UNIT-I

DIGITAL IMAGE FUNDAMENTALS AND TRANSFORMS

1. ELEMENTS OF VISUAL PERCEPTION

1.1 ELEMENTS OF HUMAN VISUAL SYSTEMS

- The following figure shows the anatomy of the human eye in cross section

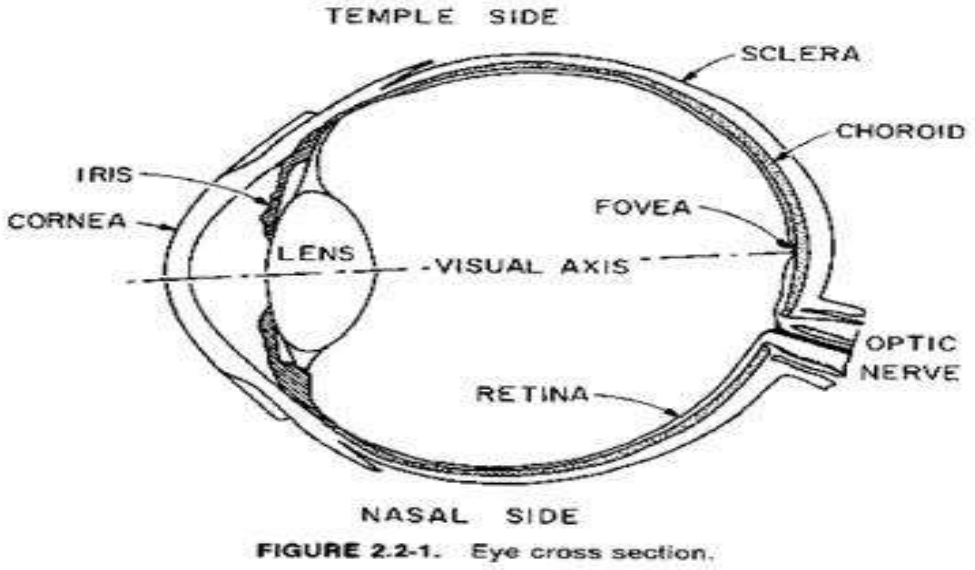


FIGURE 2.2-1. Eye cross section.

- There are two types of receptors in the retina
  - The rods are long slender receptors
  - The cones are generally shorter and thicker in structure
- The rods and cones are not distributed evenly around the retina.
- Rods and cones operate differently
  - Rods are more sensitive to light than cones.
  - At low levels of illumination the rods provide a visual response called scotopic vision
  - Cones respond to higher levels of illumination; their response is called photopic vision

Rods are more sensitive to light than the cones.

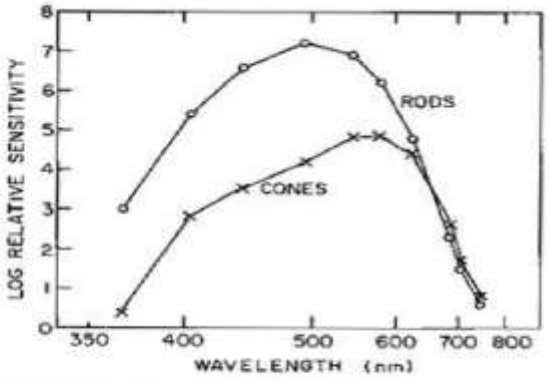
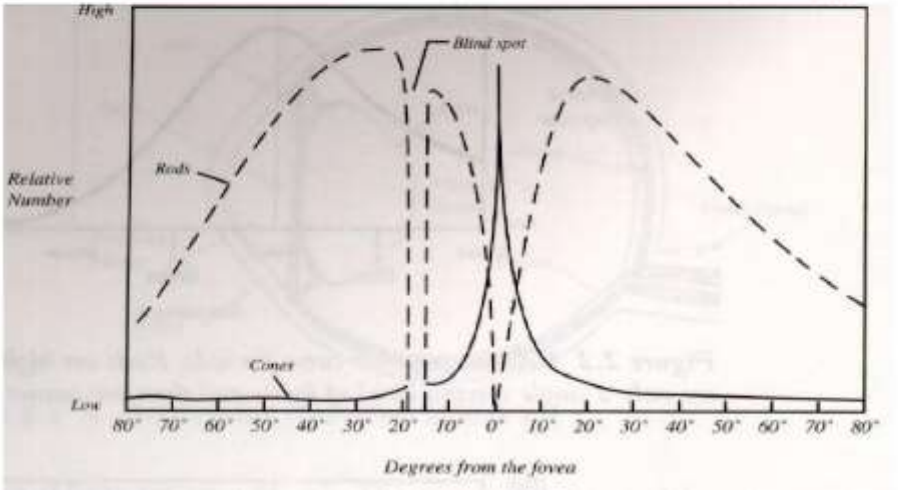


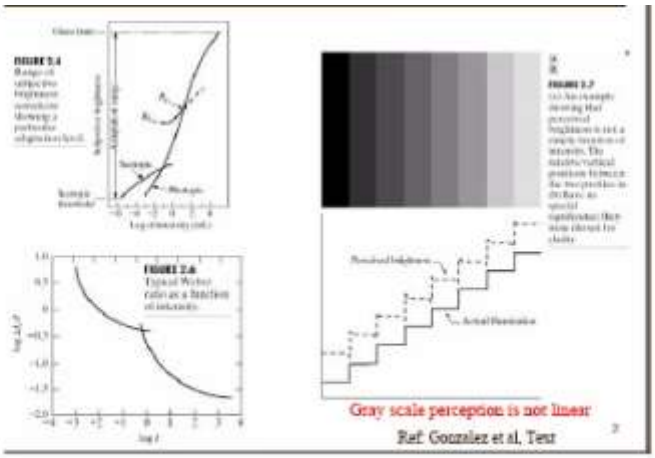
FIGURE 2.2-2. Sensitivity of rods and cones (7) [based upon measurements by Wald (8)].

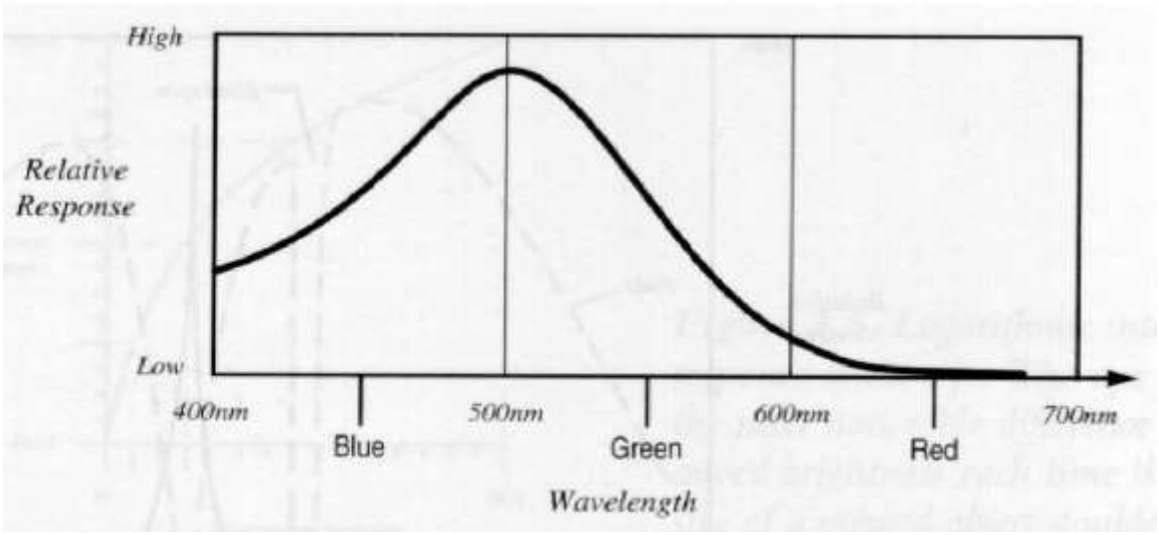


- There are three basic types of cones in the retina
- These cones have different absorption characteristics as a function of wavelength with peak absorptions in the red, green, and blue regions of the optical spectrum.
- is blue, b is green, and g is red

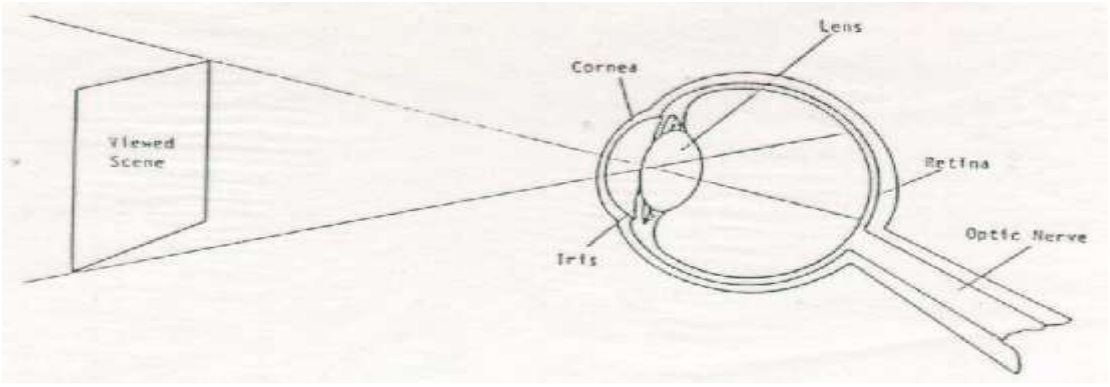
Most of the cones are at the fovea. Rods are spread just about everywhere except the fovea

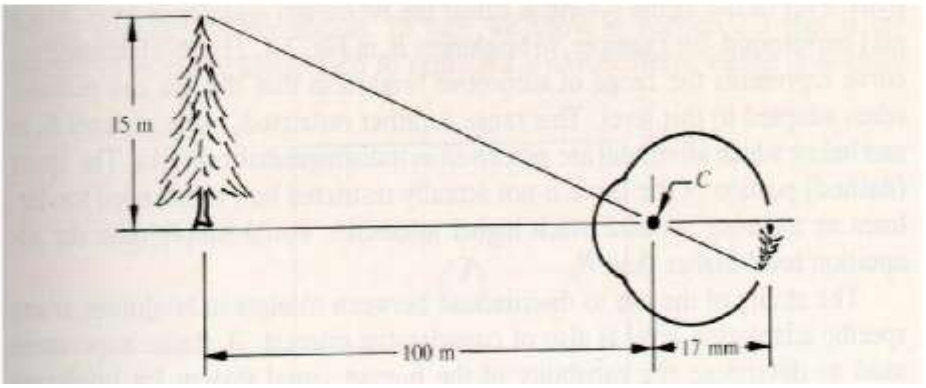
- There is a relatively low sensitivity to blue light. There is a lot of overlap


Gray scale perception is not linear
Ref. Gonzalez et al, Text


High
Relative
Response
Low
400nm
500nm
600nm
700nm
Blue
Green
Red
Wavelength

1.2 IMAGE FORMATION IN THE EYE




15 m
C
100 m
17 mm

1.3 CONTRAST SENSITIVITY

- The response of the eye to changes in the intensity of illumination is nonlinear
- Consider a patch of light of intensity i+dI surrounded by a background intensity I as shown in the following figure
- Over a wide range of intensities, it is found that the ratio dI/I, called the Weber fraction, is nearly constant at a value of about 0.02.
- This does not hold at very low or very high intensities
- Furthermore, contrast sensitivity is dependent on the intensity of the surround. Consider the second panel of the previous figure.

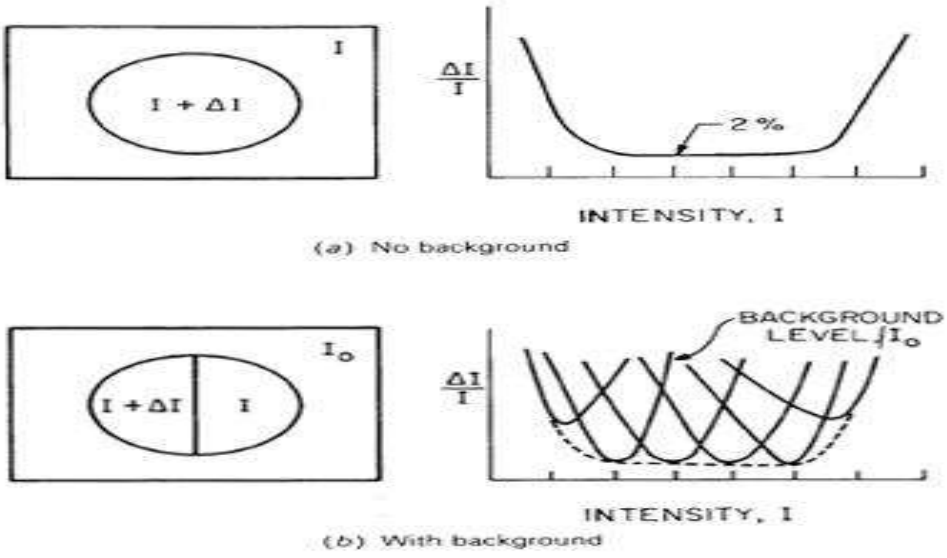


FIGURE 2.3-1. Contrast sensitivity measurements.

1.4 LOGARITHMIC RESPONSE OF CONES AND RODS

- The response of the cones and rods to light is nonlinear. In fact many image processing systems assume that the eye's response is logarithmic instead of linear with respect to intensity.

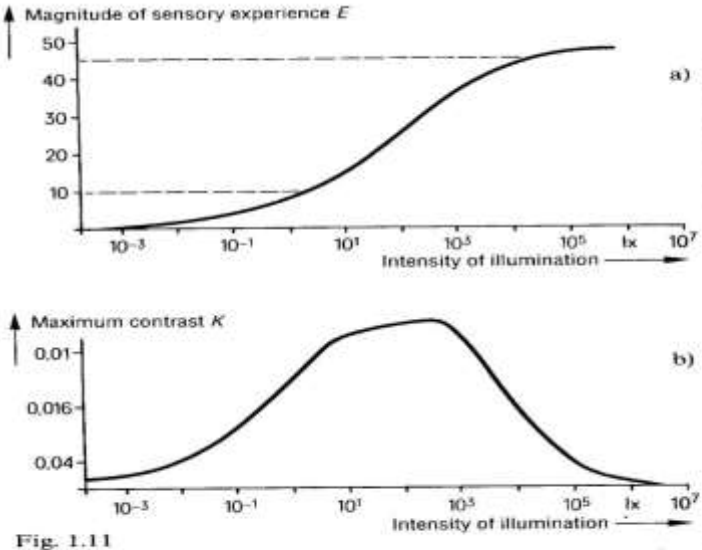
- To test the hypothesis that the response of the cones and rods are logarithmic, we examine the following two cases:

- If the intensity response of the receptors to intensity is linear, then the derivative of the response with respect to intensity should be a constant. This is not the case as seen in the next figure.

- To show that the response to intensity is logarithmic, we take the logarithm of the intensity response and then take the derivative with respect to intensity. This derivative is nearly a constant proving that intensity response of cones and rods can be modeled as a logarithmic response.

- Another way to see this is the following, note that the differential of the logarithm of intensity is d(log(I)) = dI/I. Figure 2.3-1 shows the plot of dI/I for the intensity response of the human visual system.

- Since this plot is nearly constant in the middle frequencies, we again conclude that the intensity response of cones and rods can be modeled as a logarithmic response.



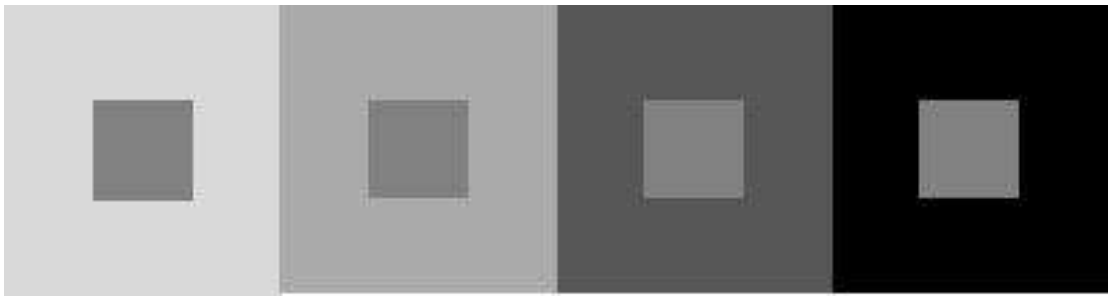
a) The magnitude of sensory experience E is represented along the ordinate as the function $K \ln^{R}/R_0$ for the example of the eye. According to the graph, the eye can distinguish 50 equally-spaced levels of sensory experience. Of these, roughly 35 lie in the linear region

b) This graph shows the actual sensitivity of the eye to difference as measured by experiment. If the Weber-Fechner law were to hold exactly (rising straight line in a)) this curve (taken from Ranke [1.1]) would be parallel to the x-axis

Fig. 1.11
Magnitude of sensory experience according to the Weber-Fechner law (a) and the actually measured sensitivity of the eye to difference (b)

1.5 SIMULTANEOUS CONTRAST

- The simultaneous contrast phenomenon is illustrated below.
- The small squares in each image are the same intensity.
- Because the different background intensities, the small squares do not appear equally bright.
- Perceiving the two squares on different backgrounds as different, even though they are in fact identical, is called the simultaneous contrast effect.
- Psychophysically, we say this effect is caused by the difference in the backgrounds, but what is the physiological mechanism behind this effect?



1.6 LATERAL INHIBITION

- Record signal from nerve fiber of receptor A.
- Illumination of receptor A alone causes a large response.
- Add illumination to three nearby receptors at B causes the response at A to decrease.
- Increasing the illumination of B further decreases A‘s response.
- Thus, illumination of the neighboring receptors inhibited the firing of receptor A.
- This inhibition is called lateral inhibition because it is transmitted laterally, across the retina, in a structure called the lateral plexus.
- A neural signal is assumed to be generated by a weighted contribution of many spatially adjacent rods and cones.
- Some receptors exert an inhibitory influence on the neural response.
- The weighting values are, in effect, the impulse response of the human visual system beyond the retina.

•

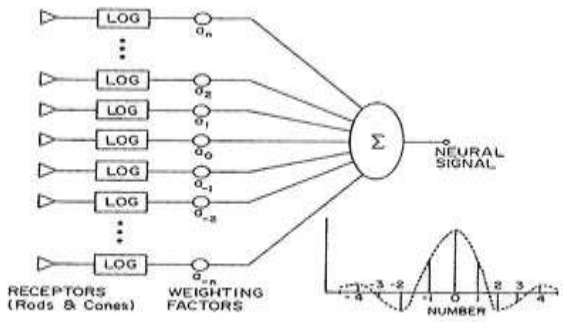
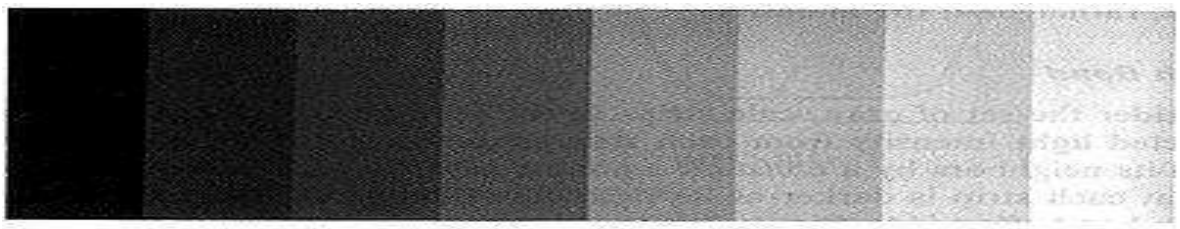


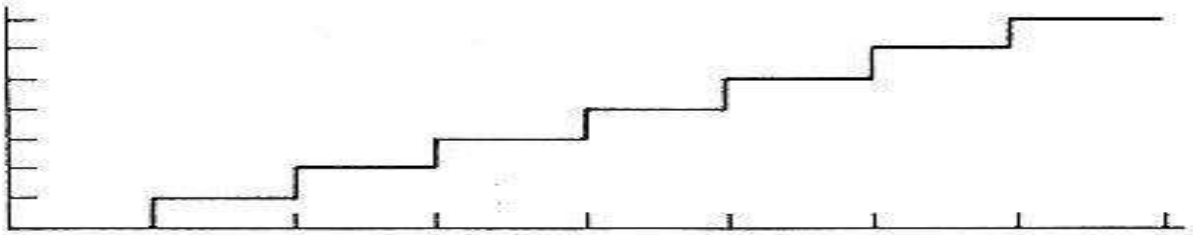
FIGURE 2.4-6. Lateral inhibition.

