp(-gamma * ||X1 - X2||^2)
In this equation, **gamma** specifies how much a single training point has on the other data points around it. ||**X1 - X2**|| is the dot product between your features.

Sigmoid

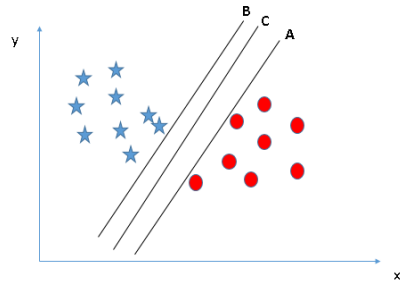
More useful in neural networks than in support vector machines, but there are occasional specific use cases.
Here's the function for a sigmoid kernel:

f(X, y) = tanh(alpha * X^T * y + C)
In this function, **alpha** is a weight vector and **C** is an offset value to account for some mis-classification of data that can happen.
There are plenty of other kernels that can used for project. This might be a decision to make to meet certain error constraints, To try and speed up the training time, or you want to super tune parameters.

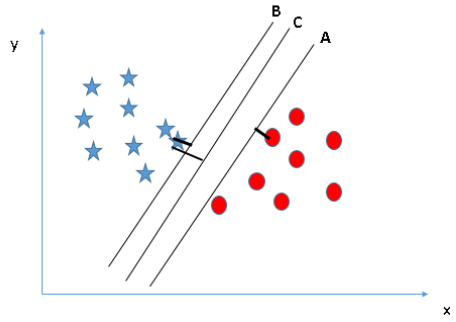
- **Identify the right hyper-plane (Scenario-1):** Here, we have three hyper-planes (A, B, and C). Now, identify the right hyper-plane to classify stars and circles.



- Remember a thumb rule to identify the right hyper-plane: "Select the hyper-plane which segregates the two classes better". In this scenario, hyper-plane "B" has excellently performed this job.
- **Identify the right hyper-plane (Scenario-2):** Here, we have three hyper-planes (A, B, and C) and all are segregating the classes well. Now, How can we identify the right hyper-plane?

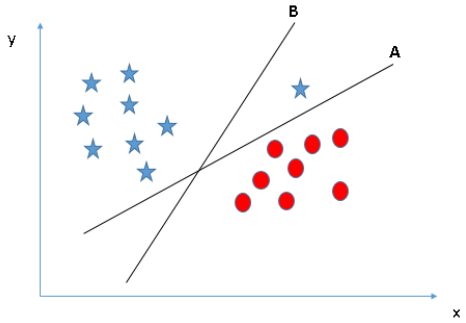


Here, maximizing the distances between nearest data point (either class) and hyper-plane will help us to decide the right hyper-plane. This distance is called as **Margin**. t:



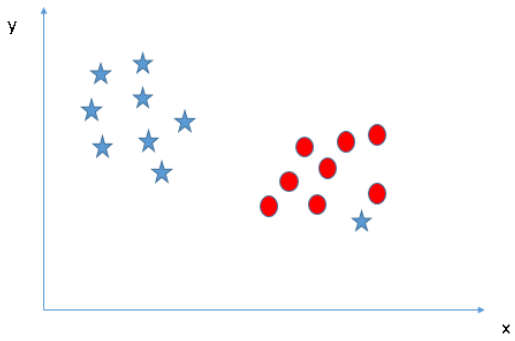
Above, shows that the margin for hyper-plane C is high as compared to both A and B. Hence, we name the right hyper-plane as C. Another lightning reason for selecting the hyper-plane with higher margin is robustness. If we select a hyper-plane having low margin then there is high chance of miss-classification.

Identify the right hyper-plane (Scenario-3): Hint: Use the rules as discussed in previous section to identify the right hyper-plane

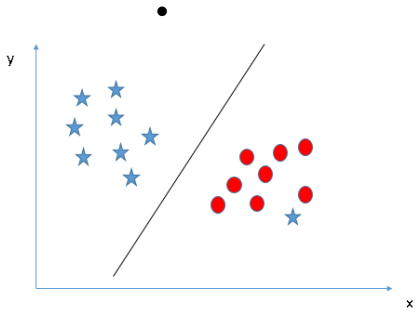


Some of you may have selected the hyper-plane **B** as it has higher margin compared to **A.** But, here is the catch, SVM selects the hyper-plane which classifies the classes accurately prior to maximizing margin. Here, hyper-plane B has a classification error and A has classified all correctly. Therefore, the right hyper-plane is **A.**

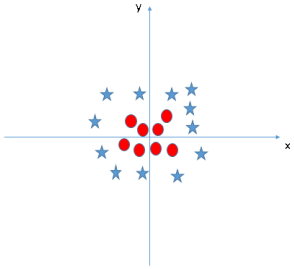
Can we classify two classes (Scenario-4) Below, I am unable to segregate the two classes using a straight line, as one of the stars lies in the territory of other(circle) class as an outlier.



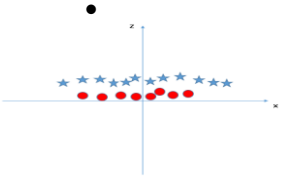
- One star at other end is like an outlier for star class. The SVM algorithm has a feature to ignore outliers and find the hyper-plane that has the maximum margin. Hence, we can say, SVM classification is robust to outliers.



- **Find the hyper-plane to segregate to classes (Scenario-5):** In the scenario below, we can't have linear hyper-plane between the two classes, so how does SVM classify these two classes? Till now, we have only looked at the linear hyper-plane.



- SVM can solve this problem. Easily! It solves this problem by introducing additional feature. Here, we will add a new feature z=x^2+y^2. Now, let's plot the data points on axis x and z:

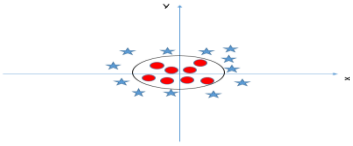


In above plot, points to consider are:

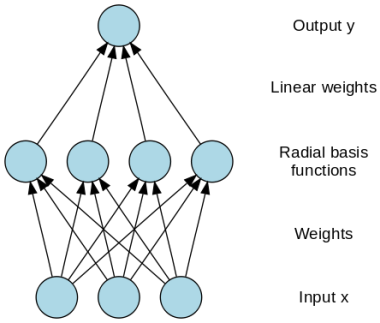
- All values for z would be positive always because z is the squared sum of both x and y
- In the original plot, red circles appear close to the origin of x and y axes, leading to lower value of z and star relatively away from the origin result to higher value of z.

In the SVM classifier, it is easy to have a linear hyper-plane between these two classes. But, another burning question which arises is, should we need to add this feature manually to have a hyper-plane. No, the SVM algorithm has a technique called the **kernel trick**. The SVM kernel is a function that takes low dimensional input space and transforms it to a higher dimensional space i.e. it converts not separable problem to separable problem. It is mostly useful in non-linear separation problem. Simply put, it does some extremely complex data transformations, then finds out the process to separate the data based on the labels or outputs you've defined.