1.7 MACH BAND EFFECT

- Another effect that can be explained by the lateral inhibition.
- The Mach band effect is illustrated in the figure below.
- The intensity is uniform over the width of each bar.
- However, the visual appearance is that each strip is darker at its right side than its left.



(a) Step chart photo



(b) Step chart intensity distribution

1.8 MACH BAND

- The Mach band effect is illustrated in the figure below.
- A bright bar appears at position B and a dark bar appears at D.

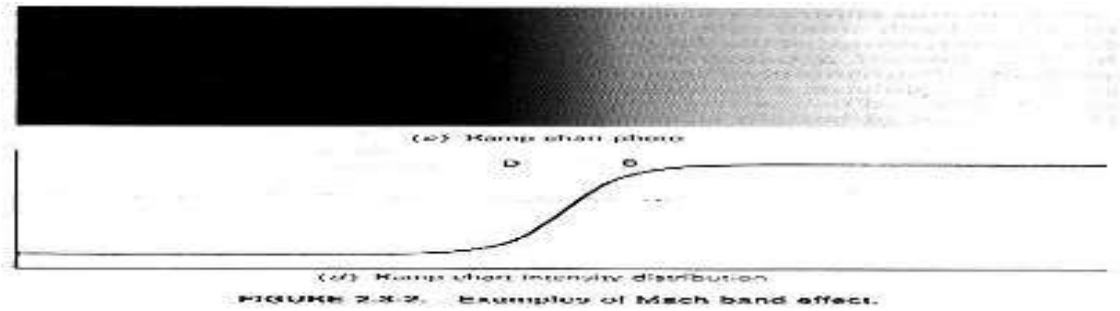


FIGURE 2.3.2. Examples of Mach band effect.

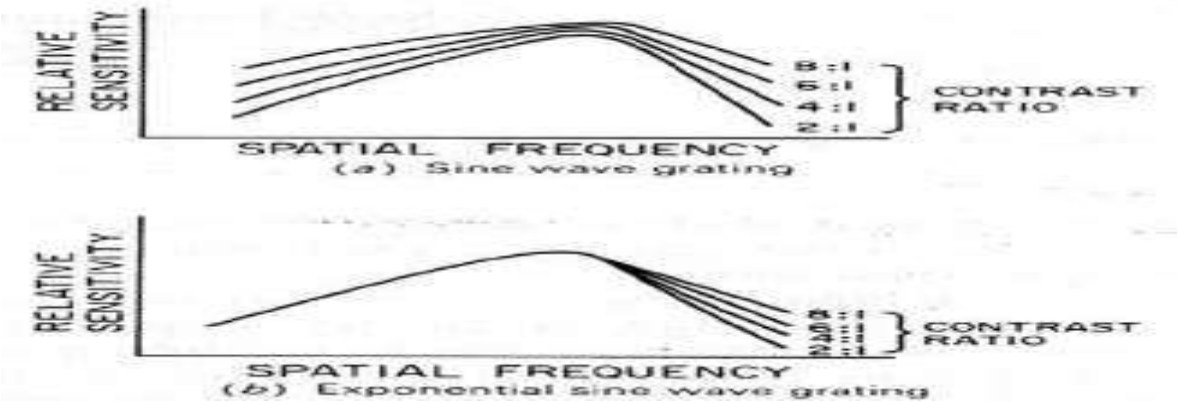
1.9 MODULATION TRANSFER FUNCTION (MTF) EXPERIMENT

- An observer is shown two sine wave grating transparencies, a reference grating of constant contrast and spatial frequency, and a variable-contrast test grating whose spatial frequency is set at some value different from that of the reference.
- Contrast is defined as the ratio

  (max-min)/(max+min)

  where max and min are the maximum and minimum of the grating intensity, respectively.

- The contrast of the test grating is varied until the brightness of the bright and dark regions of the two transparencies appear identical.
- In this manner it is possible to develop a plot of the MTF of the human visual system.
- Note that the response is nearly linear for an exponential sine wave grating.



1.10 MONOCHROME VISION MODEL

- The logarithmic/linear system eye model provides a reasonable prediction of visual response over a wide range of intensities.
- However, at high spatial frequencies and at very low or very high intensities, observed responses depart from responses predicted by the model.

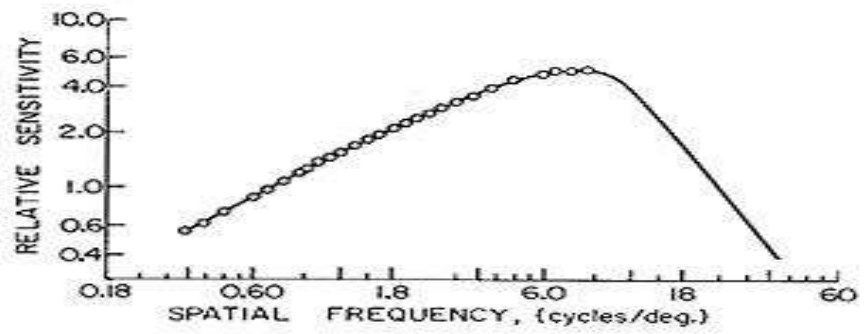
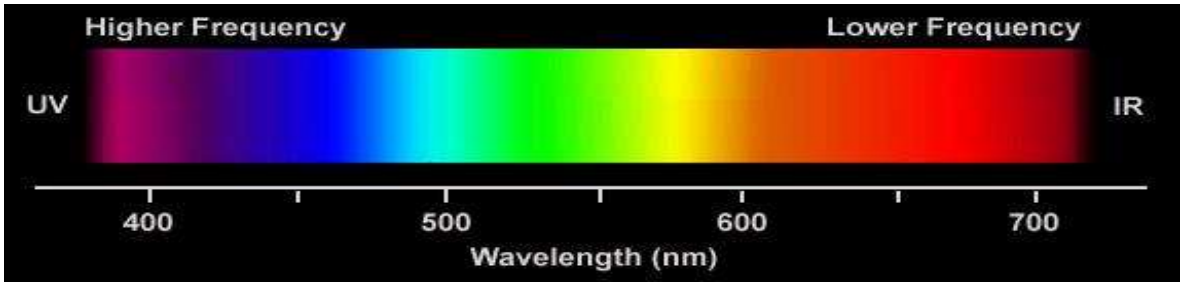


FIGURE 2.4-5. MTF measurement with exponential sine wave grating (27).

1.11 LIGHT

- Light exhibits some properties that make it appear to consist of particles; at other times, it behaves like a wave.
- Light is electromagnetic energy that radiates from a source of energy (or a source of light) in the form of waves
- Visible light is in the 400 nm – 700 nm range of electromagnetic spectrum



1.11.1 INTENSITY OF LIGHT

- The strength of the radiation from a light source is measured using the unit called the candela, or candle power. The total energy from the light source, including heat and all electromagnetic radiation, is called radiance and is usually expressed in watts.
- *Luminance* is a measure of the light strength that is actually perceived by the human eye. Radiance is a measure of the total output of the source; luminance

  measures just the portion that is perceived.

Brightness is a subjective, psychological measure of perceived intensity. Brightness is practically impossible to measure objectively. It is relative. For example, a burning candle in a darkened room will appear bright to the viewer; it will not appear bright in full sunshine.

- The strength of light diminishes in inverse square proportion to its distance from its source. This effect accounts for the need for high intensity projectors for showing multimedia productions on a screen to an audience. Human light perception is sensitive but not linear

2. SAMPLING

Both sounds and images can be considered as signals, in one or two dimensions, respectively. Sound can be described as a fluctuation of the acoustic pressure in time, while images are spatial distributions of values of luminance or color, the latter being described in its RGB or HSB components. Any signal, in order to be processed by numerical computing devices, have to be reduced to a sequence of discrete *samples*, and each sample must be represented using a finite number of bits. The first operation is called *sampling*, and the second operation is called *quantization* of the domain of real numbers.

2.1 1-D: Sounds

Sampling is, for one-dimensional signals, the operation that transforms a continuous-time signal (such as, for instance, the air pressure fluctuation at the entrance of the ear canal) into a discrete-time signal, that is a sequence of numbers. The discrete-time signal gives the values of the continuous-time signal read at intervals of T seconds. The reciprocal of the sampling interval is called *sampling rate* $F_S = 1/T$

. In this module we do not explain the theory of sampling, but we rather describe its manifestations. For a a more extensive yet accessible treatment, we point to the *Introduction to Sound Processing*. For our purposes, the process of sampling a 1-D signal can be reduced to three facts and a theorem.

- **Fact 1:** The Fourier Transform of a discrete-time signal is a function (called *spectrum*) of the continuous variable ω, and it is periodic with period 2π. Given a value of ω, the Fourier transform gives back a complex number that can be interpreted as magnitude and phase (translation in time) of the sinusoidal component at that frequency.

- **Fact 2:** Sampling the continuous-time signal $x(t)$ with interval T we get the discrete-time signal $x(n) = x(nT)$, which is a function of the discrete variable n.

- **Fact 3:** Sampling a continuous-time signal with sampling rate F_S produces a discrete-time signal whose frequency spectrum is the periodic replication of the original signal, and the replication period is F_S. The Fourier variable ω for functions of discrete variable is converted into the frequency variable f (in Hertz) by means of

- $f = \omega / 2\pi T$

The Figure 1 shows an example of frequency spectrum of a signal sampled with sampling rate F_S. In the example, the continuous-time signal had all and only the frequency components between $-F_b$ and F_b. The replicas of the original spectrum are sometimes called *images*.

Frequency spectrum of a sampled signal

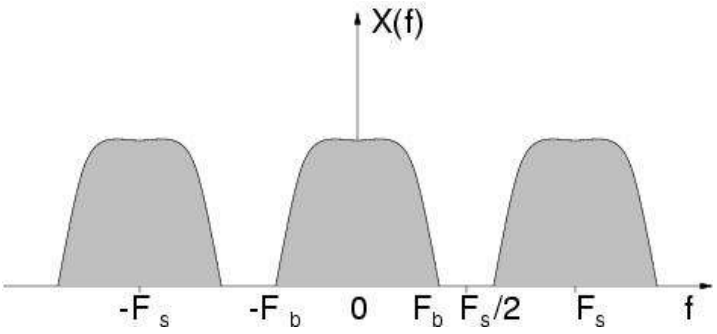


Figure 1

of the Sampling Theorem, historically attributed to the scientists Nyquist and Shannon.

2.2 2-D: Images

Let us assume we have a continuous distribution, on a plane, of values of luminance or, more simply stated, an image. In order to process it using a computer we have to reduce it to a sequence of numbers by means of sampling. There are several ways to sample an image, or read its values of luminance at discrete points. The simplest way is to use a regular grid, with spatial steps X e Y. Similarly to what we did for sounds, we define the spatial sampling rates $F_X = 1/X$

$F_Y = 1/Y$

As in the one-dimensional case, also for two-dimensional signals, or images, sampling can be described by three facts and a theorem.

- **Fact 1:** The Fourier Transform of a discrete-space signal is a function (called *spectrum*) of two continuous variables ω_X and ω_Y, and it is periodic in two dimensions with periods 2π. Given a couple of values ω_X and ω_Y, the Fourier transform gives back a complex number that can be interpreted as magnitude and phase (translation in space) of the sinusoidal component at such spatial

  frequencies.

- **Fact 2:** Sampling the continuous-space signal $s(x,y)$ with the regular grid of steps X, Y, gives a discrete-space signal $s(m,n) = s(mX,nY)$, which is a function of the discrete variables m and n.

  Fact 3: Sampling a continuous-space signal with spatial frequencies F_X and F_Y gives a discrete-space signal whose spectrum is the periodic replication along the grid of steps F_X and F_Y of the original signal spectrum. The Fourier variables ω_X and ω_Y correspond to the frequencies (in cycles per meter) represented by the variables $f_X = \omega_X / 2\pi X$

- And $f_y = \omega_Y / 2\pi Y$

. The Figure 2 shows an example of spectrum of a two-dimensional sampled signal. There, the continuous-space signal had all and only the frequency components included in the central hexagon. The hexagonal shape of the spectral support (region of non-null spectral energy) is merely illustrative. The replicas of the original spectrum are often called spectral *images*.

Spectrum of a sampled image

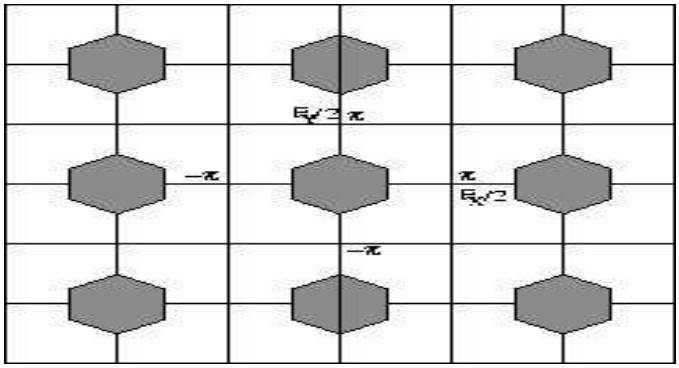


Figure 2

Given the above facts, we can have an intuitive understanding of the Sampling Theorem.

3. QUANTIZATION

With the adjective "digital" we indicate those systems that work on signals that are represented by numbers, with the (finite) precision that computing systems allow. Up to now we have considered discrete-time and discrete-space signals as if they were collections of infinite-precision numbers, or real numbers. Unfortunately, computers only allow to represent finite subsets of rational numbers. This means that our signals are subject to quantization.

For our purposes, the most interesting quantization is the linear one, which is usually occurring in the process of conversion of an analog signal into the digital domain. If the memory word dedicated to storing a number is made of b bits, then the range of such number is discretized into 2^b quantization levels. Any value that is found between two quantization levels can be approximated by truncation or rounding to the closest value. The Figure 3 shows an example of quantization with representation on 3 bits in

two's complement.

Sampling and quantization of an analog signal

The approximation introduced by quantization manifests itself as a noise, called *quantization noise*. Often, for the analysis of sound-processing circuits, such noise is assumed to be white and de-correlated with the signal, but in reality it is perceptually tied to the signal itself, in such an extent that quantization can be perceived as an effect.

To have a visual and intuitive exploration of the phenomenon of quantization, consider the applet that allows to vary between 1 and 8 the number of bits dedicated to the representation of each of the RGB channels representing color. The same number of bits is dedicated to the representation of an audio signal coupled to the image. The visual effect that is obtained by reducing the number of bits is similar to a *solarization*.

4. BASIC RELATIONSHIP BETWEEN PIXELS

4.1 PIXEL

In digital imaging, a **pixel** (or **picture element**) is a single point in a raster image. The pixel is the smallest addressable screen element; it is the smallest unit of picture that can be controlled. Each pixel has its own address. The address of a pixel corresponds to its coordinates. Pixels are normally arranged in a 2-dimensional grid, and are often represented using dots or squares. Each pixel is a sample of an original image; more samples typically provide more accurate representations of the original. The intensity of each pixel is variable. In color image systems, a color is typically represented by three or four component intensities such as red, green, and blue, or cyan, magenta, yellow, and black.

The word *pixel* is based on a contraction of *pix* ("pictures") and *el* (for "element"); similar formations with *el* for "element" include the words: voxel and texel.

4.2 Bits per pixel

The number of distinct colors that can be represented by a pixel depends on the number of bits per pixel (bpp). A 1 bpp image uses 1-bit for each pixel, so each pixel can be either on or off. Each additional bit doubles the number of colors available, so a 2 bpp image can have 4 colors, and a 3 bpp image can have 8 colors:

- 1 bpp, 2^1 = 2 colors (monochrome)
- 2 bpp, 2^2 = 4 colors
- 3 bpp, 2^3 = 8 colors
- 8 bpp, 2^8 = 256 colors

- 16 bpp, 2^{16} = 65,536 colors ("Highcolor")
- 24 bpp, $2^{24} \approx$ 16.8 million colors ("Truecolor")

For color depths of 15 or more bits per pixel, the depth is normally the sum of the bits allocated to each of the red, green, and blue components. Highcolor, usually meaning 16 bpp, normally has five bits for red and blue, and six bits for green, as the human eye is more sensitive to errors in green than in the other two primary colors. For applications involving transparency, the 16 bits may be divided into five bits each of red, green, and available: this means that each 24-bit pixel has an extra 8 bits to describe its blue, with one bit left for transparency. A 24-bit depth allows 8 bits per component. On some systems, 32-bit depth is opacity (for purposes of combining with another image).

Selected standard display resolutions include:

Name	Megapixels	Width x Height
CGA	0.064	320×200
EGA	0.224	640×350
VGA	0.3	640×480
SVGA	0.5	800×600
XGA	0.8	1024×768
SXGA	1.3	1280×1024
UXGA	1.9	1600×1200
WUXGA	2.3	1920×1200

5. BASIC GEOMETRIC TRANSFORMATIONS

Transform theory plays a fundamental role in image processing, as working with the transform of an image instead of the image itself may give us more insight into the properties of the image. Two dimensional transforms are applied to image enhancement, restoration, encoding and description.

5.1. UNITARY TRANSFORMS

5.1.1 One dimensional signals

For a one dimensional sequence $\{ f(x), 0 \le x \le N-1 \}$ represented as a vector $\underline{f} = [f(0)\ f(1)\ \ f(N-1)]^T$ of size N, a transformation may be written as

$$\underline{g} = \underline{T} \cdot \underline{f} \Rightarrow g(u) = \sum_{x=0}^{N-1} T(u, x)\, f(x),\ 0 \le u \le N-1$$

where $g(u)$ is the transform (or transformation) of $f(x)$, and $T(u, x)$ is the so called **forward transformation kernel**. Similarly, the inverse transform is the relation

$$f(x) = \sum_{u=0}^{N-1} I(x,u) g(u),\ 0 \le x \le N-1$$

or written in a matrix form

$$\underline{f} = \underline{I} \cdot \underline{g} = \underline{T}^{-1} \cdot \underline{g}$$

where $I(x,u)$ is the so called **inverse transformation kernel**.

If

$$\underline{I} = \underline{T}^{-1} = \underline{T}^{*T}$$

the matrix $\underline{T}$ is called **unitary**, and the transformation is called unitary as well. It can be

proven (**how?**) that the columns (or rows) of an $N \times N$ unitary matrix are orthonormal and therefore, form a complete set of **basis vectors** in the N – dimensional vector space. In that case

$$f = \underline{T}^{*T} \cdot g \Rightarrow f(x) = \sum_{u=0}^{N-1} T^*(u, x) g(u)$$

The columns of $\underline{T}^{*T}$, that is, the vectors $\underline{T}^*_u = \left[T^*(u,0)\ T^*(u,1)\ T^*(u, N-1) \right]^T$ are called the **basis vectors** of $\underline{T}$.

5.1.2 Two dimensional signals (images)

As a one dimensional signal can be represented by an orthonormal set of **basis vectors**, an image can also be expanded in terms of a discrete set of **basis arrays** called basis images through a **two dimensional (image) transform**.

For an $N \times N$ image $f(x, y)$ the forward and inverse transforms are given below

$$g(u, v) = \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} T(u, v, x, y) f(x, y)$$

$$f(x, y) = \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} I(x, y, u, v) g(u, v)$$

where, again, $T(u, v, x, y)$ and $I(x, y, u, v)$ are called the **forward and inverse transformation kernels**, respectively.

The forward kernel is said to be **separable** if

$$T(u, v, x, y) = T_1(u, x) T_2(v, y)$$

It is said to be **symmetric** if T_1 is functionally equal to T_2 such that

$$T(u, v, x, y) = T_1(u, x) T_1(v, y)$$

The same comments are valid for the inverse kernel.

If the kernel $T(u, v, x, y)$ of an image transform is separable and symmetric, then the transform $g(u, v) = \sum_{x=0}^{N-1}\sum_{y=0}^{N-1} T(u, v, x, y) f(x, y) = \sum_{x=0}^{N-1}\sum_{y=0}^{N-1} T_1(u, x) T_1(v, y) f(x, y)$ can be written in matrix form as follows

$$\underline{g} = \underline{T}_1 \cdot \underline{f} \cdot \underline{T}_1^T$$

where f is the original image of size $N \times N$, and $\underline{T}_1$ is an $N \times N$ transformation matrix with elements $t_{ij} = T_1(i, j)$. If, in addition, $\underline{T}_1$ is a unitary matrix then the transform is called **separable unitary** and the original image is recovered through the relationship

$$\underline{f} = \underline{T}_1^{*T} \cdot \underline{g} \cdot \underline{T}_1^*$$

5.1.3 Fundamental properties of unitary transforms

5.1.3.1 The property of energy preservation

In the unitary transformation

$$\underline{g} = \underline{T} \cdot \underline{f}$$

it is easily proven (try the proof by using the relation $\underline{T}^{-1} = \underline{T}^{*T}$) that

$$\|\underline{g}\|^2 = \|\underline{f}\|^2$$

Thus, a unitary transformation preserves the signal energy. This property is called energy preservation property.

This means that every unitary transformation is simply a rotation of the vector $\underline{f}$ in the

N - dimensional vector space.

For the 2-D case the energy preservation property is written as

$$\sum_{x=0}^{N-1}\sum_{y=0}^{N-1}\left|f(x,y)\right|^2=\sum_{u=0}^{N-1}\sum_{v=0}^{N-1}\left|g(u,v)\right|^2$$

5.1.3.2 The property of energy compaction

Most unitary transforms pack a large fraction of the energy of the image into relatively few of the transform coefficients. This means that relatively few of the transform coefficients have significant values and these are the coefficients that are close to the origin (small index coefficients).

This property is very useful for compression purposes. (**Why?**)

5.2. THE TWO DIMENSIONAL FOURIER TRANSFORM

5.2.1 Continuous space and continuous frequency

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.

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

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

$$P(u, v) = |F(u, v)|^2 = R^2(u, v) + I^2(u, v)$$

5.2.2 Discrete space and continuous frequency

For the case of a discrete sequence $f(x, y)$ of infinite duration we can define the 2-D discrete space Fourier transform pair as follows

$$F(u, v) = \sum_{x=-\infty}^{\infty} \sum_{y=-\infty}^{\infty} f(x, y) e^{-j(xu+vy)}$$

$$f(x, y) = \frac{1}{(2\pi)^2} \int_{u=-\pi}^{\pi} \int_{v=-\pi}^{\pi} F(u, v) e^{j(xu+vy)} du dv$$

$F(u, v)$ is again a complex-valued function of two real frequency variables u, v and it is periodic with a period $2\pi \times 2\pi$, that is to say $F(u, v) = F(u + 2\pi, v) = F(u, v + 2\pi)$ The Fourier transform of $f(x, y)$ is said to converge uniformly when $F(u, v)$ is finite and

$$\lim_{N_1 \to \infty} \lim_{N_2 \to \infty} \sum_{x=-N_1}^{N_1} \sum_{y=-N_2}^{N_2} f(x, y) e^{-j(xu+vy)} = F(u, v) \text{ for all } u, v.$$

When the Fourier transform of $f(x, y)$ converges uniformly, $F(u, v)$ is an analytic function and is infinitely differentiable with respect to u and v.

5.2.3 Discrete space and discrete frequency: The two dimensional Discrete Fourier Transform (2-D DFT)

If $f(x, y)$ is an $M \times N$ array, such as that obtained by sampling a continuous

function of two dimensions at dimensions M and N on a rectangular grid, then its two dimensional Discrete Fourier transform (DFT) is the array given by

$$F(u, v) = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y) e^{-j 2\pi (ux/M + vy/N)}$$

$u = 0, \ldots, M-1, v = 0, \ldots, N-1$

and the inverse DFT (IDFT) is

$$f(x, y) = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F(u, v) e^{j 2\pi (ux/M + vy/N)}$$

When images are sampled in a square array, $M = N$ and

$$F(u, v) = \frac{1}{N} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x, y) e^{-j 2\pi (ux + vy)/N}$$

$$f(x, y) = \frac{1}{N} \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} F(u, v) e^{j 2\pi (ux + vy)/N}$$

It is straightforward to prove that the two dimensional Discrete Fourier Transform is

separable, symmetric and unitary.

5.2.3.1 Properties of the 2-D DFT

Most of them are straightforward extensions of the properties of the 1-D Fourier Transform. Advise any introductory book on Image Processing.

Completeness

The discrete Fourier transform is an invertible, linear transformation

with **C** denoting the set of complex numbers. In other words, for any $N > 0$, an N-dimensional complex vector has a DFT and an IDFT which are in turn N-dimensional complex vectors.

Orthogonality

The vectors $e^{\frac{2\pi i}{N}kn}$ form an orthogonal basis over the set of N-dimensional complex vectors:

$$\sum_{n=0}^{N-1} \left(e^{\frac{2\pi i}{N}kn}\right) \left(e^{-\frac{2\pi i}{N}k'n}\right) = N\ \delta_{kk'}$$

where $\delta_{kk'}$ is the Kronecker delta. This orthogonality condition can be used to derive the formula for the IDFT from the definition of the DFT, and is equivalent to the unitarity property below.

The Plancherel theorem and Parseval's theorem

If X_k and Y_k are the DFTs of x_n and y_n respectively then the Plancherel theorem states:

$$\sum_{n=0}^{N-1} x_n y_n^* = \frac{1}{N} \sum_{k=0}^{N-1} X_k Y_k^*$$

where the star denotes complex conjugation. Parseval's theorem is a special case of the Plancherel theorem and states:

$$\sum_{n=0}^{N-1} |x_n|^2 = \frac{1}{N} \sum_{k=0}^{N-1} |X_k|^2.$$

These theorems are also equivalent to the unitary condition below.

Periodicity

If the expression that defines the DFT is evaluated for all integers *k* instead of just for $k = 0, \ldots, N-1$, then the resulting infinite sequence is a periodic extension of the DFT, periodic with period *N*.

The periodicity can be shown directly from the definition:

Similarly, it can be shown that the IDFT formula leads to a periodic extension.

The shift theorem

Multiplying x_n by a *linear phase* $e^{\frac{2\pi i}{N}nm}$ for some integer *m* corresponds to a *circular shift* of the output X_k: X_k is replaced by X_{k-m}, where the subscript is interpreted modulo *N* (i.e., periodically). Similarly, a circular shift of the input x_n corresponds to multiplying the output X_k by a linear phase. Mathematically, if $\{x_n\}$ represents the vector **x** then

if $\mathcal{F}(\{x_n\})_k = X_k$

then $\mathcal{F}(\{x_n \cdot e^{\frac{2\pi i}{N}nm}\})_k = X_{k-m}$

and

Circular convolution theorem and cross-correlation theorem

The convolution theorem for the continuous and discrete time Fourier transforms indicates that a convolution of two infinite sequences can be obtained as the inverse transform of the product of the individual transforms. With sequences and transforms of length N, a circularity arises**:**

$$\begin{aligned}
\mathcal{F}^{-1}\{\mathbf{X}\cdot\mathbf{Y}\}_n &\stackrel{\text{def}}{=} \frac{1}{N}\sum_{k=0}^{N-1} X_k \cdot Y_k \cdot e^{\frac{2\pi i}{N}kn} \\
&= \frac{1}{N}\sum_{k=0}^{N-1}\left(\sum_{l=0}^{N-1} x_l e^{-\frac{2\pi i}{N}kl}\right)\cdot\left(\sum_{m=0}^{N-1} y_m e^{-\frac{2\pi i}{N}km}\right)\cdot e^{\frac{2\pi i}{N}kn} \\
&= \sum_{l=0}^{N-1} x_l \sum_{m=0}^{N-1} y_m \left(\frac{1}{N}\sum_{k=0}^{N-1} e^{\frac{2\pi i}{N}k(n-l-m)}\right).
\end{aligned}$$

The quantity in parentheses is 0 for all values of m except those of the form $n - l - pN$, where p is any integer. At those values, it is 1. It can therefore be replaced by an infinite sum of Kronecker delta functions, and we continue accordingly. Note that we can also extend the limits of m to infinity, with the understanding that the x and y sequences are defined as 0 outside [0,N-1]**:**

which is the convolution of the $\mathbf{x}$sequence with a periodically extended $\mathbf{y}$sequence defined by**:**

$$(\mathbf{y_N})_n \stackrel{\text{def}}{=} \sum_{p=-\infty}^{\infty} y_{(n-pN)}.$$

Similarly, it can be shown that**:**

$$\mathcal{F}^{-1}\{\mathbf{X}^* \cdot \mathbf{Y}\}_n = \sum_{l=0}^{N-1} x_l^* \cdot (y_N)_{n+l} \stackrel{\text{def}}{=} (\mathbf{x} \star \mathbf{y_N})_n \ ,$$

which is the cross-correlation of $\mathbf{x}$ and $\mathbf{y_N}$.

A direct evaluation of the convolution or correlation summation (above) requires $O(N^2)$ operations for an output sequence of length N. An indirect method, using transforms, can take advantage of the $O(N\log N)$ efficiency of the fast Fourier transform (FFT) to achieve much better performance. Furthermore, convolutions can be used to efficiently compute DFTs via Rader's FFT algorithm and Bluestein's FFT algorithm.

Methods have also been developed to use circular convolution as part of an efficient process that achieves normal (non-circular) convolution with an $\mathbf{x}$or $\mathbf{y}$sequence potentially much longer than the practical transform size (**N**). Two such methods are called overlap-save and overlap-add[1].

Convolution theorem duality

It can also be shown that**:**

$$\mathcal{F}\{\mathbf{x} \cdot \mathbf{y}\}_k \stackrel{\text{def}}{=} \sum_{n=0}^{N-1} x_n \cdot y_n \cdot e^{-\frac{2\pi i}{N} kn}$$
$$= \frac{1}{N}(\mathbf{X} * \mathbf{Y_N})_k,$$

which is the circular convolution of $\mathbf{X}$ and $\mathbf{Y}$.

Trigonometric interpolation polynomial

The trigonometric interpolation polynomial

$$p(t) = \frac{1}{N}\left[X_0 + X_1 e^{it} + \cdots + X_{N/2-1} e^{(N/2-1)it} + X_{N/2}\cos(Nt/2) + X_{N/2+1} e^{(-}\right.$$

for N even ,

$$p(t) = \frac{1}{N}\left[X_0 + X_1 e^{it} + \cdots + X_{\lfloor N/2\rfloor} e^{\lfloor N/2\rfloor it} + X_{\lfloor N/2\rfloor+1} e^{-\lfloor N/2\rfloor it} + \cdots + X_{N-}\right.$$

for N odd,

where the coefficients X_k are given by the DFT of x_n above, satisfies the interpolation property $p(2\pi n / N) = x_n$ for $n = 0, \ldots, N-1$.

For even N, notice that the Nyquist component $\frac{X_{N/2}}{N}\cos(Nt/2)$ is handled specially.

This interpolation is *not unique*: aliasing implies that one could add N to any of the complex-sinusoid frequencies (e.g. changing e^{-it} to $e^{i(N-1)t}$) without changing the interpolation property, but giving *different* values in between the x_n points. The choice above, however, is typical because it has two useful properties. First, it consists of sinusoids whose frequencies have the smallest possible magnitudes, and therefore

minimizes the mean-square slope $\int |p'(t)|^2 dt$ of the interpolating function. Second, if the x_n are real numbers, then $p(t)$ is real as well.

In contrast, the most obvious trigonometric interpolation polynomial is the one in which the frequencies range from 0 to $N-1$ (instead of roughly $-N/2$ to $+N/2$ as above), similar to the inverse DFT formula. This interpolation does *not* minimize the slope, and is *not* generally real-valued for real x_n; its use is a common mistake.

The unitary DFT

Another way of looking at the DFT is to note that in the above discussion, the DFT can be expressed as a Vandermonde matrix:

$$\mathbf{F} = \begin{bmatrix} \omega_N^{0\cdot 0} & \omega_N^{0\cdot 1} & \dots & \omega_N^{0\cdot (N-1)} \\ \omega_N^{1\cdot 0} & \omega_N^{1\cdot 1} & \dots & \omega_N^{1\cdot (N-1)} \\ \vdots & \vdots & \ddots & \vdots \\ \omega_N^{(N-1)\cdot 0} & \omega_N^{(N-1)\cdot 1} & \dots & \omega_N^{(N-1)\cdot (N-1)} \end{bmatrix}$$

where

$$\omega_N = e^{-2\pi i/N}$$

is a primitive Nth root of unity. The inverse transform is then given by the inverse of the above matrix:

$$\mathbf{F}^{-1} = \frac{1}{N}\mathbf{F}^*$$

With unitary normalization constants $1/\sqrt{N}$, the DFT becomes a unitary transformation, defined by a unitary matrix:

$$\mathbf{U} = \mathbf{F}/\sqrt{N}$$
$$\mathbf{U}^{-1} = \mathbf{U}^*$$
$$|\det(\mathbf{U})| = 1$$

where *det()* is the determinant function. The determinant is the product of the eigenvalues, which are always ± 1 or $\pm i$ as described below. In a real vector space, a unitary transformation can be thought of as simply a rigid rotation of the coordinate system, and all of the properties of a rigid rotation can be found in the unitary DFT.

The orthogonality of the DFT is now expressed as an orthonormality condition (which arises in many areas of mathematics as described in root of unity):

$$\sum_{m=0}^{N-1} U_{km} U_{mn}^* = \delta_{kn}$$

If $\mathbf{X}$ is defined as the unitary DFT of the vector $\mathbf{x}$ then

$$X_k = \sum_{n=0}^{N-1} U_{kn} x_n$$

and the Plancherel theorem is expressed as:

$$\sum_{n=0}^{N-1} x_n y_n^* = \sum_{k=0}^{N-1} X_k Y_k^*$$

If we view the DFT as just a coordinate transformation which simply specifies the components of a vector in a new coordinate system, then the above is just the statement that the dot product of two vectors is preserved under a unitary DFT transformation. For

the special case $\mathbf{x} = \mathbf{y}$, this implies that the length of a vector is preserved as well—this is just Parseval's theorem:

$$\sum_{n=0}^{N-1} |x_n|^2 = \sum_{k=0}^{N-1} |X_k|^2$$

Expressing the inverse DFT in terms of the DFT

A useful property of the DFT is that the inverse DFT can be easily expressed in terms of the (forward) DFT, via several well-known "tricks". (For example, in computations, it is often convenient to only implement a fast Fourier transform corresponding to one transform direction and then to get the other transform direction from the first.)

First, we can compute the inverse DFT by reversing the inputs:

$$\mathcal{F}^{-1}(\{x_n\}) = \mathcal{F}(\{x_{N-n}\})/N$$

(As usual, the subscripts are interpreted modulo N; thus, for $n = 0$, we have $x_{N-0} = x_0$.)

Second, one can also conjugate the inputs and outputs:

$$\mathcal{F}^{-1}(\mathbf{x}) = \mathcal{F}(\mathbf{x}^*)^*/N$$

Third, a variant of this conjugation trick, which is sometimes preferable because it requires no modification of the data values, involves swapping real and imaginary parts (which can be done on a computer simply by modifying pointers). Define swap(x_n) as x_n with its real and imaginary parts swapped—that is, if $x_n = a + bi$ then swap(x_n) is $b + ai$.

Equivalently, swap(x_n) equals ix_n^*. Then

$$\mathcal{F}^{-1}(\mathbf{x}) = \text{swap}(\mathcal{F}(\text{swap}(\mathbf{x})))/N$$

That is, the inverse transform is the same as the forward transform with the real and imaginary parts swapped for both input and output, up to a normalization (Duhamel *et al.*, 1988).

The conjugation trick can also be used to define a new transform, closely related to the DFT, that is involutary—that is, which is its own inverse. In particular, $T(\mathbf{x}) = \mathcal{F}(\mathbf{x}^*)/\sqrt{N}$ is clearly its own inverse: $T(T(\mathbf{x})) = \mathbf{x}$. A closely related involutary transformation (by a factor of $(1+i)/\sqrt{2}$) is $H(\mathbf{x}) = \mathcal{F}((1+i)\mathbf{x}^*)/\sqrt{2N}$, since the $(1 + i)$ factors in $H(H(\mathbf{x}))$ cancel the 2.

For real inputs $\mathbf{x}$, the real part of $H(\mathbf{x})$ is none other than the discrete Hartley transform, which is also involutary.

Eigenvalues and eigenvectors

The eigenvalues of the DFT matrix are simple and well-known, whereas the eigenvectors are complicated, not unique, and are the subject of ongoing research.

Consider the unitary form $\mathbf{U}$ defined above for the DFT of length N, where

$\mathbf{U}_{m,n} = \omega_N^{(m-1)(n-1)} / \sqrt{N} = \exp(-2\pi i(m-1)(n-1)/N)/\sqrt{N}$. This matrix satisfies the equation:

$$\mathbf{U}^4 = \mathbf{I}.$$

This can be seen from the inverse properties above: operating $\mathbf{U}$ twice gives the original data in reverse order, so operating $\mathbf{U}$ four times gives back the original data and is thus the identity matrix. This means that the eigenvalues λ satisfy a characteristic equation:

$\lambda^4 = 1$.

Therefore, the eigenvalues of $\mathbf{U}$ are the fourth roots of unity: λ is +1, −1, +i, or −i.

Since there are only four distinct eigenvalues for this $N \times N$ matrix, they have some multiplicity. The multiplicity gives the number of linearly independent eigenvectors corresponding to each eigenvalue. (Note that there are N independent eigenvectors; a unitary matrix is never defective.)

The problem of their multiplicity was solved by McClellan and Parks (1972), although it was later shown to have been equivalent to a problem solved by Gauss (Dickinson and Steiglitz, 1982). The multiplicity depends on the value of N modulo 4, and is given by the following table:

size *N*	**λ = +1**	**λ = −1**	**λ = -*i***	**λ = +*i***
4*m*	*m* + 1	*m*	*m*	*m* − 1
4*m* + 1	*m* + 1	*m*	*m*	*m*
4*m* + 2	*m* + 1	*m* + 1	*m*	*m*
4*m* + 3	*m* + 1	*m* + 1	*m* + 1	*m*

Multiplicities of the eigenvalues λ of the unitary DFT matrix **U** as a function of the transform size *N* (in terms of an integer *m*).