When we look at the hyper-plane in original input space it looks like a circle:



Radial Basis Functions Neural Networks



- In Single Perceptron / Multi-layer Perceptron(MLP), we only have linear separability because they are composed of input and output layers(some hidden layers in MLP)

- For example, AND, OR functions are linearly-separable & XOR function is not linearly separable.

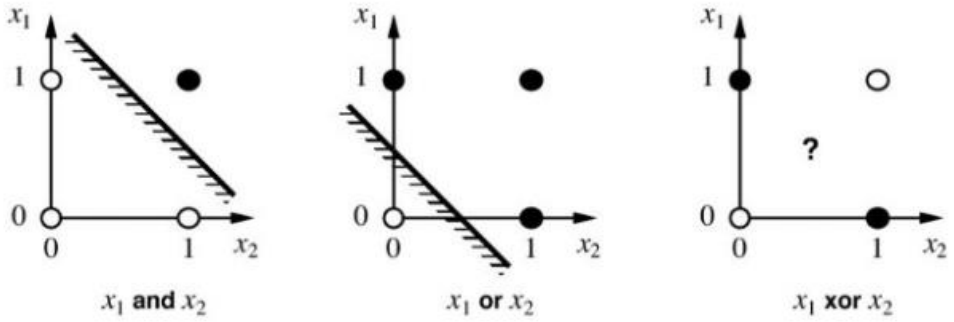


Fig 2.Linear-separability of AND, OR, XOR functions

- Atleast one hidden layer is needed to derive a non-linearity separation.
- RBNN transforms the input signal into another form, which can be then feed into the network to get linear separability.
- RBNN is structurally same as perceptron(MLP).

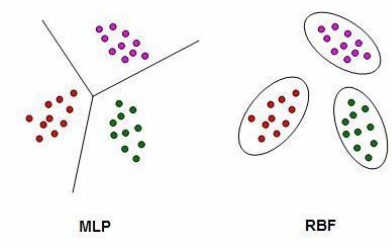


Fig. 3.Distinction between MLP and RBF

- RBNN is composed of input, hidden, and output layer. RBNN is strictly limited to have exactly one hidden layer. This hidden layer is called as feature vector.
- RBNN increases dimenion of feature vector.

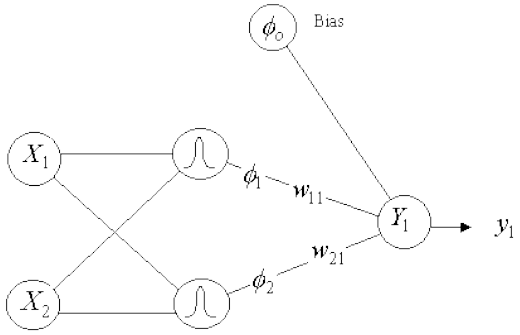


Fig.4.Simplest diagram shows the architecture of RBNN

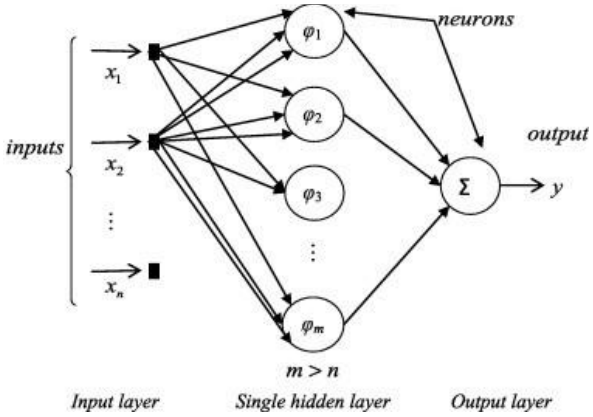


Fig.5.Extended diagram shows the architecture of RBNN with hidden functions.

- Apply non-linear transfer function to the feature vector before go for classification problem.
- When we increase the dimension of the feature vector, the linear separability of feature vector increases.

A non-linearity separable problem(pattern classification problem) is highly separable in high dimensional space than it is in low dimensional space.

- Gaussian Functions are generally used for Radian Basis Function(confrontal mapping). So we define the radial distance r = ||x- t||.

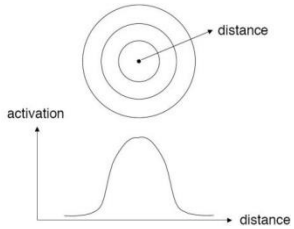


Fig. 6.Radial distance and Radial Basis function with confrontal map

Gaussian Radial Function : $\phi(r) = \exp(-r^2/2\sigma^2)$ where $\sigma > 0$

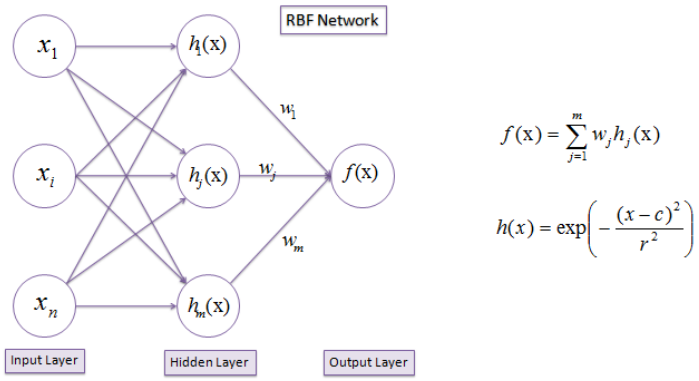


Fig. 7.RBF Network

Classification only happens on the second phase, where linear combination of hidden functions are driven to output layer.

- Example. XOR function :-

- let there is 4 inputs and there will not increase dimension at the feature vector here. So select 2 receptors here. For each transformation function $\phi(x)$, will have each receptors t.

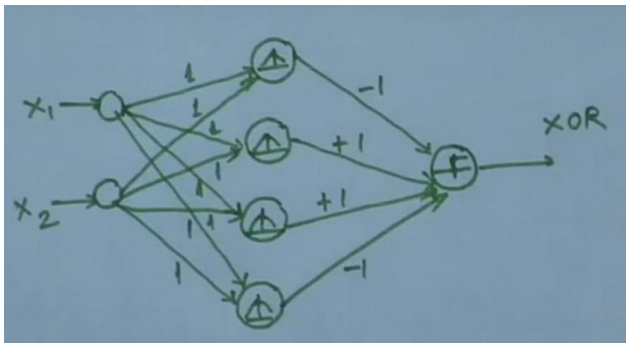
- Now consider the RBNN architecture,

- P := # of input features/ values.

- M = # of transformed vector dimensions (hidden layer width). So $M \geq P$ usually be.

- Each node in the hidden layer, performs a set of non-linear radian basis function.

- Output C will remains the same as for the classification problems(certain number of class labels as predefined).