No simple analytical formula for general eigenvectors is known. Moreover, the eigenvectors are not unique because any linear combination of eigenvectors for the same eigenvalue is also an eigenvector for that eigenvalue. Various researchers have proposed different choices of eigenvectors, selected to satisfy useful properties like orthogonality and to have "simple" forms (e.g., McClellan and Parks, 1972; Dickinson and Steiglitz, 1982; Grünbaum, 1982; Atakishiyev and Wolf, 1997; Candan *et al.*, 2000; Hanna *et al.*, 2004; Gurevich and Hadani, 2008). However two simple closed-form analytical eigenvectors for special DFT period *N* were found (Kong, 2008):

For DFT period $N = 2L + 1 = 4K + 1$, where K is an integer, the following is an eigenvector of DFT:

$$F(m) = \prod_{s=K+1}^{L} \left[cos\left(\frac{2\pi}{N}m\right) - cos\left(\frac{2\pi}{N}s\right) \right]$$

For DFT period $N = 2L = 4K$, where K is an integer, the following is an eigenvector of

DFT:

$$F(m) = sin\left(\frac{2\pi}{N}m\right) \prod_{s=K+1}^{L-1} \left[cos\left(\frac{2\pi}{N}m\right) - cos\left(\frac{2\pi}{N}s\right) \right]$$

The choice of eigenvectors of the DFT matrix has become important in recent years in order to define a discrete analogue of the fractional Fourier transform—the DFT matrix can be taken to fractional powers by exponentiating the eigenvalues (e.g., Rubio and Santhanam, 2005). For the continuous Fourier transform, the natural orthogonal eigenfunctions are the Hermite functions, so various discrete analogues of these have been employed as the eigenvectors of the DFT, such as the Kravchuk polynomials (Atakishiyev and Wolf, 1997). The "best" choice of eigenvectors to define a fractional discrete Fourier transform remains an open question, however.

The real-input DFT

If $x_0, \ldots, x_{N-1}$ are real numbers, as they often are in practical applications, then the DFT obeys the symmetry:

$$X_k = X^*_{N-k}.$$

The star denotes complex conjugation. The subscripts are interpreted modulo *N*.

Therefore, the DFT output for real inputs is half redundant, and one obtains the complete information by only looking at roughly half of the outputs $X_0, \ldots, X_{N-1}$. In this case, the "DC" element X_0 is purely real, and for even *N* the "Nyquist" element $X_{N/2}$ is also real, so there are exactly *N* non-redundant real numbers in the first half + Nyquist element of the complex output *X*.

Using Euler's formula, the interpolating trigonometric polynomial can then be interpreted as a sum of sine and cosine functions.

5.2.3.2 The importance of the phase in 2-D DFT. Image reconstruction from amplitude or phase only.

The Fourier transform of a sequence is, in general, complex-valued, and the unique representation of a sequence in the Fourier transform domain requires both the phase and the magnitude of the Fourier transform. In various contexts it is often desirable to reconstruct a signal from only partial domain information. Consider a 2-D sequence

$f(x, y)$ with Fourier transform $F(u, v) = \Im\{f(x, y)\}$ so that

$$F(u, v) = \{f(x, y)\} = |F(u, v)| e^{j\phi(u,v)}$$

It has been observed that a straightforward signal synthesis from the Fourier transform phase $\phi_f(u, v)$ alone, often captures most of the intelligibility of the original image $f(x, y)$ (**why?**). A straightforward synthesis from the Fourier transform magnitude $|F(u, v)|$ alone, however, does not generally capture the original signal's intelligibility.

The above observation is valid for a large number of signals (or images). To illustrate this, we can synthesise the phase-only signal $f_p(x, y)$ and the magnitude-only signal

$f_m(x,y)$ by

$$f_p(x,y) = \Im^{-1}\left[1e^{j\phi_f(u,v)}\right] f_m(x,y)$$

$$= \Im^{-1}\left[|F(u,v)|\, e^{j0}\right]$$

and observe the two results (**Try this exercise in MATLAB**).

An experiment which more dramatically illustrates the observation that phase-only signal synthesis captures more of the signal intelligibility than magnitude-only synthesis, can be performed as follows.

Consider two images $f(x, y)$ and $g(x, y)$. From these two images, we synthesise two other images $f_1(x, y)$ and $g_1(x, y)$ by mixing the amplitudes and phases of the original images as follows:

$$f_1(x,y) = \Im^{-1}\left[|G(u,v)|\, e^{j\phi_f(u,v)}\right]$$

$$g_1(x,y) = \Im^{-1}\left[|F(u,v)|\, e^{j\phi_g(u,v)}\right]$$

In this experiment $f_1(x, y)$ captures the intelligibility of $f(x, y)$, while $g_1(x, y)$ captures the intelligibility of $g(x, y)$ (**Try this exercise in MATLAB**).

5.3 FAST FOURIER TRANSFORM (FFT)

In this section we present several methods for computing the DFT efficiently. In view of the importance of the DFT in various digital signal processing applications, such as linear filtering, correlation analysis, and spectrum analysis, its efficient computation is a topic that has received considerable attention by many mathematicians, engineers, and applied scientists.

From this point, we change the notation that *X(k)*, instead of *y(k)* in previous sections, represents the Fourier coefficients of *x(n)*.

Basically, the computational problem for the DFT is to compute the sequence $\{X(k)\}$ of N complex-valued numbers given another sequence of data $\{x(n)\}$ of length N, according to the formula

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{kn}, \qquad 0 \le k \le N-1$$

$$W_N = e^{-j2\pi/N}$$

In general, the data sequence $x(n)$ is also assumed to be complex valued. Similarly, The IDFT becomes

$$x(n) = \frac{1}{N}\sum_{n=0}^{N-1} X(k) W_N^{-nk}, \qquad 0 \le n \le N-1$$

Since DFT and IDFT involve basically the same type of computations, our discussion of efficient computational algorithms for the DFT applies as well to the efficient computation of the IDFT.

We observe that for each value of k, direct computation of $X(k)$ involves N complex multiplications ($4N$ real multiplications) and N-1 complex additions ($4N$-2 real additions). Consequently, to compute all N values of the DFT requires N^2 complex multiplications and N^2-N complex additions.

Direct computation of the DFT is basically inefficient primarily because it does not exploit the symmetry and periodicity properties of the phase factor W_N. In particular, these two properties are :

$$\text{Symmetry property}: W_N^{k+N/2} = -W_N^k$$
$$\text{Periodicit y property}: W_N^{k+N} = W_N^k$$

The computationally efficient algorithms described in this section, known collectively as fast Fourier transform (FFT) algorithms, exploit these two basic properties of the phase factor.

6. SEPARABLE IMAGE TRANSFORMS

6.1 THE DISCRETE COSINE TRANSFORM (DCT)

6.1.1 One dimensional signals

This is a transform that is similar to the Fourier transform in the sense that the new independent variable represents again frequency. The DCT is defined below.

$$N-1 \qquad \lceil (2x+1)u\pi \rceil$$

$$C(u) = a(u)\sum_{x=0}^{N-1} f(x)\cos\left[\frac{(2x+1)u\pi}{2N}\right], \quad u = 0,1,\ \ldots,\ N-1$$

with $a(u)$ a parameter that is defined below.

$$a(u) = \begin{cases} \sqrt{1/N} & u = 0 \\ \sqrt{2/N} & u = 1,\ \ldots,\ N-1 \end{cases}$$

The inverse DCT (IDCT) is defined below.

$$f(x) = \sum_{u=0}^{N-1} a(u)C(u)\cos\left[\frac{(2x+1)u\pi}{2N}\right]$$

6.1.2 Two dimensional signals (images)

For 2-D signals it is defined as

$$C(u,v) = a(u)a(v)\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} f(x,y)\cos\left[\frac{(2x+1)u\pi}{2N}\right]\cos\left[\frac{(2y+1)v\pi}{2N}\right]$$

$$f(x,y) = \sum_{u=0}^{N-1}\sum_{v=0}^{N-1} a(u)a(v)C(u,v)\cos\left[\frac{(2x+1)u\pi}{2N}\right]\cos\left[\frac{(2y+1)v\pi}{2N}\right]$$

$a(u)$ is defined as above and $u, v = 0,1,\ \ldots,\ N-1$

6.1.3 Properties of the DCT transform

- The DCT is a real transform. This property makes it attractive in comparison to the Fourier transform.
- The DCT has excellent energy compaction properties. For that reason it is widely used in image compression standards (as for example JPEG standards).

- There are fast algorithms to compute the DCT, similar to the FFT for computing the DFT.

6.2 WALSH TRANSFORM (WT)

6.2.1 One dimensional signals

This transform is slightly different from the transforms you have met so far. Suppose we have a function $f(x)$, $x = 0, , N-1$ where $N = 2^n$ and its Walsh transform $W(u)$.

If we use binary representation for the values of the independent variables x and u we need n bits to represent them. Hence, for the binary representation of x and u we can write:

$$(x)_{10} = \left(b_{n-1}(x)b_{n-2}(x)b_0(x)\right)_2 , (u)_{10} = \left(b_{n-1}(u)b_{n-2}(u)\, b_0(u)\right)_2$$

with $b_i(x)$ 0 or 1 for $i = 0, , n-1$.

Example

If $f(x), x = 0, ,7,$ (8 samples) then $n = 3$ and for $x = 6$,

$6 = (110)_2 \Rightarrow b_2(6) = 1, b_1(6) = 1, b_0(6) = 0$

We define now the 1-D Walsh transform as

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

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

The array formed by the Walsh kernels is again a symmetric matrix having orthogonal rows and columns. Therefore, the Walsh transform is and its elements are of the form $T(u, x) = \prod_{i=0}^{n-1}(-1)^{b_i(x)b_{n-1-i}(u)}$. You can immediately observe that $T(u, x) = 0$ or 1 depending on the

values of $b_i(x)$ and $b_{n-1-i}(u)$. If the Walsh transform is written in a matrix form

$$W = \underline{T} \cdot f$$

the rows of the matrix $\underline{T}$ which are the vectors $[T(u,0)\ T(u,1)\ T(u, N-1)]$ have the form of square waves. As the variable u (which represents the index of the transform) increases, the corresponding square wave's ―frequency‖ increases as well. For example for $u = 0$ we see that $(u)_{10} = (b_{n-1}(u)b_{n-2}(u)\ b_0(u))_2 = (00\ 0)_2$ and hence, $b_{n-1-i}(u) = 0$, for any i . Thus, $T(0, x) = 1$

and $W(0) = \frac{1}{N}\sum_{x=0}^{N-1} f(x)$. We see that the first element of the Walsh transform in the mean of the original function $f(x)$ (the DC value) as it is the case with the Fourier transform.

The inverse Walsh transform is defined as follows.

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

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

6.2.2 Two dimensional signals

The Walsh transform is defined as follows for two dimensional signals.

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

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

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

The inverse Walsh transform is defined as follows for two dimensional signals.

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

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

6.2.3 Properties of the Walsh Transform

- Unlike the Fourier transform, which is based on trigonometric terms, the Walsh transform consists of a series expansion of basis functions whose values are only −1 or 1 and they have the form of square waves. These functions can be implemented more efficiently in a digital environment than the exponential basis functions of the Fourier transform.

- The forward and inverse Walsh kernels are identical except for a constant multiplicative factor of $\frac{1}{N}$ for 1-D signals.

- The forward and inverse Walsh kernels are identical for 2-D signals. This is because the array formed by the kernels is a symmetric matrix having orthogonal rows and columns, so its inverse array is the same as the array itself.

- The concept of frequency exists also in Walsh transform basis functions. We can think of frequency as the number of zero crossings or the number of transitions in a basis vector and we call this number **sequency**. The Walsh transform exhibits the property of energy compaction as all the transforms that we are currently studying. (**why**?)

- For the fast computation of the Walsh transform there exists an algorithm called **Fast Walsh Transform (FWT).** This is a straightforward modification of the FFT. Advise any introductory book for your own interest.

6.3 HADAMARD TRANSFORM (HT)

6.3.1 Definition

In a similar form as the Walsh transform, the 2-D Hadamard transform is defined as follows.

Forward

$$H(u,v) = \frac{1}{N}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} f(x,y)\left[\prod_{i=0}^{n-1}(-1)^{(b_i(x)b_i(u)+b_i(y)b_i(v))}\right], \quad N = 2^n \text{ or}$$

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

Inverse

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

6.3.2 Properties of the Hadamard Transform

- **Most of the comments made for Walsh transform are valid here.**
- The Hadamard transform differs from the Walsh transform only in the order of basis functions. The order of basis functions of the Hadamard transform **does not** allow the fast computation of it by using a straightforward modification of the FFT. An extended version of the Hadamard transform is the **Ordered Hadamard Transform** for which a fast algorithm called **Fast Hadamard Transform (FHT)** can be applied.

- An important property of Hadamard transform is that, letting H_N represent the matrix of order N, the recursive relationship is given by the expression

$$H_{2N} = \begin{bmatrix} H_N & H_N \\ H_N & -H_N \end{bmatrix}$$

6.4 KARHUNEN-LOEVE (KLT) or HOTELLING TRANSFORM

The Karhunen-Loeve Transform or KLT was originally introduced as a series expansion for continuous random processes by Karhunen and Loeve. For discrete signals Hotelling first studied what was called a method of principal components, which is the discrete equivalent of the KL series expansion. Consequently, the KL transform is also called the Hotelling transform or the method of principal components. **The term KLT is the most widely used**.

6.4.1 The case of many realisations of a signal or image (Gonzalez/Woods)

The concepts of **eigenvalue** and **eigevector** are necessary to understand the KL transform.

If $\underline{C}$ is a matrix of dimension $n \times n$, then a scalar λ is called an eigenvalue of $\underline{C}$ if there is a nonzero vector $\underline{e}$ in R^n such that

$$\underline{C}\underline{e} = \lambda \underline{e}$$

The vector $\underline{e}$ is called an eigenvector of the matrix $\underline{C}$ corresponding to the eigenvalue λ.

(If you have difficulties with the above concepts consult any elementary linear algebra book.)

Consider a population of random vectors of the form

$$\underline{x} = \begin{bmatrix} x_1 \\ x^2 \\ \\ x_n \end{bmatrix}$$

The **mean vector** of the population is defined as

$$\underline{m}_x = E\{\underline{x}\}$$

The operator E refers to the expected value of the population calculated theoretically using the probability density functions (pdf) of the elements x_i and the joint probability density functions between the elements x_i and x_j.

The covariance matrix of the population is defined as

$$\underline{C}_x = E\left\{(\underline{x} - \underline{m}_x)(\underline{x} - \underline{m}_x)^T\right\}$$

Because x is n-dimensional, C_x and $(x - m_x)(x - m_x)^T$ are matrices of order $n \times n$. The element c_{ii} of C_x is the variance of x_i, and the element c_{ij} of C_x is the covariance between the elements x_i and x_j. If the elements x_i and x_j are uncorrelated, their covariance is zero and, therefore, $c_{ij} = c_{ji} = 0$.

For M vectors from a random population, where M is large enough, the mean vector and covariance matrix can be approximately calculated from the vectors by using the following relationships where all the expected values are approximated by summations

$$m_x = \frac{1}{M}\sum_{k=1}^{M} x_k$$

$$C_x = \frac{1}{M}\sum_{k=1}^{M} x_k x_k^T - m_x m_x^T$$

Very easily it can be seen that $\underline{C}_x$ is real and symmetric. In that case a set of n orthonormal (**at this point you are familiar with that term**) eigenvectors always exists. Let $\underline{e}_i$ and λ_i, $i = 1,2,\ldots,n$, be this set of eigenvectors and corresponding eigenvalues of $\underline{C}_x$, arranged in descending order so that $\lambda_i \geq \lambda_{i+1}$ for $i = 1,2,\ldots,n-1$. Let $\underline{A}$ be a matrix whose rows are formed from the eigenvectors of $\underline{C}_x$, ordered so that the first row of $\underline{A}$ is the eigenvector corresponding

to the largest eigenvalue, and the last row the eigenvector corresponding to the smallest eigenvalue.

Suppose that $\underline{A}$ is a transformation matrix that maps the vectors $\underline{x}'s$ into vectors $y's$ by using the following transformation

$$\underline{y} = \underline{A}(\underline{x} - \underline{m}_x)$$

The above transform is called the **Karhunen-Loeve** or **Hotelling** transform. The mean of the y vectors resulting from the above transformation is zero (**try to prove that**)

$$\underline{m}_y = \underline{0}$$

the covariance matrix is (**try to prove that**)

$$\underline{C}_y = \underline{A}\underline{C}_x\underline{A}^T$$

and $\underline{C}_y$ is a diagonal matrix whose elements along the main diagonal are the eigenvalues of $\underline{C}_x$ (**try to prove that**)

$$C_y = \begin{bmatrix} \lambda_1 & & 0 \\ & \lambda_2 & \\ & & \\ 0 & & \lambda_n \end{bmatrix}$$

The off-diagonal elements of the covariance matrix are 0, so the elements of the y vectors are uncorrelated.

Lets try to reconstruct any of the original vectors $\underline{x}$ from its corresponding y. Because the rows of $\underline{A}$ are orthonormal vectors (**why?**), then $\underline{A}^{-1} = \underline{A}^T$, and any vector $\underline{x}$ can by recovered from its corresponding vector y by using the relation

$$x = \underline{A}^T\underline{y} + \underline{m}_x$$

Suppose that instead of using all the eigenvectors of $\underline{C}_x$ we form matrix $\underline{A}_K$ from the K eigenvectors corresponding to the K largest eigenvalues, yielding a transformation matrix of

order $K \times n$. The y vectors would then be K dimensional, and the reconstruction of any of the

original vectors would be approximated by the following relationship

$$\hat{\underline{x}} = \underline{A}_K^T \underline{y} + \underline{m}_x$$

The mean square error between the perfect reconstruction $\underline{x}$ and the approximate reconstruction $\hat{\underline{x}}$ is given by the expression

$$e_{ms} = \sum_{j=1}^{n} \lambda_j - \sum_{j=1}^{K} \lambda_j = \sum_{j=K+1}^{n} \lambda_j .$$

By using $\underline{A}_K$ instead of $\underline{A}$ for the KL transform we achieve compression of the available data.

6.4.2 The case of <u>one</u> realisation of a signal or image

The derivation of the KLT for the case of one image realisation assumes that the two dimensional signal (image) is **ergodic**. This assumption allows us to calculate the statistics of the image using only one realisation. Usually we divide the image into blocks and we apply the KLT in each block. This is reasonable because the 2-D field is likely to be ergodic within a small block since the nature of the signal changes within the whole image. Let's suppose that f is a vector

obtained by lexicographic ordering of the pixels $f(x, y)$ within a block of size $M \times M$ (placing the rows of the block sequentially).

The mean vector of the random field inside the block is a scalar that is estimated by the approximate relationship

$$m_f = \frac{1}{M^2} \sum_{k=1}^{M^2} f(k)$$

and the covariance matrix of the 2-D random field inside the block is $\underline{C}_f$ where

$$c_{ii} = \frac{1}{M^2} \sum f(k) f(k) - m_f^2$$

and

$$c_{ij} = c_{i-j} = \frac{1}{M^2} \sum f(k) f(k+i-j) - m_f^2$$

After knowing how to calculate the matrix $\underline{C}_f$, the KLT for the case of a single realisation is the same as described above.

6.4.3 Properties of the Karhunen-Loeve transform

Despite its favourable theoretical properties, the KLT is not used in practice for the following reasons.

- Its basis functions depend on the covariance matrix of the image, and hence they have to recomputed and transmitted for every image.
- Perfect decorrelation is not possible, since images can rarely be modelled as realisations of ergodic fields.
- There are no fast computational algorithms for its implementation.

UNIT II

IMAGE ENHANCEMENT TECHNIQUES

Spatial Domain methods: Basic grey level transformation – Histogram equalization –

Image subtraction – Image averaging –Spatial filtering: Smoothing, sharpening filters –

Laplacian filters – Frequency domain filters: Smoothing – Sharpening filters –

1. SPATIAL DOMAIN METHODS

* Suppose we have a digital image which can be represented by a two dimensional random field $f(x, y)$.

* An image processing operator in the spatial domain may be expressed as a $[]\cdot$ applied to the image $f(x, y)$ to produce a new imagemathematicalfunction$T$ $g(x, y) = T[f(x, y)]$ as follows.

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

The operator T applied on $f(x, y)$ may be defined over:

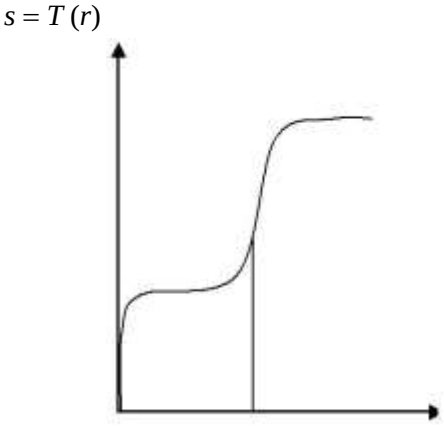
(i) A single pixel (x, y). In this case T is a grey level transformation (or mapping) function.

(ii) Some neighbourhood of (x, y).

(iii) T may operate to a set of input images instead of a single image.

Example 1

The result of the transformation shown in the figure below is 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. This technique is known as **contrast stretching**.

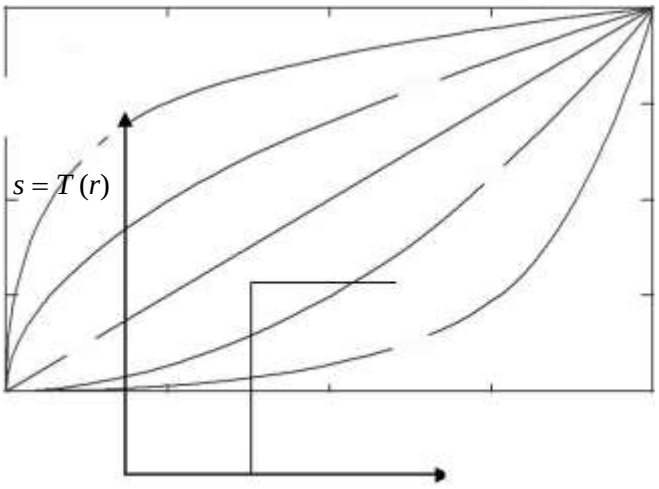


r

m

Example 2

The result of the transformation shown in the figure below is to produce a binary image.



2. BASIC GRAY LEVEL TRANSFORMATIONS

* We begin the study of image enhancement techniques b y discussing gray-level transformation functions. These are among the simplest of all image enhancement techniques.

* The values of pixels, before and after processing, will be denoted by *r* and *s*, respectively. As indicated in the previous section, these values are related by an

expression of the form s=T(r), where T is a transformation that maps a pixel value *r* into a pixel value *s*.

* Since we are dealing with digital quantities, values of the transformation function typically are stored in a one-dimensional a r r a y and the mappings from *r* to *s* are implemented v i a table lookups. For an 8-bit environment, a lookup table containing the values of *T* will have 256 entries.

* As an introduction to gray-level transformations, consider Fig. 3.3, which shows three basic types of functions used frequently for image enhancement:

linear (negative and identity transformations), logarithmic (log and inverse-log transformations), and power-law (*n*th power and *n*th root transformations).

* The identity function is the trivial case in which output int e ns it ie s

ar e identical to input intensities. It is included in the graph only for completeness.

2.1 Image Negatives

The negative of an image with gray levels in the range [0, L-1] is obtained by using the negative transformation shown in Fig. 3.3, which is given by the expression

Reversing the intensity levels of an image in this manner produces the equivalent of a photographic negative. This type of processing is particularly s u i t e d for enhancing whit e or gray detail embedded i n dark regions of an image, especially when the black areas are dominant i n size. An example is shown in Fig. 3.4. The original image is a digital mammogram s h o w i n g a small lesion. In spite of the fact that the visual content is the same in both images, note how much easier it is to analyze the breast tissue in the negative image in this particular case.

2.2 Log Transformations

The general form of the log transformation shown in Fig. 3.3 is

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

where *c* is a constant, and it is assumed that r 0.

- The shape of the log curve in Fig. 3.3 shows that this transformation maps a narrow range of low gray-level values in the input image into a wider range of output levels.
- The opposite is true of higher values of input levels. We would use a transformation of this type to expand the values of dark pixels in an image while compressing the higher-level values. The opposite is true of the inverse log transformation.

* Any curve having the general shape of the log functions shown in

Fig. 3.3 would accomplish this spreading/compressing of gray levels in an image. In fact, the power-law t ransfor mat ions discussed in the next section are much more versatile for this purpose t han the log transformation. However, the log function has

the important char a ct er ist ic t hat it compresses the dynamic range of images with

large variations in pixel values.

* A classic illustration o f an application in which pixel values have a large dynamic range is the Fourier spectrum; we are concerned o nly with the image characteristics o f spectra. It is not unusual to encounter s p e c t r u m values that range from 0 to 10^6 or higher. While processing numbers such as these presents no problems

for a computer, image display systems generally will not be able to reproduce

f a i t h f u l l y such a wide range of intensity values. The net effect is that a significant

degree of detail will be lost in the display of a typical Fourier spectrum.

As an illustration of log transformations, F i g . 3.5(a) shows a Fourier spectrum with

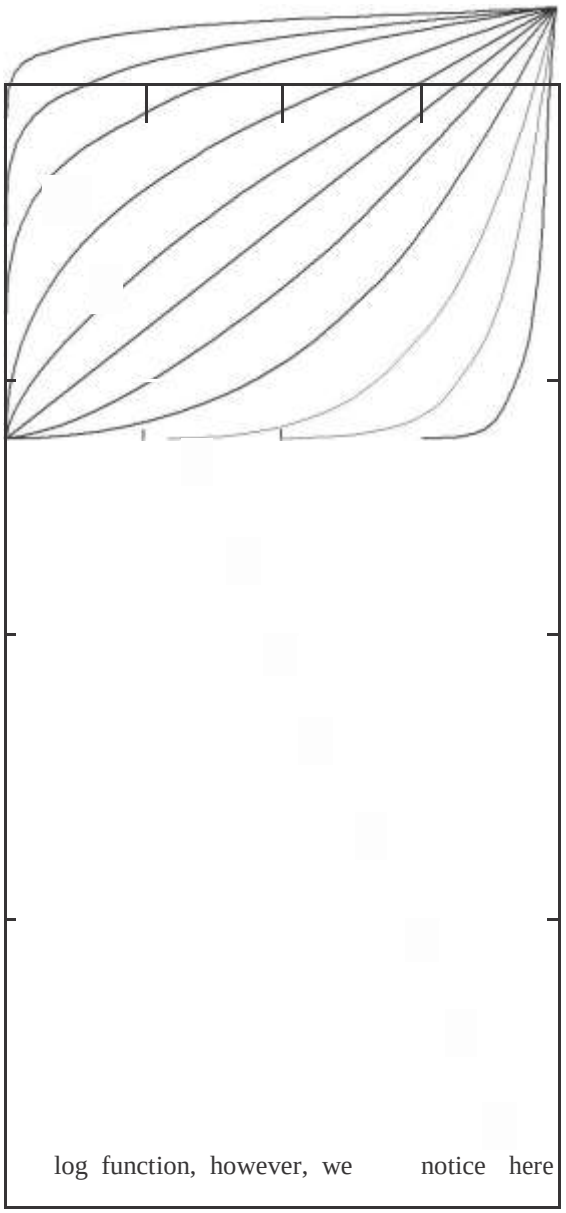
values in the range 0 to $1.5*10^6$. When these values are scaled linearly for display in an 8-bit system, the brightest pixels will dominate the display, at the expense of lower

(and just as important) v a l u e s of the spectrum. The effect of this dominance i s illustrated v i v i d l y by the relatively small area of the image in Fig. 3.5(a) that is not perceived as black. If, instead of displaying the values in this manner, we first apply Eq. (3.2-2) (with c=1 in this case) to the spectrum values, then the range of values of the result become 0 to 6.2, a more manageable number. Figure 3.5(b) shows the result of scaling this new range linearly and displaying the spectrum in the same 8-bit display. The wealth of detail visible in this image as compared to a straight display of the spectrum is evident from these pictures. Most of the Fourier spect r a seen in image processing publications h a v e been scaled in just this manner.

2.3 Power-Law Transformations

Power-law transformations have the basic form $s = cr^{g}$

where *c* and g are positive constants. Sometimes Eq. (3.2-3) is written as $s = c(r + e)^{g}$ to account for an offset (that is, a measurable output when the input is zero). However, offsets typically are an issue of display calibration and as a result they are normally ignored in Eq. (3.2-3). Plots of *s* versus *r* for various values of g are shown in Fig. 3.6. As in the case of the log transformation, power-law curves with fractional values of g 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 g. As expected, we see in Fig. 3.6

that curves generated w i t h values of g>1 have exactly the opposite e ffect as those generated w i t h values of g<1. Finally, we note that Eq. (3.2-3) reduces to the identity transformation when c= g =1 .

* A variety of devices used for image capture, printing, and display respond according to a power law. By convention, the exponent in the power-law equation is referred to as *gamma* [hence our use of this symbol in Eq. (3.2-3)]. The process used to correct t his power-law response p he no me no n is called *gamma correction*. For example, cathode r a y tube (CRT) d e vic e s have intensity-to-volt- age response that is a p o w e r function, with exponents varying from approximately 1 . 8 to 2.5.

With reference t o the curve for g=2.5 in Fig. 3.6, we see that such display systems would tend to produce i m a g e s that are darker than intended. This effect is illustrated in Fig. 3.7. Figure 3.7(a) shows a simple gray-scale linear wedge input into a CRT monitor. As expected, the output of the monitor appears darker than the input, as shown in Fig. 3.7(b).

* Gamma correction in this case is straightforward. All we need to do is preprocess t he input image before inputt ing it into the monitor by performing t h e transformations $= r^{1\ 2.5} = r^{0.4}$. The result is shown in Fig. 3.7(c). When input into the same monitor, this gamma-corrected input produces a n output t ha t is close in appearance t o the original image, as shown in Fig. 3.7(d). A similar analysis would apply to other imaging devices such as scanners and printers. The only difference would be the device-dependent value of gamma (Poynton [1996]).

* Gamma correction is important i f displaying an image accurately on a computer screen is of concern. Images that are not corrected pro per ly can look either bleached out, or, what is more likely, too dark. Trying to reproduce c o l o r s accurately also requires some knowledge of gamma correction because varying the value of gamma correction changes not only the brightness, but also the ratios of red to green to blue. Gamma co r r e ct io n h a s become increasingly important in the past few years, as use of digital images for commercial purposes over the Internet h a s increased. It is not unusual that images created for a popular Web site will be

viewed by millions of people, the majority of whom will have different monitors and/or monitor settings.

2. Spatial domain: Enhancement by point processing

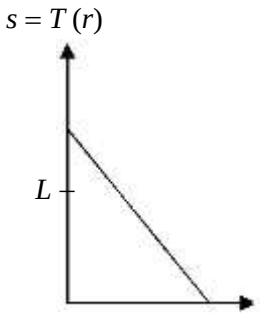
We are dealing now with image processing methods that are based only on the intensity of single pixels.

2.1 Intensity transformations

2.1.1 Image Negatives

The negative of a digital image is obtained by the transformation function $s = T(r) = L - 1 - r$ shown in the following figure, where L is the number of grey levels.

The idea is that the intensity of the output image decreases as the intensity of the input increases. This is useful in numerous applications such as displaying medical images.



2.1.2 Contrast Stretching

Low contrast images occur often due to poor or non uniform lighting conditions, or due to nonlinearity, or small dynamic range of the imaging sensor. In the figure of Example 1 above you have seen a typical contrast stretching transformation.

2.2 Histogram processing. Definition of the histogram of an image.

By processing (modifying) the histogram of an image we can create a new image with specific desired properties.

Suppose we have a digital image of size $N \times N$ with grey levels in the range $[0, L-1]$. The histogram of the image is defined as the following discrete function:

$$p(r_k) = \frac{n_k}{N^2}$$

where

r_k is the kth grey level, $k = 0,1, \ldots, L-1$

n_k is the number of pixels in the image with grey level r_k

N^2 is the total number of pixels in the image

The histogram represents the frequency of occurrence of the various grey levels in the image. A plot of this function for all values of k provides a global description of the appearance of the image.

2.3 Global histogram equalization

In this section we will assume that the image to be processed has a continuous intensity that lies within the interval $[0, L-1]$. Suppose we divide the image intensity with its maximum value $L-1$. Let the variable r represent the new grey levels (image intensity) in the image, where now $0 \le r \le 1$ and let $p_r(r)$ denote the probability density function (pdf) of the variable r . We now apply the following transformation function to the intensity

$$s = T(r) = \int_0^r p_r(w)dw , \qquad 0 \le r \le 1$$

(1) By observing the transformation of equation (1) we immediately see that it possesses the following properties:

(i) $0 \le s \le 1$.

(ii) $r_2 > r_1 \Rightarrow T(r_2) \ge T(r_1)$, i.e., the function $T(r)$ is increase ng with r .

(iii) $s = T(0) = \int_0^0 p_r(w)dw = 0$ and $s = T(1) = \int_0^1 p_r(w)dw = 1$. Moreover, if the original image has intensities **only** within a certain range $[r_{\min}, r_{\max}]$ then $s = T(r_{\min}) = \int_0^{r_{\min}} p_r(w)dw = 0$ and $s = T(r_{\max}) = \int_0^{r_{\max}} p_r(w)dw = 1$ since $p_r(r) = 0$, $r < r_{\min}$ and $r > r_{\max}$. Therefore, the new intensity s takes always all values within the available range [0 1].

Suppose that $P_r(r)$, $P_s(s)$ are the probability distribution functions (PDF's) of the variables r and s respectively.

Let us assume that the original intensity lies within the values r and $r + dr$ with dr a small quantity. dr can be assumed small enough so as to be able to consider the function $p_r(w)$ constant within the interval $[r, r + dr]$ and equal to $p_r(r)$. Therefore,

$$P_r[r, r + dr] = \int_r^{r+dr} p_r(w)dw \cong p_r(r) \int_r^{r+dr} dw = p_r(r)dr .$$

Now suppose that $s = T(r)$ and $s_1 = T(r + dr)$. The quantity dr can be assumed small enough so as to be able to consider that $s_1 = s + ds$ with ds small enough so as to be able to consider the function $p_s(w)$ constant within the interval $[s, s + ds]$ and equal to $p_s(s)$. Therefore,

$$P_s[s, s + ds] = \int_s^{s+ds} p_s(w)dw \cong p_s(s) \int_s^{s+ds} dw = p_s(s)ds$$

Since $s = T(r)$, $s + ds = T(r + dr)$ and the function of equation (1) is increasing with r , all and only the values within the interval $[r, r + dr]$ will be mapped within the interval $[s, s + ds]$. Therefore,

$$P_r[r, r + dr] = P_s[s, s + ds] \Rightarrow p_r(r)dr \overset{r = T^{-1}(s)}{=} p_s(s)ds \Rightarrow p_s(s) = p_r(r) \left| \frac{dr}{ds} \right|_{r = T^{-1}(s)}$$

From equation (1) we see that

$$\frac{ds}{dr} = p_r(r)$$

and hence,

$$p_s(s) = \left[p_r(r) \left| \frac{1}{p_r(r)} \right| \right]_{r = T^{-1}(s)} = 1, \quad 0 \le s \le 1$$

Summary

From the above analysis it is obvious that the transformation of equation

(1) Converts the original image into a new image with uniform probability density function. This means that in the new image all intensities are present [look at property (iii) above] and with equal probabilities. The whole ranges of intensities from the absolute black to the absolute white are explored and the new image will definitely have higher contrast compared to the original image.

Unfortunately, in a real life scenario we must deal with digital images. The discrete form of histogram equalisation is given by the relation

$$k \quad n_j \quad k$$

$$s_k = T(r_k) = \sum_{j=0}^{k} \frac{n_j}{N} = \sum_{j=0}^{k} p_r(r_j),\ 0 \le r_k \le 1,\ k = 0,1,\ \ldots,\ L-1 \qquad (2)$$

The quantities in equation (2) have been defined in Section 2.2. To see results of histogram equalisation look at any introductory book on Image Processing.

The improvement over the original image is quite evident after using the technique of histogram equalisation. The new histogram is **not flat** because of the discrete approximation of the probability density function with the histogram function. Note, however, that the grey levels of an image that has been subjected to histogram equalisation are spread out and always reach white. This process increases the dynamic range of grey levels and produces an increase in image contrast.

2.4 Local histogram equalisation

* Global histogram equalisation is suitable for overall enhancement. It is often necessary to enhance details over small areas. The number of pixels in these areas my have negligible influence on the computation of a global transformation, so the use of this type of transformation does not necessarily guarantee the desired local enhancement.

* The solution is to devise transformation functions based on the grey level distribution – or other properties – in the neighbourhood of every pixel in the image. The histogram processing technique previously described is easily adaptable to local enhancement.

* The procedure is to define a square or rectangular neighbourhood and move the centre of this area from pixel to pixel. At each location the histogram of the points in the neighbourhood is computed and a histogram equalisation transformation function is obtained.

* This function is finally used to map the grey level of the pixel centred in the neighbourhood. The centre of the neighbourhood region is then moved to an adjacent pixel location and the procedure is repeated. Since only one new row or column of the neighbourhood changes during a pixel-to-pixel translation of the region, updating the histogram obtained in the previous location with the new data introduced at each motion step is possible quite easily.

* This approach has obvious advantages over repeatedly computing the histogram over all pixels in the neighbourhood region each time the region is moved one pixel location. Another approach often used to reduce computation is to utilise non overlapping regions, but this methods usually produces an undesirable checkerboard effect.

2.5 Histogram specification

- Suppose we want to specify a particular histogram shape (not necessarily uniform) which is capable of highlighting certain grey levels in the image.

Let us suppose that:

$p_r(r)$ is the original probability density function

$p_z(z)$ is the desired probability density function

- Suppose that histogram equalisation is first applied on the original image r

$$s = T(r) = \int_0^r p_r(w)dw$$

- Suppose that the desired image z is available and histogram equalisation is applied as well

$$v = G(z) = \int_0^z p_z(w)dw$$

$p_s(s)$ and $p_v(v)$ are both uniform densities and they can be considered as identical. Note that the final result of histogram equalisation is independent of the density inside the integral. So in equation $v = G(z) = \int_0^z p_z(w)dw$ we can use the symbol s instead of v.

The inverse process $z = G^{-1}(s)$ will have the desired probability density function. Therefore, the process of histogram specification can be summarised in the following steps.

(i) We take the original image and equalise its intensity using the relation $s = T(r) = \int_0^r p_r(w)dw$.

(ii) From the given probability density function $p_z(z)$ we specify the probability distribution function $G(z)$.

(iii) We apply the inverse transformation function $z = G^{-1}(s) = G^{-1}[T(r)]$

3. Spatial domain: Enhancement in the case of many realizations of an image of interest available

3.1 Image averaging

- Suppose that we have an image $f(x, y)$ of size $M \times N$ pixels corrupted by noise $n(x, y)$, so we obtain a noisy image as follows.

$$g(x, y) = f(x, y) + n(x, y)$$

For the noise process $n(x, y)$ the following assumptions are made.

(i) The noise process $n(x, y)$ is ergodic.

(ii) It is zero mean, i.e., $E\{n(x, y)\} = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} n(x, y) = 0$

(ii) It is white, i.e., the autocorrelation function of the noise process defined as

$R[k,l] = E\{n(x, y)n(x + k, y + l)\} = \frac{1}{(M-k)(N-l)} \sum_{x=0}^{M-1-k} \sum_{y=0}^{N-1-l} n(x, y)n(x + k, y + l)$ is zero,

apart for the pair $[k,l] = [0,0]$.

Therefore, $R[k,l] = \frac{1}{(M-k)(N-l)} \sum_{x=0}^{M-1-k} \sum_{y=0}^{N-1-l} n(x, y)n(x + k, y + l) = \sigma_n^{2}{}_{(x, y)} \delta(k,l)$ where

$\sigma_n^{2}{}_{(x, y)}$ is the variance of noise.