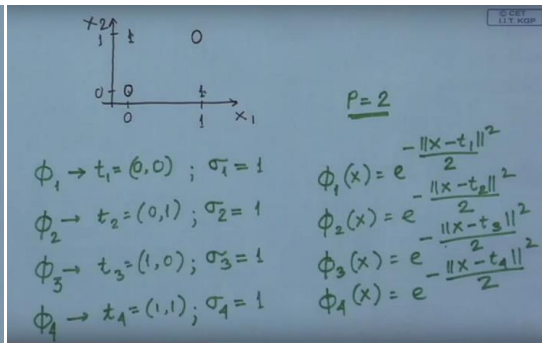
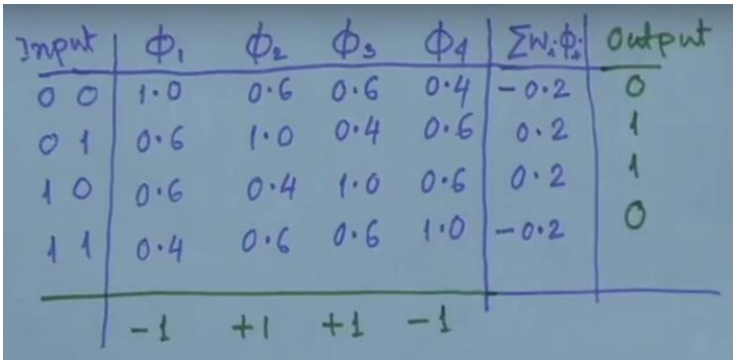


Fig.8.Transformation function with receptors and variances.



Input	ϕ_1	ϕ_2	ϕ_3	ϕ_4	$\sum w_i \phi_i$	output
0 0	1.0	0.6	0.6	0.4	−0.2	0
0 1	0.6	1.0	0.4	0.6	0.2	1
1 0	0.6	0.4	1.0	0.6	0.2	1
1 1	0.4	0.6	0.6	1.0	−0.2	0
	−1	+1	+1	−1		

Fig.9.Output → linear combination of transformation function is tabulated.

- Only Nodes in the hidden layer perform the radian basis transformation function.

- Output layer performs the linear combination of the outputs of the hidden layer to give a final probabilistic value at the output layer.

- So the classification is only done only @ (hidden layer → output layer)

Training the RBNN :-

- First, train the hidden layer using back propagation.

- Neural Network training(back propagation) is a curve fitting method. It fits a non-linear curve during the training phase. It runs through stochastic approximation, which we call the back propagation.

- For each of the node in the hidden layer, find t(receptors) & the variance (σ)[variance — the spread of the radial basis function]

- On the second training phase, update the weighting vectors between hidden layers & output layers.

- In hidden layers, each node represents each transformation basis function. Any of the function could satisfy the non-linear separability OR even combination of set of functions could satisfy the non-linear separability.

- So in our hidden layer transformation, all the non-linearity terms are included. Say like $X^2 + Y^2 + 5XY$; its all included in a hyper-surface equation(X & Y are inputs).

- Therefore, the first stage of training is done by clustering algorithm. Define the number of cluster centers needed. And by clustering algorithm, compute the cluster centers, which then is assigned as the receptors for each hidden neurons.

For cluster N samples or observations into M clusters (N > M).

- So the output "clusters" are the "receptors".

- for each receptors, find the variance as "the squared sum of the distances between the respective receptor & the each cluster nearest samples" := $1/N * ||X — t||^2$

- The interpretation of the first training phase is that the "feature vector is projected onto the transformed space".

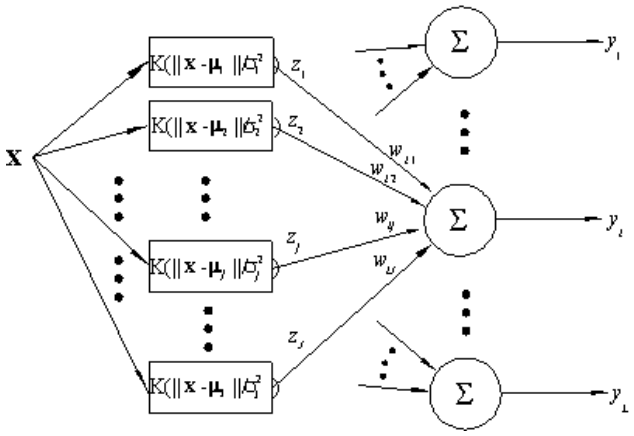


Fig.10. Complex diagram depicting the RBNN

Advantages of using RBNN than the MLP :-

1. Training in RBNN is faster than in Multi-layer Perceptron (MLP) → takes many interactions in MLP.

2. We can easily interpret what is the meaning / function of the each node in hidden layer of the RBNN. This is difficult in MLP.

3. (what should be the # of nodes in hidden layer & the # of hidden layers) this parameterization is difficult in MLP. But this is not found in RBNN.

4. Classification will take more time in RBNN than MLP.

Dimension Reduction:

Dimensionality reduction refers to techniques for reducing the number of input variables in training data. Fewer input dimensions often mean correspondingly fewer parameters or a simpler structure in the machine learning model, referred to as degrees of freedom.

In machine learning classification problems, there are often too many factors on the basis of which the final classification is done. These factors are basically variables called features. The higher the number of features, the harder it gets to visualize the training set and then work on it. Sometimes, most of these features are correlated, and hence redundant. This is where dimensionality reduction algorithms come into play. Dimensionality reduction is the process of reducing the number of random variables under consideration, by obtaining a set of principal variables. It can be divided into feature selection and feature extraction.

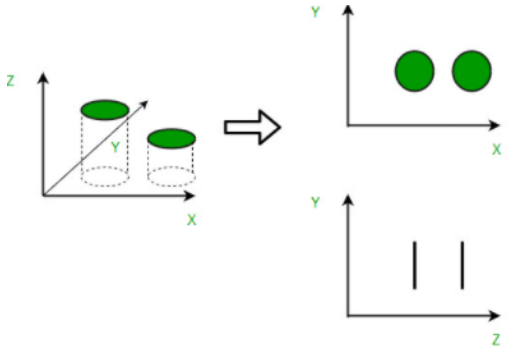


Fig. 11.Components of Dimensionality Reduction

There are two components of dimensionality reduction:

- **Feature selection:** In this, we try to find a subset of the original set of variables, or features, to get a smaller subset which can be used to model the problem. It usually involves three ways:
  1. Filter
  2. Wrapper
  3. Embedded
- **Feature extraction:** This reduces the data in a high dimensional space to a lower dimension space, i.e. a space with lesser no. of dimensions.

Methods of Dimensionality Reduction

The various methods used for dimensionality reduction include:

- Principal Component Analysis (PCA)
- Linear Discriminant Analysis (LDA)
- Generalized Discriminant Analysis (GDA)

Dimensionality reduction may be both linear or non-linear, depending upon the method used. The prime linear method, called Principal Component Analysis, or PCA, is discussed below.

Principal Component Analysis

This method was introduced by Karl Pearson. It works on a condition that while the data in a higher dimensional space is mapped to data in a lower dimension space, the variance of the data in the lower dimensional space should be maximum.