- Suppose now that we have L different noisy realisations of the same image $f(x, y)$ as $g_i(x, y) = f(x, y) + n_i(x, y)$, $i = 0,1, \ldots, L$. Each noise process $n_i(x, y)$

satisfies the properties (i)-(iii) given above. Moreover, $\sigma^2_{n_i(x,y)} = \sigma^2$. We form the image $\bar{g}(x,y)$ by averaging these L noisy images as follows:

$$\bar{g}(x,y) = \frac{1}{L}\sum_{i=1}^{L} g_i(x,y) = \frac{1}{L}\sum_{i=1}^{L}(f(x,y) + n_i(x,y)) = f(x,y) + \frac{1}{L}\sum_{i=1}^{L} n_i(x,y)$$

Therefore, the new image is again a noisy realisation of the original image $f(x,y)$ with noise $n(x,y) = \frac{1}{L}\sum_{i=1}^{L} n_i(x,y)$.

The mean value of the noise $n(x,y)$ is found below.

$$E\{n(x,y)\} = E\{\frac{1}{L}\sum_{i=1}^{L} n_i(x,y)\} = \frac{1}{L}\sum_{i=1}^{L} E\{n_i(x,y)\} = 0$$

The variance of the noise $n(x,y)$ is now found below.

$$\sigma^2_{n(x,y)} = E\{n^2(x,y)\} = E\left\{\left[\frac{1}{L}\sum_{i=1}^{L} n_i(x,y)\right]^2\right\} = \frac{1}{L^2}E\left\{\left[\sum_{i=1}^{L} n_i(x,y)\right]^2\right\}$$

$$= \frac{1}{L^2}E\{(\sum_{i=1}^{L} n_i(x,y))^2\} = \frac{1}{L^2}E\{(\sum_{i=1}^{L}\sum_{\substack{j=1 \\ i\neq j}}^{L}(n_i(x,y)n_j(x,y))\} = \frac{1}{L^2}\sum_{i=1}^{L} E\{n_i^2(x,y)\} + \frac{1}{L^2}\sum_{i=1}^{L}\sum_{\substack{j=1 \\ i\neq j}}^{L} E\{n_i(x,y)n_j(x,y)\}$$

$$= \frac{1}{L^2}\sum_{i=1}^{L}\sigma^2 + 0 = \frac{1}{L}\sigma^2$$

Therefore, we have shown that image averaging produces an image $\bar{g}(x,y)$, corrupted by

noise with variance less than the variance of the noise of the original noisy images. Note that if $L \to \infty$ we have $\sigma_n^2{}_{(x,y)} \to 0$, the resulting noise is negligible.

4. Spatial domain: Enhancement in the case of a single image

4.1Spatial masks

Many image enhancement techniques are based on spatial operations performed on local neighborhoods of input pixels.

The image is usually convolved with a finite impulse response filter called spatial mask.

The use of spatial masks on a digital image is called spatial filtering.

Suppose that we have an image $f(x, y)$ of size N^2 and we define a neighbourhood around each pixel. For example let this neighbourhood to be a rectangular window of size

3×3

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

If we replace each pixel by a weighted average of its neighbourhood pixels then the response of the linear mask for the pixel z_5 is $\sum_{i=1}^{9} w_i z_i$. We may repeat the same process for the whole image.

4.2 Low pass and high pass spatial filtering

A 3×3 spatial mask operating on an image can produce (a) a smoothed version of the image (which contains the low frequencies) or (b) it can enhance the edges and suppress essentially the constant background information. The behaviour is basically dictated by the signs of the elements of the mask.

Let us suppose that the mask has the following form

a	b	c
d	1	e
f	g	h

To be able to estimate the coefficients a,b,c,d,e,f,g,h effects of the above mask with relation to the sign of the , we will consider the equivalent one dimensional mask

d 1 e

Let us suppose that the above mask is applied to a signal $x(n)$. The output of this operation will be a signal $y(n)$ as

$y(n) = dx(n-1) + x(n) + ex(n+1) \Rightarrow Y(z) = dz^{-1}X(z) + X(z) + ezX(z) \Rightarrow$

$Y(z) = (dz^{-1} + 1 + ez)X(z) \Rightarrow \dfrac{Y(z)}{X(z)} = H(z) = dz^{-1} + 1 + ez$. This is the transfer function of a system that produces the above input-output relationship. In the frequency domain we have $H(e^{j\omega}) = d\exp(-j\omega) + 1 + e\exp(j\omega)$.

The values of this transfer function at frequencies $\omega = 0$ and $\omega = \pi$ are:

$$H(e^{j\omega})\Big|_{\omega=0} = d + 1 + e$$

$$H(e^{j\omega})\Big|_{\omega=\pi} = -d + 1 - e$$

If a lowpass filtering (smoothing) effect is required then the following condition must hold

$$H(e^{j\omega})\Big|_{\omega=0} \geq H(e^{j\omega})\Big|_{\omega=\pi} = d + e \geq 0$$

If a highpass filtering effect is required then

$$H(e^{j\omega})\Big|_{\omega=0} \leq H(e^{j\omega})\Big|_{\omega=\pi} = d + e \leq 0$$

The most popular masks for lowpass filtering are masks with all their coefficients positive and for highpass filtering, masks where the central pixel is positive and the surrounding pixels are negative or the other way round.

4.3 Popular techniques for low pass spatial filtering

4.3.1 Uniform filtering

The most popular masks for low pass filtering are masks with all their coefficients positive and equal to each other as for example the mask shown below. Moreover, they sum up to 1 in order to maintain the mean of the image.

$$\frac{1}{9} \times \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

4.3.2 Gaussian filtering

The two dimensional Gaussian mask has values that attempts to approximate the continuous function

$$\frac{1}{} \quad {}^{-x^2+y^2}$$

$$G(x, y) = \frac{}{2\pi\sigma^2} e^{\sigma^2}$$

In theory, the Gaussian distribution is non-zero everywhere, which would require an infinitely large convolution kernel, but in practice it is effectively zero more than about three standard deviations from the mean, and so we can truncate the kernel at this point. The following shows a suitable integer-valued convolution kernel that approximates a Gaussian with a σ of 1.0.

4.3.3 Median filtering

The median *m* of a set of values is the value that possesses the property that half the values in the set are less than *m* and half are greater than *m* . Median filtering is the operation that replaces each pixel by the median of the grey level in the neighbourhood of that pixel.

Median filters are non linear filters because for two sequences $x(n)$ and $y(n)$

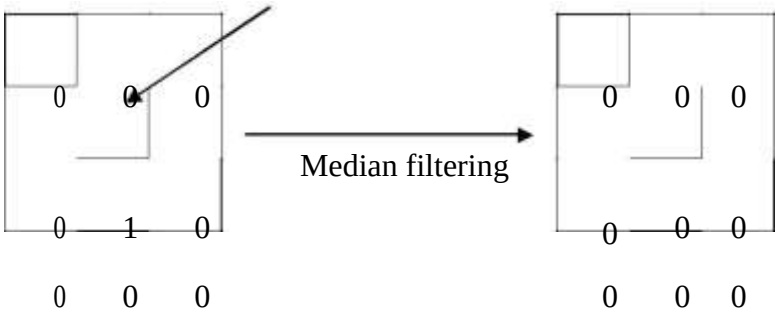
$$\text{median}\{x(n) + y(n)\} \neq \text{median}\{x(n)\} + \text{median}\{y(n)\}$$

Median filters are useful for removing isolated lines or points (pixels) while preserving spatial resolutions. They perform very well on images containing binary (**salt and pepper**) noise but perform poorly when the noise is Gaussian. Their performance is also poor when the number of noise pixels in the window is greater than or half the number of pixels in the window (**why?**)

$\frac{1}{273} \times$

1	4	7	4	1
4	16	26	16	4
7	26	41	26	7
4	16	26	16	4
1	4	7	4	1

Isolated



4.3.4 Directional smoothing

To protect the edges from blurring while smoothing, a directional averaging filter can be useful. Spatial averages $g(x, y:\theta)$ are calculated in several selected directions (for example could be horizontal, vertical, main diagonals)

$$g(x, y:\theta) = \frac{1}{N_\theta} \sum_{(k,l) \in W_\theta} \sum f(x-k, y-l)$$

and a direction θ^* is found such that $\left| f(x, y) - g(x, y:\theta^*) \right|$ is minimum. (Note that W_θ is the neighbourhood along the direction θ and N_θ is the number of pixels within this neighbourhood).

Then by replacing $g(x, y:\theta)$ with $g(x, y:\theta^*)$ we get the desired result.

4.3.5 High Boost Filtering

A high pass filtered image may be computed as the difference between the original image and a lowpass filtered version of that image as follows:

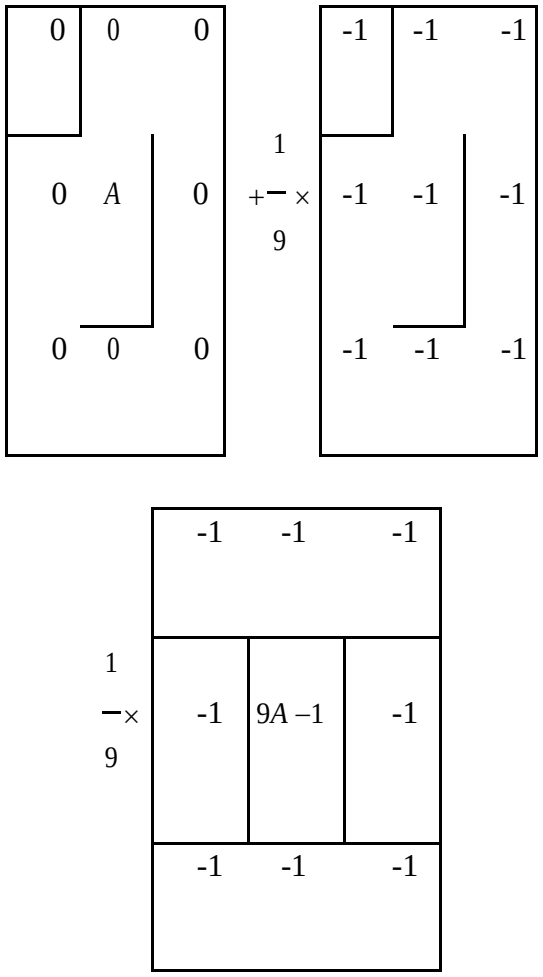
(Highpass part of image)=(Original)-(Lowpass part of image)

Multiplying the original image by an amplification factor denoted by A, yields the so called high boost filter:

(Highboost image)= (A) (Original)-(Lowpass)= $(A-1)$ (Original)+(Original)-(Lowpass)

= $(A-1)$ (Original)+(Highpass)

The general process of subtracting a blurred image from an original as given in the first line is called ***unsharp masking***. A possible mask that implements the above procedure could be the one illustrated below.



The high-boost filtered image looks more like the original with a degree of edge enhancement, depending on the value of A .

4.4 Popular techniques for high pass spatial filtering. Edge detection using derivative filters

4.4.1 About two dimensional high pass spatial filters

An edge is the boundary between two regions with relatively distinct grey level properties. The idea underlying most edge detection techniques is the computation of a local derivative operator. The magnitude of the first derivative calculated within a neighbourhood around the pixel of interest, can be used to detect the presence of an edge in an image.

The gradient of an image $f(x, y)$ at location (x, y) is a vector that consists of the partial derivatives of $f(x, y)$ as follows.

$$\begin{bmatrix} \partial f(x, y) \\ \hline \end{bmatrix}$$

$$\nabla f(x,y) = \begin{bmatrix} \dfrac{\partial}{\partial x} \\ \dfrac{\partial f(x,y)}{\partial y} \end{bmatrix}$$

The magnitude of this vector, generally referred to simply as the gradient ∇f is

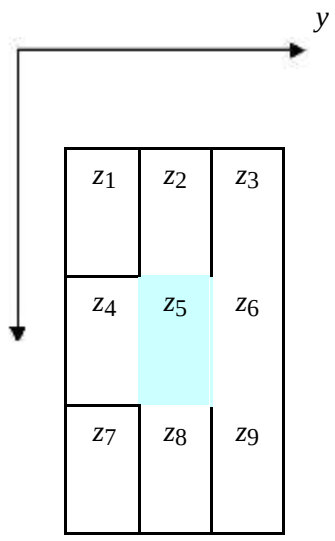
$$\nabla f(x,y) = \text{mag}(\nabla f(x,y)) = \left[\left(\frac{\partial f(x,y)}{\partial x} \right)^2 + \left(\frac{\partial f(x,y)}{\partial y} \right)^2 \right]^{1/2}$$

Common practice is to approximate the gradient with absolute values which is simpler to implement as follows.

$$\nabla f(x,y) \cong \left| \frac{\partial f(x,y)}{\partial x} \right| + \left| \frac{\partial f(x,y)}{\partial y} \right|$$

(1) Consider a pixel of interest $f(x,y) = z_5$ and a rectangular neighbourhood of size

$3 \times 3 = 9$ pixels (including the pixel of interest) as shown below.



x

4.4.2 Roberts operator

Equation (1) can be approximated at point z_5 in a number of ways. The simplest is to use the difference $(z_5 - z_8)$ in the *x* direction and $(z_5 - z_6)$ in the *y* direction. This

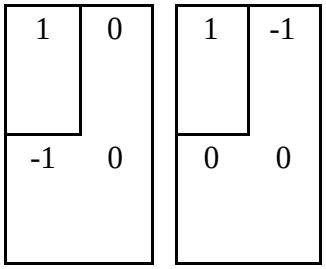
approximation is known as the **Roberts** operator, and is expressed mathematically as follows.

$$\nabla f \cong |z_5 - z_8| + |z_5 - z_6|$$

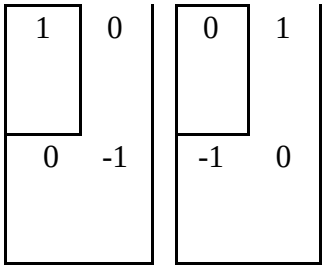
(2) Another approach for approximating (1) is to use cross differences

$$\nabla f \cong |z_5 - z_9| + |z_6 - z_8|$$

(3) Equations (2), (3) can be implemented by using the following masks. The original image is convolved with both masks separately and the absolute values of the two outputs of the convolutions are added.



Roberts operator



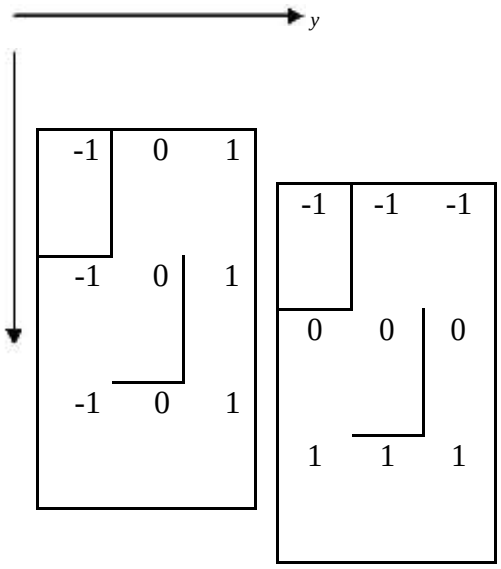
Roberts operator

4.4.3 Prewitt operator

Another approximation of equation (1) but using now a 3×3 mask is the following.

$$\nabla f \cong (z_7 + z_8 + z_9) - (z_1 + z_2 + z_3) + (z_3 + z_6 + z_9) - (z_1 + z_4 + z_7) \quad (4)$$

This approximation is known as the **Prewitt** operator. Equation (4) can be implemented by using the following masks. Again, the original image is convolved with both masks separately and the absolute values of the two outputs of the convolutions are added.



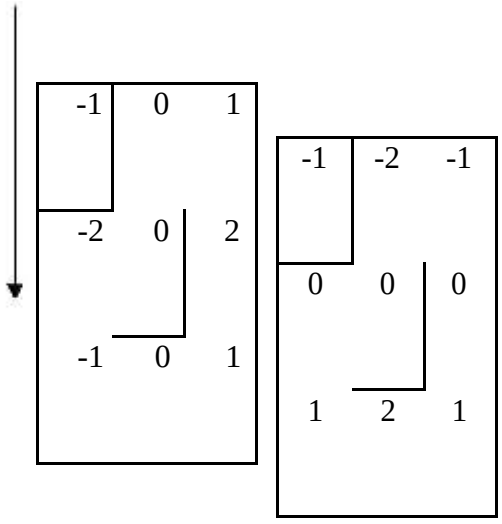
Prewitt operator

4.4.4 Sobel operator. Definition and comparison with the Prewitt operator

The most popular approximation of equation (1) but using a 3×3 mask is the following.

$$\nabla f \cong (z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3) + (z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7) \quad (5)$$

This approximation is known as the **Sobel** operator.

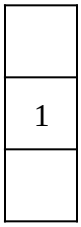

-1 0 1
-2 0 2
-1 0 1
-1 -2 -1
0 0 0
1 2 1

x

If we consider the left mask of the Sobel operator, this causes differentiation along the *y*

direction. A question that arises is the following: What is the effect caused by the same mask along the *x* direction?

If we isolate the following part of the mask

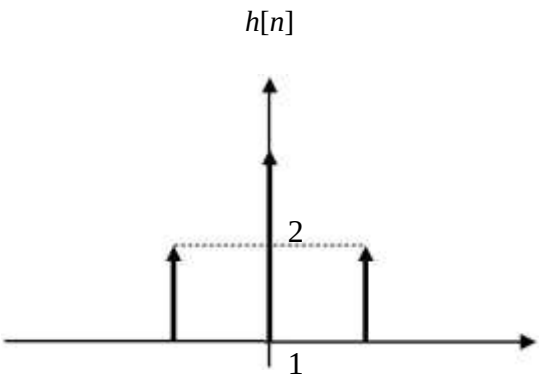


1

and treat it as a one dimensional mask, we are interested in finding the effects of that mask. We will therefore, treat this mask as a one dimensional impulse response $h[n]$ of the form

$$h[n] = \begin{cases} 1 & n = -1 \\ 2 & n = 0 \\ 1 & n = 1 \\ \text{ot herwise} \end{cases}$$

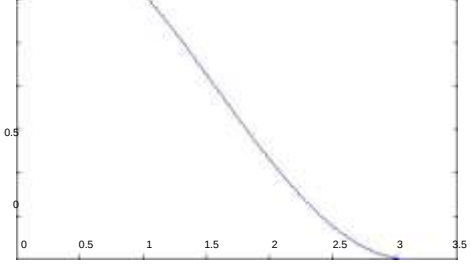
Or



The above response applied to a signal $x[$ signal $y[n] = x[n$ $+ 2x[n] + x[n+1]$

or in z-transform domain $Y(z) = (z^{-1} + 2 + z)\, X(z) \Rightarrow Y(j\omega) = 2(\cos\omega + 1)\, X(j\omega)$. Therefore, $h[n]$is the impulse response of a system with transfer function $H(j\omega) = 2(\cos\omega + 1) = |H(j\omega)|$ shown in the figure below for $[0, \pi]$. This is a lowpass filter type of response. Therefore, we can claim that the Sobel operator has a differentiation effect along one of the two directions and a smoothing effect along the other direction.

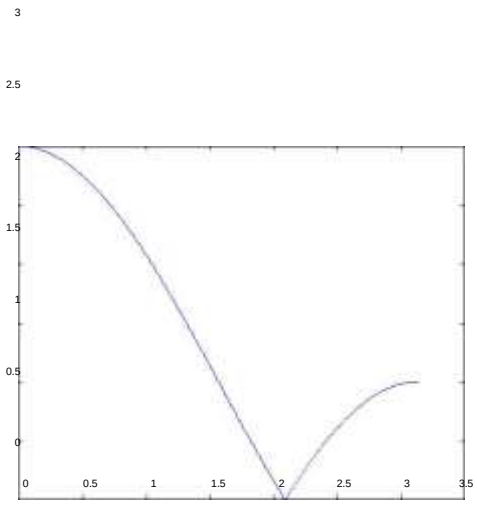
4
3.5
3
2.5
2
1.5
1



The same analysis for the Prewitt operator would give $Y(z) = (z^{-1} + 1 + z)\, X(z)$

$\Rightarrow Y(j\omega) = (2\cos\omega + 1)\, X(j\omega) \Rightarrow |H(j\omega)| = |2\cos\omega + 1|$ shown in the figure below for $[0, \pi]$.

This response looks ―strange‖ since it decreases up to the point $2\cos\omega + 1 = 0 \Rightarrow \cos\omega = -0.5$ and then starts increasing.



Based on the above analysis it is stated in the literature that the Sobel operator have the advantage of providing both a differencing a smoothing effect while Prewitt does not. However, if you implement both operators you cannot see any visual difference.

4.4.5 Laplacian operator

The **Laplacian** of a 2-D function $f(x, y)$ is a second order derivative defined as

$$\nabla^2 f(x,y) = \frac{\partial^2 f(x,y)}{\partial x^2} + \frac{\partial^2 f(x,y)}{\partial y^2}$$

In practice it can be also implemented using a 3x3 mask as follows (**why?**)

$$\nabla^2 f = 4z_5 - (z_2 + z_4 + z_6 + z_8)$$

The main disadvantage of the Laplacian operator is that it produces double edges (**why?**).

1.2 Frequency domain methods

(*Remaining topics clarification please refer Gonzalez-digital image processing*)

Let $g(x, y)$ be a desired image formed by the convolution of an image $f(x, y)$

and a linear, position invariant operator $h(x, y)$, that is:

$$g(x, y) = h(x, y) * f(x, y)$$

The following frequency relationship holds:

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

We can select $H(u, v)$ so that the desired image

$$g(x, y) = \Im^{-1}\{H(u, v)F(u, v)\}$$

exhibits some highlighted features of $f(x, y)$. For instance, edges in $f(x, y)$ can be accentuated by using a function $H(u, v)$ that emphasises the high frequency components of $F(u, v)$.

UNIT-III

IMAGE RESTORATION

Model of Image Degradation/restoration process – Noise models – Inverse filtering - Least mean square filtering – Constrained least mean square filtering – Blind image restoration – Pseudo inverse – Singular value decomposition.

1.1 What is image restoration?

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
- during acquisition mode, or
- during processing mode

The degradations may be due to

- sensor noise
- blur due to camera misfocus
- relative object-camera motion
- random atmospheric turbulence
- others

In most of the existing image restoration methods we assume that the degradation process can be described using a mathematical model.

1.2 How well can we do?

Depends on how much we know about

- the original image
- the degradations
  (how accurate our models are)

1.3 Image restoration and image enhancement differences

- Image restoration differs from image enhancement in that the latter is concerned more with accentuation or extraction of image features rather than restoration of degradations.

- Image restoration problems can be quantified precisely, whereas enhancement criteria are difficult to represent mathematically.

1.4 Image observation models

Typical parts of an imaging system: image formation system, a detector and a recorder. A general model for such a system could be:

$$y(i, j) = r\left[w(i, j)\right] + n(i, j)$$

$$w(i, j) = H\left[f(i, j)\right] = \int\int_{-\infty}^{\infty} h(i, j, i', j')\, f(i', j')\, di'\, dj'$$

$$n(i, j) = g\left\{r[w(i, j)]\right\} n_1(i, j) + n_2(i, j)$$

where $y(i, j)$ is the degraded image, $f(i, j)$ is the original image and $h(i, j, i', j')$ is an operator that represents the degradation process, for example a blurring process. Functions $g(\cdot)$ and $r(\cdot)$

are generally nonlinear, and represent the characteristics of detector/recording mechanisms. $n(i, j)$ is the additive noise, which has an image-dependent random component

$g\left\{r\left[H[f(i, j)]\right]\right\} n_1(i, j)$ and an image-independent random component $n_2(i, j)$.

1.5 Detector and recorder models

The response of image detectors and recorders in general is nonlinear. An example is the response of image scanners

$$r(i, j) = \alpha w(i, j)^{\beta}$$

where α and β are device-dependent constants and $w(i, j)$ is the input blurred image.

For photofilms

$$r(i, j) = \gamma \log_{10} w(i, j) - r_0$$

where γ is called the gamma of the film, $w(i, j)$ is the incident light intensity and $r(i, j)$ is called the optical density. A film is called positive if it has negative γ.

I. Noise models

The general noise model

$$n(i,j) = g\{r[w(i,j)]\}n_1(i,j) + n_2(i,j)$$

is applicable in many situations. Example, in photoelectronic systems we may have $g(x) = \sqrt{x}$. Therefore,

$$n(i,j) = \sqrt{\alpha w(i,j)^{\beta}}\, n_1(i,j) + n_2(i,j)$$

where n_1 and n_2 are zero-mean, mutually independent, Gaussian white noise fields. The term $n_2(i,j)$ may be referred as thermal noise. In the case of films there is no thermal noise and the noise model is

$$n(i,j) = \sqrt{\gamma \log_{10} w(i,j) - r_0}\, n_1(i,j)$$

Because of the signal-dependent term in the noise model, restoration algorithms are quite difficult. Often $w(i,j)$ is replaced by its spatial average, μ_w, giving

$$n(i,j) = g[r[\mu_w]]n_1(i,j) + n_2(i,j)$$

which makes $n(i,j)$ a Gaussian white noise random field. A lineal observation model for photoelectronic devices is

$$y(i,j) = w(i,j) + \sqrt{\mu_w}\, n_1(i,j) + n_2(i,j)$$

For photographic films with $\gamma = -1$

$$y(i,j) = -\log_{10} w(i,j) - r_0 + an_1(x,y)$$

where r_0, a are constants and r_0 can be ignored.

The light intensity associated with the observed optical density $y(i,j)$ is $I(i,j)$

$$= 10^{-y(i,j)} = w(i,j)10^{-an_1(i,j)} = w(i,j)n(i,j)$$

where $n(i,j) = \hat{} 10^{-an_1(i,j)}$ now appears as multiplicative noise having a log-normal distribution.

1.7 A general model of a simplified digital image degradation process

A simplified version for the image restoration process model is $y(i,j) = H[f(i,j)] + n(i,j)$

where

$y(i,j)$ the degraded image

$f(i,j)$ the original image

H an operator that represents the degradation process

$n(i,j)$ the external noise which is assumed to be image-independent

1.8 Possible classification of restoration methods

Restoration methods could be classified as follows:

- **deterministic:** we work with sample by sample processing of the observed (degraded) image
- **stochastic:**we work with the statistics of the images involved in the process

- **non-blind:**the degradation process H is known
- **blind:**the degradation process H is unknown
- **semi-blind:** the degradation process H could be considered partly known

From the viewpoint of implementation:

- **direct**
- **iterative**
- **recursive**

2. Linear position invariant degradation models

2.1 Definition

We again consider the general degradation model

$$y(i, j) = H\big[f(i, j)\big] + n(i, j)$$

If we ignore the presence of the external noise $n(i, j)$ we get

$$y(i, j) = H\big[f(i, j)\big]$$

H is ***linear*** if

$$H\big[k_1 f_1(i, j) + k_2 f_2(i, j)\big] = k_1 H\big[f_1(i, j)\big] + k_2 H\big[f_2(i, j)\big]$$

- is ***position*** (or ***space***) ***invariant*** if

From now on we will deal with linear, space invariant type of degradations.

In a real life problem many types of degradations can be approximated by linear, position invariant processes.

Advantage: Extensive tools of linear system theory become available.

Disadvantage: In some real life problems ***nonlinear*** and ***space variant*** models would be more appropriate for the description of the degradation phenomenon.

2.2 Typical linear position invariant degradation models

- **Motion blur. It occurs when there is relative motion between the object and the camera during exposure.**

$$h(i) = \begin{cases} \dfrac{1}{L}, & \text{if } -\dfrac{L}{2} \le i \le \dfrac{L}{2} \\ 0, & \text{otherwise} \end{cases}$$

- Atmospheric turbulence. It is due to random variations in the reflective index of the medium between the object and the imaging system and it occurs in the imaging of astronomical objects.

$$h(i,j) = K \exp\left[-\frac{i^2 + j^2}{2\sigma} \right]$$

- Uniform out of focus blur

$$h(i,j) = \begin{cases} \frac{1}{\pi R^2}, & \text{if } i^2 + j^2 \le R^2 \\ 0, & \text{otherwise} \end{cases}$$

- Uniform 2-D blur

$$h(i,j) = \begin{cases} \frac{1}{(L)^2}, & \text{if } -\frac{L}{2} \le i, j \le \frac{L}{2} \\ 0, & \text{otherwise} \end{cases}$$

- ...

2.3 Some characteristic metrics for degradation models

- **Blurred Signal-to-Noise Ratio (BSNR)**: a metric that describes the degradation model.

$$\mathrm{BSNR} = 10\log_{10}\left\{\frac{\frac{1}{MN}\sum_i\sum_j [g(i,j) - \bar{g}(i,j)]^2}{\sigma_n^2}\right\}$$

$$g(i,j) = y(i,j) - n(i,j)$$

$$\bar{g}(i,j) = E\{g(i,j)\}$$

σ_n^2 : variance of additive noise

- **Improvement in SNR (ISNR)**: validates the performance of the image restoration algorithm.

$$\mathrm{ISNR} = 10\log_{10}\left\{\frac{\sum_i\sum_j [f(i,j) - y(i,j)]^2}{\sum_i\sum_j [\hat{f}(i,j) - f(i,j)]^2}\right\}$$

where $\hat{f}(i,j)$ is the restored image.

Both BSNR and ISNR can only be used for simulation with artificial data.

2.4 One dimensional discrete degradation model. Circular convolution

Suppose we have a one-dimensional discrete signal $f(i)$ of size A samples $f(0)$, $f(1)$, , $f(A-1)$, which is due to a degradation process. The degradation can be modeled by a one-dimensional discrete impulse response $h(i)$ of size B samples. If we assume that the degradation is a causal function we have the samples $h(0), h(1), , h(B-1)$ .We form the extended versions of $f(i)$ and $h(i)$, both of size $M \geq A + B - 1$ and periodic with period M .

These can be denoted as $f_e(i)$ and $h_e(i)$. For a time invariant degradation process we obtain the discrete convolution formulation as follows

$$y_e(i) = \sum_{m=0}^{M-1} f_e(m) h_e(i-m) + n_e(i)$$

Using matrix notation we can write the following form

$$\mathbf{y} = \mathbf{H}\mathbf{f} + \mathbf{n}$$

$$\mathbf{f} = \begin{bmatrix} f_e(0) \\ f_e(1) \\ \\ f_e(M-1) \end{bmatrix},$$

$$\mathbf{H}_{(M \times M)} = \begin{bmatrix} h_e(0) & h_e(-1) & & h_e(M-1) & h_e(M-2) \\ h_e(1) & h_e(0) & & & \end{bmatrix}$$

At the moment we decide to ignore the external noise we have that

$$\begin{bmatrix} h_e(-M+1) \\ h_e(-M+2) \\ \vdots \\ h_e(0) \end{bmatrix}$$

n . Because h is periodic with period M

$$\mathbf{H}_{(M\times M)} = \begin{bmatrix} h_e(0) & h_e(M-1) & h_e(1) \\ h_e(1) & h_e(0) & h_e(2) \\ \vdots & \vdots & \vdots \\ h_e(M-1) & h_e(M-2) & h_e(0) \end{bmatrix}$$

We define $\lambda(k)$ to be

$$\lambda(k) = h_e(0) + h_e(M-1)\exp(j\frac{2\pi}{M}k) + h_e(M-2)\exp(j\frac{2\pi}{M}2k) +$$

$$+ h_e(1)\exp[j\frac{2\pi}{M}(M-1)k], \qquad k = 0,1, \ldots, M-1$$

Because $\exp[j\frac{2\pi}{M}(M-i)k] = \exp(-j\frac{2\pi}{M}ik)$ we have that

$$\lambda(k) = MH(k)$$

$H(k)$ is the discrete Fourier transform of $h_e(i)$.

I define $\mathbf{w}(k)$ to be

$$\mathbf{w}(k) = \begin{bmatrix} 1 \\ \exp(j\frac{2\pi}{M}k) \\ \vdots \\ \exp[j\frac{2\pi}{M}(M-1)k] \end{bmatrix}$$

It can be seen that

$$\mathbf{H}\mathbf{w}(k) = \lambda(k)\mathbf{w}(k)$$

This implies that $\lambda(k)$ is an eigenvalue of the matrix **H** and $\mathbf{w}(k)$ is its corresponding eigenvector.

We form a matrix **w** whose columns are the eigenvectors of the matrix **H** , that is to say

$$\mathbf{W} = \left[\mathbf{w}(0) \quad \mathbf{w}(1) \quad \cdots \quad \mathbf{w}(M-1)\right]$$

$$w(k,i) = \exp\left\{ j \left\lceil \frac{2\pi}{M} \right\rceil ki \right\} \text{ and } w^{-1}(k,i) = \frac{1}{M} \exp\left\{ -j \left\lceil \frac{2\pi}{M} \right\rceil ki \right\}$$

We can then diagonalize the matrix **H** as follows

$$\mathbf{H} = \mathbf{WDW^{-1}} \Rightarrow \mathbf{D} = \mathbf{W^{-1}HW}$$

where

$$\mathbf{D} = \begin{bmatrix} \lambda(0) & & \mathbf{0} \\ & \lambda(1) & \\ & & \ddots \\ \mathbf{0} & & \lambda(M-1) \end{bmatrix}$$

Obviously **D** is a diagonal matrix and

$$D(k,k) = \lambda(k) = MH(k)$$

If we go back to the degradation model we can write

$$\bullet = \mathbf{Hf} \Rightarrow \mathbf{y} = \mathbf{WDW^{-1}f} \Rightarrow \mathbf{W^{-1}}\,\mathbf{y} = \mathbf{DW^{-1}f}$$
$$\Rightarrow Y(k) = MH(k)F(k),\ k = 0,1,\ ,M-1$$

$= 0,1,\ ,M-1$ are the $M-$ sample discrete Fourier transforms of $y(i), h(i), f(i)$, respectively. So by choosing $\lambda(k)$ and $\mathbf{w}(k)$ as above and assuming that $h_e(i)$ is periodic, we start with a matrix problem and end up with M scalar problems.

Suppose we have a two-dimensional discrete signal $f(i, j)$ of size $A \times B$ samples which is due to a degradation process. The degradation can now be modeled by a two dimensional discrete impulse response $h(i, j)$ of size $C \times D$ samples. We form the extended versions of $f(i, j)$ and $h(i, j)$, both of size $M \times N$, where $M \geq A + C - 1$ and $N \geq B + D - 1$, and periodic with period $M \times N$. These can be denoted as $f_e(i, j)$ and $h_e(i, j)$. For a space invariant degradation process we obtain

$$y_e(i, j) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f_e(m, n) h_e(i - m, j - n) + n_e(i, j)$$

Using matrix notation we can write the following form

$$\mathbf{y} = \mathbf{H}\mathbf{f} + \mathbf{n}$$

where $\mathbf{f}$ and $\mathbf{y}$ are MN – dimensional column vectors that represent the lexicographic ordering of images $f_e(i, j)$ and $h_e(i, j)$ respectively.

$$\mathbf{H} = \begin{bmatrix} \mathbf{H}_0 & \mathbf{H}_{M-1} & \cdots & \mathbf{H}_1 \\ \mathbf{H}_1 & \mathbf{H}_0 & & \mathbf{H}_2 \\ \vdots & & & \vdots \\ \mathbf{H}_{M-1} & \mathbf{H}_{M-2} & \cdots & \mathbf{H}_0 \end{bmatrix}$$

$$\mathbf{H}_j = \begin{bmatrix} h_e(j,0) & h_e(j, N-1) & \cdots & h_e(j,1) \\ h_e(j,1) & h_e(j,0) & & h_e(j,2) \\ \vdots & & & \vdots \\ h_e(j, N-1) & h_e(j, N-2) & \cdots & h_e(j,0) \end{bmatrix}$$

$$\lfloor h_e \rfloor$$

The analysis of the diagonalisation of **H** is a straightforward extension of the one-dimensional case.

In that case we end up with the following set of $M \times N$ scalar problems.

$$Y(u, v) = MNH(u, v)F(u, v)(+N(u, v))$$

$$\bullet = 0,1, \quad , M-1, v = 0,1, \quad , N-1$$

In the general case we may have two functions $f(i), A \le i \le B$ and $h(i), C \le i \le D$, where A, C

can be also negative (in that case the functions are **non-causal**). For the periodic convolution we have to extend the functions from both sides knowing that the convolution is

$g(i) = h(i) * f(i), A + C \le i \le B + D$.

- **Direct deterministic approaches to restoration**

3.1 Inverse filtering

The objective is to minimize

$$J(\mathbf{f}) = \|\mathbf{n}(\mathbf{f})\|^2 = \|\mathbf{y} - \mathbf{H}\mathbf{f}\|^2$$

We set the first derivative of the cost function equal to zero

$$\frac{\partial J(\mathbf{f})}{\partial \mathbf{f}} = 0 \Rightarrow -2\mathbf{H}^T(\mathbf{y} - \mathbf{H}\mathbf{f}) = \mathbf{0}$$

$$\mathbf{f} = (\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T\mathbf{y}$$

If $M = N$ and $\mathbf{H}^{-1}$ exists then

$$\mathbf{f} = \mathbf{H}^{-1}\mathbf{y}$$

According to the previous analysis if **H** (and therefore $\mathbf{H}^{-1}$) is block circulant the above problem can be solved as a set of $M \times N$ scalar problems as follows

$$F(u,v) = \frac{H^*(u,v)Y(u,v)}{|H(u,v)|^2} \Rightarrow f(i,j) = \Im^{-1}\left[\frac{H^*(u,v)Y(u,v)}{|H(u,v)|^2}\right]$$

3.1.1 Computational issues concerning inverse filtering

- Suppose first that the additive noise $n(i, j)$ is negligible. A problem arises if $H(u, v)$ becomes very small or zero for some point (u, v) or for a whole region in the (u, v) plane. In that region inverse filtering cannot be applied. Note that in most real applications $H(u, v)$ drops off rapidly as a function of distance from the origin. The solution is that if these points are known they can be neglected in the computation of $F(u, v)$.

(II) In the presence of external noise we have that

$$\hat{F}(u,v) = \frac{H^*(u,v)\big(Y(u,v)+N(u,v)\big)}{|H(u,v)|^2} =$$

$$\frac{H^*(u,v)Y(u,v)}{|H(u,v)|^2} + \frac{H^*(u,v)N(u,v)}{|H(u,v)|^2} \Rightarrow$$

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

If $H(u, v)$ becomes very small, the term $N(u, v)$ dominates the result. The solution is again to carry out the restoration process in a limited neighborhood about the origin where $H(u, v)$ is not very small. This procedure is called **pseudoinverse filtering**. In that case we set

$$\hat{F}(u,v) = \begin{cases} \dfrac{H^*(u,v)Y(u,v)}{|H(u,v)|^2} & \bullet\ (u,v) \geq T \\ 0 & H(u,v) < T \end{cases}$$

The threshold T is defined by the user. In general, the noise may very well possess large components at high frequencies (u, v) , while $H(u, v)$ and $Y(u, v)$ normally will be

dominated by low frequency components.

3.2 Constrained least squares (CLS) restoration

It refers to a very large number of restoration algorithms.