Principal Component Analysis, or PCA, is a dimensionality-reduction method that is often used to reduce the dimensionality of large data sets, by transforming a large set of variables into a smaller one that still contains most of the information in the large set.

Reducing the number of variables of a data set naturally comes at the expense of accuracy, but the trick in dimensionality reduction is to trade a little accuracy for simplicity. Because smaller data sets are easier to explore and visualize and make analyzing data much easier and faster for machine learning algorithms without extraneous variables to process.

So to sum up, the idea of PCA is simple — reduce the number of variables of a data set, while preserving as much information as possible.

It involves the following steps:
Construct the covariance matrix of the data.

- Compute the eigenvectors of this matrix.
- Eigenvectors corresponding to the largest eigenvalues are used to reconstruct a large fraction of variance of the original data.

Hence, we are left with a lesser number of eigenvectors, and there might have been some data loss in the process. But, the most important variances should be retained by the remaining eigenvectors.

STEP 1: STANDARDIZATION

The aim of this step is to standardize the range of the continuous initial variables so that each one of them contributes equally to the analysis.

More specifically, the reason it is critical to perform standardization prior to PCA, is that the latter is quite sensitive regarding the variances of the initial variables. That is, if there are large differences between the ranges of initial variables, those variables with larger ranges will dominate over those with small ranges (For example, a variable that ranges between 0 and 100 will dominate over a variable that ranges between 0 and 1), which will lead to biased results. So, transforming the data to comparable scales can prevent this problem.

Mathematically, this can be done by subtracting the mean and dividing by the standard deviation for each value of each variable.

$$z = \frac{value - mean}{standard\ deviation}$$

Once the standardization is done, all the variables will be transformed to the same scale.

STEP 2: COVARIANCE MATRIX COMPUTATION

The aim of this step is to understand how the variables of the input data set are varying from the mean with respect to each other, or in other words, to see if there is any relationship between them. Because sometimes, variables are highly correlated in such a way that they contain redundant information. So, in order to identify these correlations, we compute the covariance matrix.

The covariance matrix is a $p \times p$ symmetric matrix (where p is the number of dimensions) that has as entries the covariances associated with all possible pairs of the initial variables. For example, for a 3-dimensional data set with 3 variables *x*, *y*, and *z*, the covariance matrix is a 3×3 matrix of this from:

$$\begin{bmatrix} Cov(x,x) & Cov(x,y) & Cov(x,z) \\ Cov(y,x) & Cov(y,y) & Cov(y,z) \\ Cov(z,x) & Cov(z,y) & Cov(z,z) \end{bmatrix}$$

Fig .13.Covariance Matrix for 3-Dimensional Data

Since the covariance of a variable with itself is its variance (Cov(a,a)=Var(a)), in the main diagonal (Top left to bottom right) actually have the variances of each initial variable. And since the covariance is commutative (Cov(a,b)=Cov(b,a)), the entries of the covariance matrix are symmetric with respect to the main diagonal, which means that the upper and the lower triangular portions are equal. The covariance matrix is not more than a table that summaries the correlations between all the possible pairs of variables.

STEP 3: COMPUTE THE EIGENVECTORS AND EIGENVALUES OF THE COVARIANCE MATRIX TO IDENTIFY THE PRINCIPAL COMPONENTS

Eigenvectors and eigenvalues are the linear algebra concepts that we need to compute from the covariance matrix in order to determine the ***principal components*** of the data.
Principal components are new variables that are constructed as linear combinations or mixtures of the initial variables. These combinations are done in such a way that the new variables (i.e., principal components) are uncorrelated and most of the information within the initial variables is squeezed or compressed into the first components. So, the idea is 10-dimensional data gives 10 principal components, but PCA tries to put maximum possible information in the first component, then maximum remaining information in the second and so on, until having something like shown in the screen plot below.

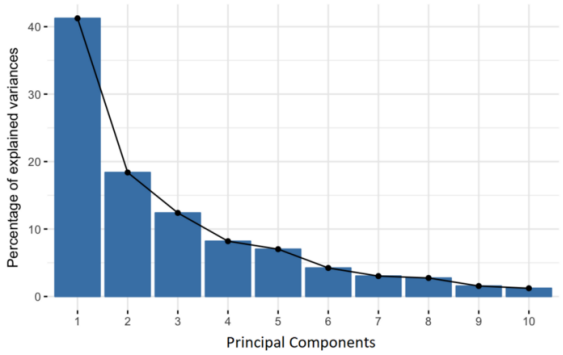


Fig 14.Percentage of Variance (Information) for each by PC

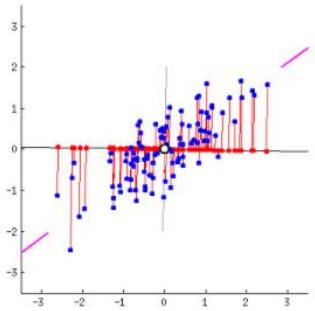
Organizing information in principal components this way, will allow to reduce dimensionality without losing much information, and this by discarding the components with low information and considering the remaining components as your new variables.

An important thing to realize here is that, the principal components are less interpretable and don't have any real meaning since they are constructed as linear combinations of the initial variables.

Geometrically speaking, principal components represent the directions of the data that explain a **maximal amount of variance**, that is to say, the lines that capture most information of the data. The relationship between variance and information here, is that, the larger the variance carried by a line, the larger the dispersion of the data points along it, and the larger the dispersion along a line, the more the information it has. To put all this simply, just think of principal components as new axes that provide the best angle to see and evaluate the data, so that the differences between the observations are better visible.

Constructs the Principal Components

As there are as many principal components as there are variables in the data, principal components are constructed in such a manner that the first principal component accounts for the **largest possible variance** in the data set. For example, let's assume that the scatter plot of our data set is as shown below, can we guess the first principal component ? Yes, it's approximately the line that matches the purple marks because it goes through the origin and it's the line in which the projection of the points (red dots) is the most spread out. Or mathematically speaking, it's the line that maximizes the variance (the average of the squared distances from the projected points (red dots) to the origin).



The second principal component is calculated in the same way, with the condition that it is uncorrelated with (i.e., perpendicular to) the first principal component and that it accounts for the next highest variance.This continues until a total of p principal components have been calculated, equal to the original number of variables. Eigen values are simply the coefficients attached to eigenvectors, which give the *amount of variance carried in each Principal Component*. By ranking your eigenvectors in order of their eigen values, highest to lowest, you get the principal components in order of significance.

STEP 4: FEATURE VECTOR

In the previous step, computing the eigenvectors and ordering them by their eigenvalues in descending order, allow us to find the principal components in order of significance. In this step is, to choose whether to keep all these components or discard those of lesser significance (of low eigenvalues), and form with the remaining ones a matrix of vectors that we call *Feature vector*. So, the feature vector is simply a matrix that has as columns the eigenvectors of the components that we decide to keep. This makes it the first step towards dimensionality reduction, because if we choose to keep only ***p*** eigenvectors (components) out of ***n***, the final data set will have only ***p*** dimensions.

LAST STEP: RECAST THE DATA ALONG THE PRINCIPAL COMPONENTS AXES

In this step, which is the last one, the aim is to use the feature vector formed using the eigenvectors of the covariance matrix, to reorient the data from the original axes to the ones represented by the principal components (hence the name Principal Components Analysis). This can be done by multiplying the transpose of the original data set by the transpose of the feature vector.

$$FinalDataSet = FeatureVector^T * StandardizedOriginalDataSet^T$$

Advantages of Dimensionality Reduction

- It helps in data compression, and hence reduced storage space.
- It reduces computation time.
- It also helps remove redundant features, if any.

Disadvantages of Dimensionality Reduction

- It may lead to some amount of data loss.
- PCA tends to find linear correlations between variables, which is sometimes undesirable.
- PCA fails in cases where mean and covariance are not enough to define datasets.
- We may not know how many principal components to keep- in practice, some thumb rules are applied.

INDEPENDENT COMPONENT ANALYSIS

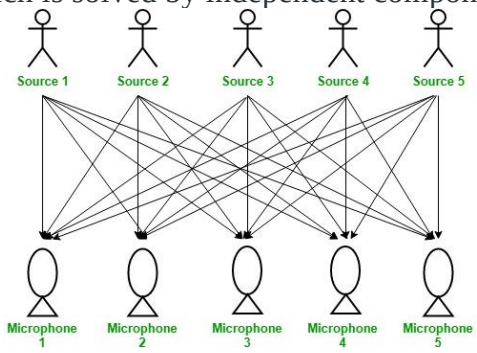
Independent Component Analysis (ICA) is a machine learning technique to separate independent sources from a mixed signal. Unlike principal component analysis which focuses on maximizing the variance of the data points, the independent component analysis focuses on independence, i.e. independent components.

Problem: To extract independent sources' signals from a mixed signal composed of the signals from those sources.
Given: Mixed signal from five different independent sources.
Aim: To decompose the mixed signal into independent sources:

- Source 1
- Source 2
- Source 3
- Source 4
- Source 5

Solution: Independent Component Analysis (ICA).

Consider *Cocktail Party Problem* or *Blind Source Separation* problem to understand the problem which is solved by independent component analysis.



Here, There is a party going into a room full of people. There is 'n' number of speakers in that room and they are speaking simultaneously at the party. In the same room, there are also 'n' number of microphones placed at different distances from the speakers which are recording 'n' speakers' voice signals. Hence, the number of speakers is equal to the number must of microphones in the room.

Now, using these microphones' recordings, we want to separate all the 'n' speakers' voice signals in the room given each microphone recorded the voice signals coming from each speaker of different intensity due to the difference in distances between them. Decomposing the mixed signal of each microphone's recording into independent source's speech signal can be done by using the machine learning technique, independent component analysis.
[X1, X2, ..., Xn] => [Y1, Y2, ..., Yn]
where, X1, X2, …, Xn are the original signals present in the mixed signal and Y1, Y2, …, Yn are the new features and are independent components which are independent of each other.

Restrictions on ICA –

1. The independent components generated by the ICA are assumed to be statistically independent of each other.
2. The independent components generated by the ICA must have non-gaussian distribution.
3. The number of independent components generated by the ICA is equal to the number of observed mixtures.

Difference between PCA and ICA –

Principal Component Analysis	Independent Component Analysis
It reduces the dimensions to avoid the problem of overfitting.	It decomposes the mixed signal into its independent sources' signals.
It deals with the Principal Components.	It deals with the Independent Components.
It focuses on maximizing the variance.	It doesn't focus on the issue of variance among the data points.
It focuses on the mutual orthogonality property of the principal components.	It doesn't focus on the mutual orthogonality of the components.
It doesn't focus on the mutual independence of the components.	It focuses on the mutual independence of the components.

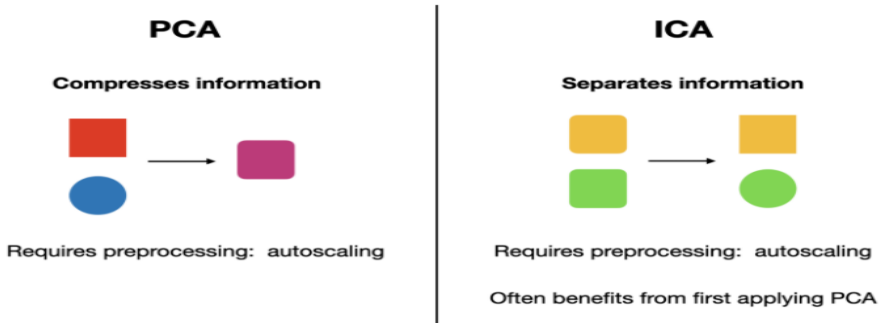
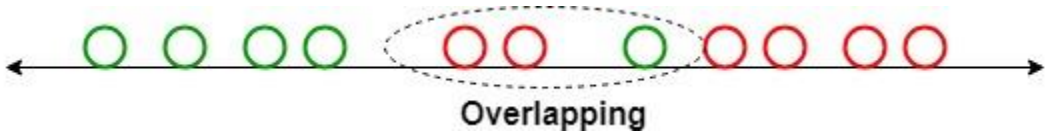


Fig. 14.Comparison of PCA and ICA

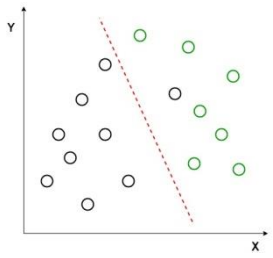
LINEAR DISCRIMINANT ANALYSIS

Linear Discriminant Analysis or **Normal Discriminant Analysis** or **Discriminant Function Analysis** is a dimensionality reduction technique which is commonly used for the supervised classification problems. It is used for modeling differences in groups i.e. separating two or more classes. It is used to project the features in higher dimension space into a lower dimension space.

For example, two classes and need to separate them efficiently. Classes can have multiple features. Using only a single feature to classify them may result in some overlapping as shown in the below figure. So, keep on increasing the number of features for proper classification.

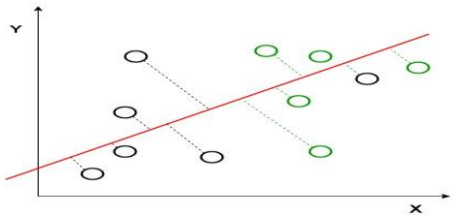


Example:
Suppose there are two sets of data points belonging to two different classes that want to be classify. As shown in the given 2D graph, when the data points are plotted on the 2D plane, there's no straight line that can separate the two classes of the data points completely. Hence, in this case, LDA (Linear Discriminant Analysis) is used which reduces the 2D graph into a 1D graph in order to maximize the separability between the two classes.



Here, Linear Discriminant Analysis uses both the axes (X and Y) to create a new axis and projects data onto a new axis in a way to maximize the separation of the two categories and hence, reducing the 2D graph into a 1D graph.
Two criteria are used by LDA to create a new axis:

1. Maximize the distance between means of the two classes.
2. Minimize the variation within each class.



In the above graph, it can be seen that a new axis (in red) is generated and plotted in the 2D graph such that it maximizes the distance between the means of the two classes and minimizes the variation within each class. In simple terms, this newly generated axis increases the separation between the data points of the two classes. After generating this new axis using the above-mentioned criteria, all the data points of the classes are plotted on this new axis and are shown in the figure given below.

But Linear Discriminant Analysis fails when the mean of the distributions are shared, as it becomes impossible for LDA to find a new axis that makes both the classes linearly separable. In such cases, we use non-linear discriminant analysis.

Mathematics
Let's suppose we have two classes and a d- dimensional samples such as x1, x2 … xn, where:

- n1 samples coming from the class (c1) and n2 coming from the class (c2).

If xi is the data point, then its projection on the line represented by unit vector v can be written as vTxi

Let’s consider u1 and u2 be the means of samples class c1 and c2 respectively before projection and u1hat denotes the mean of the samples of class after projection and it can be calculated by:

$$\widetilde{\mu_1} = \frac{1}{n_1} \sum_{x_i \in c_1}^{n_1} v^T x_i = v^T \mu_1$$

Similarly,

$$\widetilde{\mu_2} = v^T \mu_2$$

Now, In LDA we need to normalize |\widetilde{\mu_1} -\widetilde{\mu_2} |. Let y_i = v^{T}x_i be the projected samples, then scatter for the samples of c1 is:

$$\widetilde{s_1^2} = \sum_{y_i \in c_1} (y_i - \mu_1)^2$$

Similarly:

$$\widetilde{s_2^2} = \sum_{y_i \in c_1} (y_i - \mu_2)^2$$

Now, we need to project our data on the line having direction v which maximizes

$$J(v) = \frac{\widetilde{\mu_1} - \widetilde{\mu_2}}{\widetilde{s_1^2} + \widetilde{s_2^2}}$$

For maximizing the above equation we need to find a projection vector that maximizes the difference of means of reduces the scatters of both classes. Now, scatter matrix of s1 and s2 of classes c1 and c2 are:

$$s_1 = \sum_{x_i \in c_1} (x_i - \mu_1)(x_i - \mu_1)^T$$

and s2

$$s_2 = \sum_{x_i \in c_2} (x_i - \mu_2)(x_i - \mu_2)^T$$

After simplifying the above equation, we get:

Now, we define, scatter within the classes(s_w) and scatter b/w the classes(s_b):

$$s_w = s_1 + s_2$$

$$s_b = (\mu_1 - \mu_2)(\mu_1 - \mu_2)^T$$

Now, we try to simplify the numerator part of J(v)

$$J(v) = \frac{|\widetilde{\mu_1} - \widetilde{\mu_2}|}{\widetilde{s_1^2} + \widetilde{s_2^2}} = \frac{v^T s_b v}{v^T s_w v}$$

Now, To maximize the above equation we need to calculate differentiation with respect to v

$$\frac{dJ(v)}{dv} = s_b v - \frac{v^t s_b v (s_w v)}{v^T s_w v}$$

$$= s_b v - \lambda s_w v = 0$$

$$s_b v = \lambda s_w v$$

$$s_w^{-1} s_b v = \lambda v$$

$$Mv = \lambda v$$

$$where,$$

$$\lambda = \frac{v^T s_b v}{v^T s_w v} and$$

$$M = s_w^{-1} s_b$$

Here, for the maximum value of J(v) we will use the value corresponding to the highest eigenvalue. This will provide us the best solution for LDA.

Extensions to LDA:

1. **Quadratic Discriminant Analysis (QDA):** Each class uses its own estimate of variance (or covariance when there are multiple input variables).
2. **Flexible Discriminant Analysis (FDA):** Where non-linear combinations of inputs is used such as splines.

3. **Regularized Discriminant Analysis (RDA):** Introduces regularization into the estimate of the variance (actually covariance), moderating the influence of different variables on LDA.

Naive Bayes Classifiers

Naive Bayes classifiers are a collection of classification algorithms based on **Bayes' Theorem**. It is not a single algorithm but a family of algorithms where all of them share a common principle, i.e. every pair of features being classified is independent of each other.

To start with, let us consider a dataset.

Consider a fictional dataset that describes the weather conditions for playing a game of golf. Given the weather conditions, each tuple classifies the conditions as fit("Yes") or unfit("No") for plaing golf.

Here is a tabular representation of our dataset.

Outlook	Temperature	Humidity	Windy	Play Golf
Rainy	Hot	High	False	No
Rainy	Hot	High	True	No
Overcast	Hot	High	False	Yes
Sunny	Mild	High	False	Yes
Sunny	Cool	Normal	False	Yes
Sunny	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Rainy	Mild	High	False	No

Outlook	Temperature	Humidity	Windy	Play Golf
Rainy	Cool	Normal	False	Yes
Sunny	Mild	Normal	False	Yes
Rainy	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Sunny	Mild	High	True	No

The dataset is divided into two parts, namely, **feature matrix** and the **response vector**.

- Feature matrix contains all the vectors(rows) of dataset in which each vector consists of the value of **dependent features**. In above dataset, features are 'Outlook', 'Temperature', 'Humidity' and 'Windy'.
- Response vector contains the value of **class variable**(prediction or output) for each row of feature matrix. In above dataset, the class variable name is 'Play golf'.

Assumption:

The fundamental Naive Bayes assumption is that each feature makes an:

- independent
- equal

contribution to the outcome.

With relation to our dataset, this concept can be understood as:

- We assume that no pair of features are dependent. For example, the temperature being 'Hot' has nothing to do with the humidity or the outlook being 'Rainy' has no effect on the winds. Hence, the features are assumed to be **independent**.
- Secondly, each feature is given the same weight(or importance). For example, knowing only temperature and humidity alone can't predict the outcome accuratey. None of the attributes is irrelevant and assumed to be contributing **equally** to the outcome.

Note: The assumptions made by Naive Bayes are not generally correct in real-world situations. In-fact, the independence assumption is never correct but often works well in practice.

Now, before moving to the formula for Naive Bayes, it is important to know about Bayes' theorem.

Bayes' Theorem finds the probability of an event occurring given the probability of another event that has already occurred. Bayes' theorem is stated mathematically as the following equation:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

where A and B are events and P(B) ? 0.

- Basically, we are trying to find probability of event A, given the event B is true. Event B is also termed as **evidence**.
- P(A) is the **priori** of A (the prior probability, i.e. Probability of event before evidence is seen). The evidence is an attribute value of an unknown instance(here, it is event B).
- P(A|B) is a posteriori probability of B, i.e. probability of event after evidence is seen.

Now, with regards to our dataset, apply Bayes' theorem in following way:

$$P(y|X) = \frac{P(X|y)P(y)}{P(X)}$$

where, y is class variable and X is a dependent feature vector (of size *n*) where:

$$X = (x_1, x_2, x_3, ..., x_n)$$

Just to clear, an example of a feature vector and corresponding class variable can be:
(refer 1st row of dataset)

```
X = (Rainy, Hot, High, False)
y = No
```

So basically, P(y|X) here means, the probability of "Not playing golf" given that the weather conditions are "Rainy outlook", "Temperature is hot", "high humidity" and "no wind".

Naive assumption

Now, its time to put a naive assumption to the Bayes' theorem, which is, **independence** among the features. So now, split **evidence** into the independent parts.
Now, if any two events A and B are independent, then,

P(A,B) = P(A)P(B)
Hence, we reach to the result:

$$P(y|x_1,...,x_n) = \frac{P(x_1|y)P(x_2|y)...P(x_n|y)P(y)}{P(x_1)P(x_2)...P(x_n)}$$

which can be expressed as:

$$P(y|x_1,...,x_n) = \frac{P(y)\prod_{i=1}^{n} P(x_i|y)}{P(x_1)P(x_2)...P(x_n)}$$

Now, as the denominator remains constant for a given input, we can remove that term:

$$P(y|x_1,...,x_n) \propto P(y)\prod_{i=1}^{n} P(x_i|y)$$

Now, need to create a classifier model. For this, find the probability of given set of inputs for all possible values of the class variable *y* and pick up the output with maximum probability. This can be expressed mathematically as:

$$y = argmax_y P(y)\prod_{i=1}^{n} P(x_i|y)$$

calculate P(y) and P(x_i | y).
P(y) is also called **class probability** and P(x_i | y) is called **conditional probability**.
The different naive Bayes classifiers differ mainly by the assumptions they make regarding the distribution of P(x_i | y).

We need to find P(x_i | y_j) for each x_i in X and y_j in y. All these calculations have been demonstrated in the tables below:

Outlook

	Yes	No	P(yes)	P(no)
Sunny	2	3	2/9	3/5
Overcast	4	0	4/9	0/5
Rainy	3	2	3/9	2/5
Total	9	5	100%	100%

Temperature

	Yes	No	P(yes)	P(no)
Hot	2	2	2/9	2/5
Mild	4	2	4/9	2/5
Cool	3	1	3/9	1/5
Total	9	5	100%	100%

Humidity

	Yes	No	P(yes)	P(no)
High	3	4	3/9	4/5
Normal	6	1	6/9	1/5
Total	9	5	100%	100%

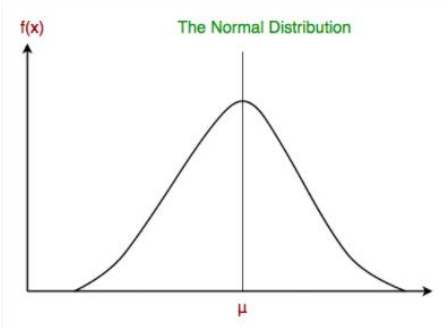
Wind

	Yes	No	P(yes)	P(no)
False	6	2	6/9	2/5
True	3	3	3/9	3/5
Total	9	5	100%	100%

Play		P(Yes)/P(No)
Yes	9	9/14
No	5	5/14
Total	14	100%

Gaussian Naive Bayes classifier

In Gaussian Naive Bayes, continuous values associated with each feature are assumed to be distributed according to a **Gaussian distribution**. A Gaussian distribution is also called Normal distribution. When plotted, it gives a bell shaped curve which is symmetric about the mean of the feature values as shown below:



The likelihood of the features is assumed to be Gaussian, hence, conditional probability is given by:

$$P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} exp\left(-\frac{(x_i-\mu_y)^2}{2\sigma_y^2}\right)$$

Other popular Naive Bayes classifiers are:

- **Multinomial Naive Bayes**: Feature vectors represent the frequencies with which certain events have been generated by a **multinomial distribution**. This is the event model typically used for document classification.
- **Bernoulli Naive Bayes**: In the multivariate Bernoulli event model, features are independent booleans (binary variables) describing inputs. Like the multinomial model, this model is popular for document classification tasks, where binary term occurrence(i.e. a word occurs in a document or not) features are used rather than term frequencies(i.e. frequency of a word in the document).

MARKOV RANDOM FIELD
Algorithm

A **Markov random field** is an undirected graph where each node captures the (discrete or Gaussian) probability distribution of a variable and the edges represent dependencies between those variables and are weighted to represent the relative strengths of the dependencies. Because of its undirected nature, a Markov random field can contain subgraphs of three or more interconnected nodes that interact in terms of the joint probabilities they represent in a way reminiscent of the Hopfield network. Furthermore, Markov random fields can also be configured to take into account the probabilities of individual variables having certain values (**unary factors**): for example, the impact of a very improbable value for a given variable on the rest of the network can be reduced on the basis that it is more likely to be a sampling error than a genuine reading.

Markov random fields represent a very general model that specifies little about how they might be built, trained or used. Clearly, when all the variables in a large Markov random field except one are known, it will be easy to calculate the single missing probability. In fact, it can be shown that it is possible for any group of node values within a Markov random field to be calculated based on the remaining node values, but this is an NP-hard problem that, in practice, has to be solved using approximation techniques.

The defining difference between a Bayesian network and a Markov random field is that a Bayesian network is directed while a Markov random field is undirected. There are some graphs that can be converted from one type to the other without losing information (apart from the directedness), but each type of model is also able to capture information that the other type cannot. Contrary to what the name might suggest, a hidden Markov model is directed and thus a sub-type of Bayesian network rather than a sub-type of Markov random field.

A **conditional random field** is a Markov random field that has specifically been set up to allow the values of a specific variable or variables to be predicted based on the values of the other variables. Conditional random fields are closely related to other algorithms:

- Conditional random fields with discrete probability distributions are essentially isomorphic with (simple or multinominal) logistic regression in that the aim is to calculate the probabilities of variables having specific values based on known values of other variables. The main difference is that conditional random fields, like hidden Markov models, are normally used to model sequential time series: the network structure of a conditional random field is more suited to modelling such data than a simple logistic regression equation is.

- In as much as the relationships within a conditional random field are expressed as weights that are trained using backpropagation, the conditional random field also has a lot in common with neural network architectures like the perceptron . Differences are:

    - the standard type of conditional random field can only solve linear problems (although the same is true of the original form of the one-row perceptron; and there are also multilayer conditional random fields that can be used for more complex problem spaces);

    - a conditional random field is normally parametric, i.e. the data scientist specifies the precise nature of the relationships being modelled, while a perceptron is normally non-parametric, i.e. the data scientist supplies the raw data and expects the perceptron to learn the nature of the relationships as well as their relative weights.

Maximum Entropy Inference

Maximum entropy is a powerful concept that entails a sharp separation between relevant and irrelevant variables. It is typically invoked in inference, once an assumption is made on what the relevant variables are, in order to estimate a model from data, that affords predictions on all other (dependent) variables. Conversely, maximum entropy can be invoked to retrieve the relevant variables (sufficient statistics) directly from the data, once a model is identified by Bayesian model selection. We explore this approach in the case of spin models with interactions of arbitrary order, and we discuss how relevant interactions can be inferred. In this perspective, the dimensionality of the inference problem is not set by the number of parameters in the model, but by the frequency distribution of the data.

Statistical mechanics stems from classical (or quantum) mechanics. The latter prescribes which are the relevant quantities (i.e., the conserved ones). The former brings this further, and it predicts that the probability to observe a system in a microscopic state s, in thermal equilibrium, is given by:

$$P(\boldsymbol{s}) = \frac{1}{Z} e^{-\beta H[\boldsymbol{s}]}$$

where H[s] is the energy of configuration s. The inverse temperature β is the only relevant parameter that needs to be adjusted, so that the ensemble average [H] matches the observed

energy U. It has been argued that the recipe that leads from H[s] to the distribution P(s) is maximum entropy: among all distributions that satisfy [H] = U, the one maximizing the entropy S = − ∑s P(s)log P(s) should be chosen. Information theory clarifies that the distribution Equation (1) is the one that assumes nothing else but hHi = U, or equivalently, that all other observables can be predicted from the knowledge of H[s]. This idea carries through more generally to inference problems: given a dataset of N observations sˆ = {s (1) , . . . , s (N)} of a system, one may invoke maximum entropy to infer the underlying distribution P(s) that reproduces the empirical averages of a set M of observables φ μ (s) (μ ∈ M). This leads to Equation (1) with:

$$-\beta H[\boldsymbol{s}] = \sum_{\mu \in \mathcal{M}} g^{\mu} \phi^{\mu}(\boldsymbol{s})$$

where the parameters g^{μ} should be fixed by solving the convex optimization problems:

$$\langle \phi^{\mu} \rangle = \overline{\phi^{\mu}} \equiv \frac{1}{N} \sum_{i=1}^{N} \phi^{\mu} \left(\boldsymbol{s}^{(i)} \right)$$

that result from entropy maximization and are also known to coincide with maximum likelihood estimation (see [2–4]).

For example, in the case of spin variables $\boldsymbol{s} \in \{\pm 1\}^n$, the distribution that reproduces empirical averages $\overline{s_i}$ and correlations $\overline{s_i s_j}$ is the pairwise model:

$$H[\boldsymbol{s}] = -\sum_i h_i s_i - \sum_{i<j} J_{ij} s_i s_j \,, \tag{4}$$

which in the case Jij ≡ J ∀i, j and hi ≡ h ∀i is the celebrated Ising model. The literature on inference of Ising models, stemming from the original paper on Boltzmann learning [5] to early applications to neural data [6] has grown considerably (see [7] for a recent review), to the point that some suggested [8] that a purely data-based statistical mechanics is possible.

We contrast a view of inference as parameter estimation of a preassigned (pairwise) model, where maximum entropy serves merely an ancillary purpose, with the one where the ultimate goal of statistical inference is precisely to identify the minimal set M of sufficient statistics that, for a given dataset sˆ, accurately reproduces all empirical averages. In this latter perspective, maximum entropy plays a key role in that it affords a sharp distinction between relevant variables (φ μ (s), μ ∈ M) (which are the sufficient statistics) and irrelevant ones, i.e., all other operators that are not a linear combination of the relevant ones, but whose values can be predicted through theirs. To some extent, understanding amounts precisely to distinguishing the relevant variables from the “dependent” ones: those whose values can be predicted.

Bayesian model selection provides a general recipe for identifying the best model M; the procedure is computationally unfeasible for spin models with interactions of arbitrary order, even for moderate dimensions (n = 5). Our strategy will then be to perform model selection within the

class of mixture models, where it is straightforward , and then, to project the result on spin models. The most likely models in this setting are those that enforce a symmetry among configurations that occur with the same frequency in the dataset. These symmetries entail a decomposition of the log-likelihood with a flavor that is similar to principal component analysis. This directly predicts the sufficient statistics ψλ(s) that need to be considered in maximum entropy inference. Interestingly, we find that the number of sufficient statistics depends on the frequency distribution of observations in the data. This implies that the dimensionality of the inference problem is not determined by the number of parameters in the model, but rather by the richness of the data.

BAYESIAN IMAGE ANALYSIS

Statistical approaches to image analysis using the Bayesian paradigm have proved to be very successful. Initially, the methodology was primarily developed for low-level image analysis but is increasingly used for high-level tasks. Using the Bayesian paradigm one requires a prior model which represents our initial knowledge about the objects in a particular scene and a likelihood model which is the joint probability distribution of the image, dependent on the objects in the scene. By us in^ Bayes' theorem, we obtain the posterior distribution of the objects in the scene, which can be used for inference, e.g. segmentation and object recognition. In low-level image analysis, the prior could be saying an Ising model, specifying that nearby pixels will tend to have similar grey levels, i.e. the scene is composed mainly of large homogeneous objects.

Bayesian analysis revolves around the posterior probability, which summarizes the degree of one's certainty concerning a given situation. Bayes's law states that the posterior probability is proportional to the product of the likelihood and the prior probability. The likelihood encompasses the information contained in the new data. The prior expresses the degree of certainty concerning the situation before the data are taken. Although the posterior probability completely describes the state of certainty about any possible image, it is often necessary to select a single image as the 'result' or reconstruction. A typical choice is that image that maximizes the posterior probability, which is called the MAP estimate. Other choices for the estimator maybe more desirable, for example, the mean of the posterior density function. In situations where only very limited data are available, the data alone may not be sufficient to specify a unique solution to the problem. The prior introduced with the Bayesian method can help guide the result toward a preferred solution. As the MAP solution differs from the maximum likelihood (ML) solution solely because of the prior, choosing the prior is one of the most critical aspects of Bayesian analysis.

By Bayes' theorem, the posterior density n(S, GJF) of the deformed template S and generated image G, given the observed image F is proportional to

$$L(F|G)\pi(G|S)\pi(S)$$

if the intermediate image is generated stochastically, and

$$L(F|G(S))\pi(S)$$

if deterministic. Note that sometimes the construction of an intermediate image is not necessary and we have

$$\pi(S|F) \propto L(F|S)\pi(S).$$

In all these cases the solution to maximising the expression with respect to S and G is the maximum a posteriori (MAP) estimate of the true scene. The MAP is found either by a global search (which is often impractical due to the large number of parameters) or by techniques such as simulated annealing (Geman and Geman, 1984) or iterative conditional modes (ICM) (Besag, 1986). Alternatively, Markov chain Monte Carlo (MCMC) algorithms provide efficient techniques for simulating from any arbitrary posterior density.

TEXT / REFERENCE BOOKS

1. Donald D Hearn, M. Pauline Baker, Computer Graphics C version, Pearson Education.
2. Dave Shreiner, Mason Woo, Jackie Neider, Tom Davis, OpenGL Programming Guide: The Official Guide to Learning OpenGL, (2013).
3. James D. Foley, Andries van Dam, Steven K. Feiner, John F. Hughes, Computer Graphics: Principles & Practice in C, Addison Wesley Longman.
4. Zhigang Xiang, Roy A Plastock, Computer Graphics, Schaums Outline, TMH.