The problem can be formulated as follows.

minimize

$$J(\mathbf{f}) = \|\mathbf{n}(\mathbf{f})\|^2 = \|\mathbf{y} - \mathbf{H}\mathbf{f}\|^2$$

subject to

$$\|\mathbf{C}\mathbf{f}\|^2 < \varepsilon$$

where **C f** is a high pass filtered version of the image. **The idea behind the above constraint is that the highpass version of the image contains a considerably large amount of noise!** Algorithms of the above type can be handled using optimization techniques. Constrained least squares (CLS) restoration can be formulated by choosing an **f** to minimize the Lagrangian

$$\min\left(\|\mathbf{y} - \mathbf{H}\mathbf{f}\|^2 + \alpha\|\mathbf{C}\mathbf{f}\|^2\right)$$

Typical choice for **C** is the 2-D Laplacian operator given by

$$\mathbf{C} = \begin{bmatrix} 0.00 & -0.25 & 0.00 \\ -0.25 & 1.00 & -0.25 \\ 0.00 & -0.25 & 0.00 \end{bmatrix}$$

□ represents either a Lagrange multiplier or a fixed parameter known as **regularisation**

parameter and it controls the relative contribution between the term $\|\mathbf{y} - \mathbf{H}\mathbf{f}\|^2$

$\|\mathbf{Cf}\|^2$. The minimization of the above leads to the following estimate for the original image $\mathbf{f}$ =

$$\left(\mathbf{H}^T\mathbf{H} + \alpha\mathbf{C}^T\mathbf{C}\right)^{-1}\mathbf{H}^T\mathbf{y}$$

3.2.1 Computational issues concerning the CLS method

3. Choice of α

The problem of the choice of α has been attempted in a large number of studies and different techniques have been proposed. One possible choice is based on a **set theoretic approach**: a restored image is approximated by an image which lies in the intersection of the two ellipsoids defined by

$$Q_{\mathbf{f}|\mathbf{y}} = \{\mathbf{f} \mid \|\mathbf{y} - \mathbf{Hf}\|^2 \le E^2\} \text{ and}$$

$$Q_{\mathbf{f}} = \{\mathbf{f} \mid \|\mathbf{Cf}\|^2 \le \varepsilon^2\}$$

The center of one of the ellipsoids which bounds the intersection of $Q_{\mathbf{f}|\mathbf{y}}$ and $Q_{\mathbf{f}}$, is given by the equation

$$\mathbf{f} = \left(\mathbf{H}^T\mathbf{H} + \alpha\mathbf{C}^T\mathbf{C}\right)^{-1}\mathbf{H}^T\mathbf{y}$$

with $\alpha = (E/\varepsilon)^2$. Another problem is then the choice of E^2 and ε^2. One choice could be

- $= \frac{1}{\text{BSNR}}$

Comments

With larger values of α, and thus more regularisation, the restored image tends to have more ***ringing***. With smaller values of α, the restored image tends to have more ***amplified noise effects***. The variance and bias of the error image in frequency domain are

$$= \sigma_n^2 \sum_{u=0}^{M} \sum_{v=0}^{N} \frac{|H(u,v)|^2}{\left(|H(u,v)|^2 + \alpha |C(u,v)|^2\right)^2}$$

$$= \sigma_n^2 \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} \frac{|F(u,v)|^2 \alpha^2 |C(u,v)|^4}{\left(|H(u,v)|^2 + \alpha |C(u,v)|^2\right)^2}$$

$Var(\alpha)$

$Bias(\alpha)$

The minimum MSE is encountered close to the intersection of the above functions.

A good choice of α is one that gives the best compromise between the variance and bias of the error image.

Iterative deterministic approaches to restoration

They refer to a large class of methods that have been investigated extensively over the last decades. They possess the following advantages.

- There is no need to explicitly implement the inverse of an operator. The restoration process is monitored as it progresses. Termination of the algorithm may take place before convergence.
- The effects of noise can be controlled in each iteration.
- The algorithms used can be spatially adaptive.
- The problem specifications are very flexible with respect to the type of degradation. Iterative techniques can be applied in cases of spatially varying or nonlinear degradations or in cases where the type of degradation is completely unknown (blind restoration).

In general, iterative restoration refers to any technique that attempts to minimize a function of the form

$$M(\mathbf{f})$$

using an updating rule for the partially restored image.

4.1 Least squares iteration

In that case we seek for a solution that minimizes the function

$$M(\mathbf{f}) = \left\| \mathbf{y} - \mathbf{H}\mathbf{f} \right\|^2$$

A necessary condition for $M(\mathbf{f})$ to have a minimum is that its gradient with respect to $\mathbf{f}$ is equal to zero. This gradient is given below

$$\frac{\partial M(\mathbf{f})}{\partial \mathbf{f}} = \nabla_{\mathbf{f}} M(\mathbf{f}) = 2(-\mathbf{H}^T\mathbf{y} + \mathbf{H}^T\mathbf{H}\mathbf{f})$$

and by using the steepest descent type of optimization we can formulate an iterative rule as follows:

$$\mathbf{f}_0 = \beta \mathbf{H}^T \mathbf{y}$$

$$\mathbf{f}_{k+1} = \mathbf{f}_k - \mu \frac{\partial M(\mathbf{f}_k)}{\partial \mathbf{f}_k} = \mathbf{f}_k + \beta \mathbf{H}^T (\mathbf{y} - \mathbf{H}\mathbf{f}_k) = \beta \mathbf{H}^T \mathbf{y} + (\mathbf{I} - \beta \mathbf{H}^T \mathbf{H})\mathbf{f}_k$$

4.2 Constrained least squares iteration

In this method we attempt to solve the problem of constrained restoration iteratively. As already mentioned the following functional is minimized

$$M(\mathbf{f}, \alpha) = \|\mathbf{y} - \mathbf{H}\mathbf{f}\|^2 + \alpha \|\mathbf{C}\mathbf{f}\|^2$$

The necessary condition for a minimum is that the gradient of $M(\mathbf{f}, \alpha)$ is equal to zero. That gradient is

$$\Phi(\mathbf{f}) = \nabla_{\mathbf{f}} M(\mathbf{f}, \alpha) = 2[(\mathbf{H}^T \mathbf{H} + \alpha \mathbf{C}^T \mathbf{C})\mathbf{f} - \mathbf{H}^T \mathbf{y}]$$

The initial estimate and the updating rule for obtaining the restored image are now given by

$$\mathbf{f}_0 = \beta \mathbf{H}^T \mathbf{y}$$

$$\mathbf{f}_{k+1} = \mathbf{f}_k + \beta[\mathbf{H}^T \mathbf{y} - (\mathbf{H}^T \mathbf{H} + \alpha \mathbf{C}^T \mathbf{C})\mathbf{f}_k]$$

It can be proved that the above iteration (known as **Iterative CLS** or **Tikhonov-Miller Method**) converges if

$$0 < \beta < \frac{2}{|\lambda_{max}|}$$

where λ_{max} is the maximum eigenvalue of the matrix

$$(\mathbf{H}^T \mathbf{H} + \alpha \mathbf{C}^T \mathbf{C})$$

If the matrices **H** and **C** are block-circulant the iteration can be implemented in the frequency domain.

4.3 Projection onto convex sets (POCS)

The set-based approach described previously can be generalized so that any number of prior constraints can be imposed as long as the constraint sets are closed convex.

If the constraint sets have a non-empty intersection, then a solution that belongs to the intersection set can be found by the method of POCS. Any solution in the intersection set is consistent with the a priori constraints and therefore it is a feasible solution. Let $Q_1, Q_2, \ldots, Q_m$ be closed convex sets in a finite dimensional vector space, with $P_1, P_2, \ldots, P_m$ their respective projectors. The iterative procedure

$$\mathbf{f}_{k+1} = P_1 P_2 \ldots P_m \mathbf{f}_k$$

converges to a vector that belongs to the intersection of the sets Q_i, $i = 1,2,\ldots,m$, for any starting vector $\mathbf{f}_0$. An iteration of the form $\mathbf{f}_{k+1} = P_1 P_2 \mathbf{f}_k$ can be applied in the problem described previously, where we seek for an image which lies in the intersection of the two ellipsoids defined by

$$Q_{\mathbf{f}|\mathbf{y}} = \{\mathbf{f} \,|\, \|\mathbf{y} - \mathbf{H}\mathbf{f}\|^2 \le E^2\} \text{ and } Q_{\mathbf{f}} = \{\mathbf{f} \,|\, \|\mathbf{C}\mathbf{f}\|^2 \le \varepsilon^2\}$$

The respective projections $P_1\mathbf{f}$ and $P_2\mathbf{f}$ are defined by

$$P_1\mathbf{f} = \mathbf{f} + \lambda_1 \left(\mathbf{I} + \lambda_1 \mathbf{H}^T\mathbf{H}\right)^{-1} \mathbf{H}^T (\mathbf{y} - \mathbf{H}\mathbf{f})$$

$$P_2\mathbf{f} = [\mathbf{I} - \lambda_2 \left(\mathbf{I} + \lambda_2 \mathbf{C}^T\mathbf{C}\right)^{-1} \mathbf{C}^T\mathbf{C}]\mathbf{f}$$

4.4 Spatially adaptive iteration

The functional to be minimized takes the form

$$M(\mathbf{f}, \alpha) = \|\mathbf{y} - \mathbf{H}\mathbf{f}\|^2_{\mathbf{w}_1} + \alpha \|\mathbf{C}\mathbf{f}\|^2_{\mathbf{w}_2}$$

where

$$\|\mathbf{y} - \mathbf{Hf}\|^2_{w1} = (\mathbf{y} - \mathbf{Hf})^T \mathbf{W_1} (\mathbf{y} - \mathbf{Hf})$$

$$\|\mathbf{Cf}\|^2_{w2} = (\mathbf{Cf})^T \mathbf{W_2} (\mathbf{Cf})$$

$\mathbf{W_1}$,$\mathbf{W_2}$ are diagonal matrices, the choice of which can be justified in various ways. The entries in both matrices are non-negative values and less than or equal to unity. In that case

$$\Phi(\mathbf{f}) = \nabla_{\mathbf{f}} M(\mathbf{f}, \alpha) = (\mathbf{H}^T \mathbf{W_1 H} + \alpha \mathbf{C}^T \mathbf{W_2 C})\mathbf{f} - \mathbf{H}^T \mathbf{W_1 y}$$

A more specific case is

$$M(\mathbf{f}, \alpha) = \|\mathbf{y} - \mathbf{Hf}\|^2 + \alpha \|\mathbf{Cf}\|^2_{\mathbf{w}}$$

where the weighting matrix is incorporated only in the regularization term. This method is known as **weighted regularised image restoration**. The entries in matrix **W** will be chosen so that the high-pass filter is only effective in the areas of low activity and a very little smoothing takes place in the edge areas.

4.5 Robust functionals

Robust functionals allow for the efficient supression of a wide variety of noise processes and permit the reconstruction of sharper edges than their quadratic counterparts. We are seeking to minimize

$$M(\mathbf{f}, \alpha) = R_n(\mathbf{y} - \mathbf{Hf}) + \alpha R_x \mathbf{Cf}$$

$R_n()$, $R_x()$ are referred to as **residual** and **stabilizing** functionals respectively.

4.6 Computational issues concerning iterative techniques

- Convergence

  The **contraction mapping theorem** usually serves as a basis for establishing convergence of iterative algorithms. According to it iteration

  $\mathbf{f_0} = \mathbf{0}$

  $\mathbf{f_{k+1}} = \mathbf{f_k} + \beta\Phi(\mathbf{f_k}) = \Psi(\mathbf{f_k})$

converges to a unique fixed point $\mathbf{f}^*$, that is, a point such that $\Psi(\mathbf{f}^*) = \mathbf{f}^*$, for any initial vector, if the operator or transformation $\Psi(\mathbf{f})$ is a **contraction**. This means that for any two vectors $\mathbf{f_1}$ and $\mathbf{f_2}$ in the domain of $\Psi(\mathbf{f})$ the following relation holds

$$\left\|\Psi(\mathbf{f_1}) - \Psi(\mathbf{f_2})\right\| \leq \eta \left\|\mathbf{f_1} - \mathbf{f_2}\right\|$$

with $\eta \leq 1$ and $\|.\|$ any norm. The above condition is **norm dependent.**

v Rate of convergence

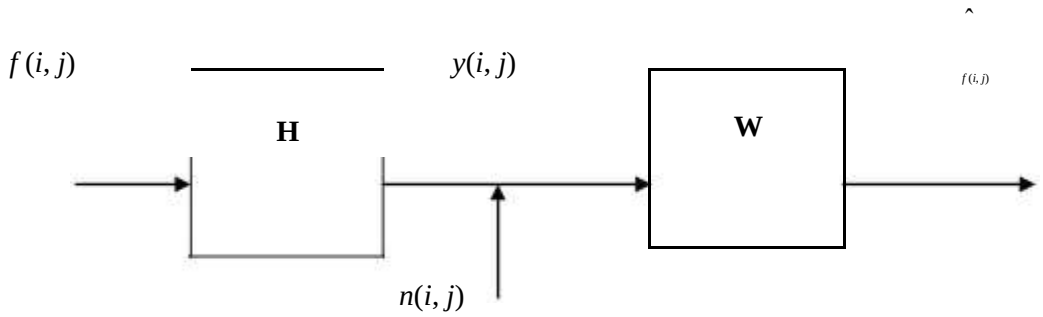
The termination criterion most frequently used compares the normalized change in energy at each iteration to a threshold such as

$$\frac{\left\|\mathbf{f}_{k+1} - \mathbf{f}_k\right\|_2}{\left\|\mathbf{f}_k\right\|_2} \leq 10^{-6}$$

g Stochastic approaches to restoration

5.1 Wiener estimator (stochastic regularisation)

The image restoration problem can be viewed as a system identification problem as follows:



The objective is to minimize the following function

$$E\{(\hat{\mathbf{f}} - \mathbf{f})^T (\hat{\mathbf{f}} - \mathbf{f})\}$$

To do so the following conditions should hold:

(i) $E\{\hat{\mathbf{f}}\} = E\{\mathbf{f}\} \Rightarrow E\{\hat{\mathbf{f}}\} = \mathbf{W}E\{\mathbf{y}\}$

(ii) The error must be orthogonal to the observation about the mean

$$E\{(\mathbf{f} - \hat{\mathbf{f}})(\mathbf{y} - E\{\mathbf{y}\})^{\mathbf{T}}\} = 0$$

From (i) and (ii) we have that

$$E\{(\mathbf{Wy} - \mathbf{f})(\mathbf{y} - E\{\mathbf{y}\})^{\mathbf{T}}\} = 0 \Rightarrow E\{(\mathbf{Wy} + E\{\mathbf{f}\} - \mathbf{W}E\{\mathbf{y}\} - \mathbf{f})(\mathbf{y} - E\{\mathbf{y}\})^{\mathbf{T}}\} = 0 \Rightarrow$$

$$E\{[\mathbf{W}(\mathbf{y} - E\{\mathbf{y}\}) - (\mathbf{f} - E\{\mathbf{f}\})](\mathbf{y} - E\{\mathbf{y}\})^{\mathbf{T}}\} = 0$$

If $\tilde{\mathbf{y}} = \mathbf{y} - E\{\mathbf{y}\}$ and $\tilde{\mathbf{f}} = \mathbf{f} - E\{\mathbf{f}\}$ then

$$E\{(\mathbf{W}\tilde{\mathbf{y}} - \tilde{\mathbf{f}})\tilde{\mathbf{y}}^{\mathbf{T}}\} = 0 \Rightarrow E\{\mathbf{W}\tilde{\mathbf{y}}\tilde{\mathbf{y}}^{\mathbf{T}}\} = E\{\tilde{\mathbf{f}}\tilde{\mathbf{y}}^{\mathbf{T}}\} \Rightarrow \mathbf{W}E\{\tilde{\mathbf{y}}\tilde{\mathbf{y}}^{\mathbf{T}}\} = E\{\tilde{\mathbf{f}}\tilde{\mathbf{y}}^{\mathbf{T}}\} \Rightarrow \mathbf{WR}_{\tilde{\mathbf{y}}\tilde{\mathbf{y}}} = \mathbf{R}_{\tilde{\mathbf{f}}\tilde{\mathbf{y}}}$$

If the original and the degraded image are both zero mean then $\mathbf{R}_{\tilde{\mathbf{y}}\tilde{\mathbf{y}}} = \mathbf{R}_{\mathbf{yy}}$ and $\mathbf{R}_{\tilde{\mathbf{f}}\tilde{\mathbf{y}}} = \mathbf{R}_{\mathbf{fy}}$. In that case we have that $\mathbf{WR}_{\mathbf{yy}} = \mathbf{R}_{\mathbf{fy}}$. If we go back to the degradation model and find the autocorrelation matrix of the degraded image then we get that

$$\mathbf{J} = \mathbf{Hf} + \mathbf{n} \Rightarrow \mathbf{y}^{\mathbf{T}} = \mathbf{f}^{\mathbf{T}}\mathbf{H}^{\mathbf{T}} + \mathbf{n}^{\mathbf{T}}$$

$$E\{\mathbf{yy}^{\mathbf{T}}\} = \mathbf{HR}_{\mathbf{ff}}\mathbf{H}^{\mathbf{T}} + \mathbf{R}_{\mathbf{nn}} = \mathbf{R}_{\mathbf{yy}}$$

$$E\{\mathbf{fy}^{\mathbf{T}}\} = \mathbf{R}_{\mathbf{ff}}\mathbf{H}^{\mathbf{T}} = \mathbf{R}_{\mathbf{fy}}$$

From the above we get the following results

$$\hat{\mathbf{g}} = \mathbf{R}_{\mathbf{fy}}\mathbf{R}_{\mathbf{yy}}^{-\mathbf{1}} = \mathbf{R}_{\mathbf{ff}}\mathbf{H}^{\mathbf{T}}(\mathbf{HR}_{\mathbf{ff}}\mathbf{H}^{\mathbf{T}} + \mathbf{R}_{\mathbf{nn}})^{-\mathbf{1}}$$

$$\hat{\mathbf{f}} = \mathbf{R}_{\mathbf{ff}}\mathbf{H}^{\mathbf{T}}(\mathbf{HR}_{\mathbf{ff}}\mathbf{H}^{\mathbf{T}} + \mathbf{R}_{\mathbf{nn}})^{-\mathbf{1}}\mathbf{y}$$

Note that knowledge of $\mathbf{R}_{\mathbf{ff}}$ and $\mathbf{R}_{\mathbf{nn}}$ is assumed. In frequency domain

$$W(u,v) = \frac{S_{ff}(u,v)H^{*}(u,v)}{S_{ff}(u,v)\left|H(u,v)\right|_{2} + S_{nn}(u,v)}$$

$$\hat{F}(u,v) = \frac{S_{ff}(u,v)H^*(u,v)}{S_{ff}(u,v)\left|H(u,v)\right|^2 + S_{nn}(u,v)} Y(u,v)$$

5.1.1 Computational issues

The noise variance has to be known, otherwise it is estimated from a flat region of the observed image. In practical cases where a single copy of the degraded image is available, it is quite common to use $S_{yy}(u,v)$ as an estimate of $S_{ff}(u,v)$. **This is very often a poor estimate.**

5.1.2 Wiener smoothing filter

In the absence of any blur, $H(u,v)=1$ and

$$W(u,v) = \frac{S_{ff}(u,v)}{S_{ff}(u,v) + S_{nn}(u,v)} = \frac{(SNR)}{(SNR)+1}$$

h $(SNR) >> 1 \Rightarrow W(u,v) \cong 1$

i $(SNR) << 1 \Rightarrow W(u,v) \cong (SNR)$

(SNR) is high in low spatial frequencies and low in high spatial frequencies so $W(u,v)$ can be implemented with a lowpass (smoothing) filter.

5.1.3 Relation with inverse filtering

If $S_{nn}(u,v) = 0 \Rightarrow W(u,v) = \frac{1}{H(u,v)}$ which is the inverse filter

$$\text{If } S_{nn}(u,v) \to 0 \Rightarrow \lim W(u,v) = \begin{cases} \frac{1}{H(u,v)} & H(u,v) \neq 0 \\ 0 & H(u,v) = 0 \end{cases}$$

$S_{nn} \to 0$

which is the pseudoinverse filter.

5.1.4 Iterative Wiener filters

They refer to a class of iterative procedures that successively use the Wiener filtered signal as an improved prototype to update the covariance estimates of the original image as follows.

Step 0: Initial estimate of $\mathbf{R}_{\mathbf{ff}}$

$$\mathbf{R}_{\mathbf{ff}}(0) = \mathbf{R}_{\mathbf{yy}} = E\{\mathbf{y}\mathbf{y}^{\mathbf{T}}\}$$

Step 1: Construct the i^{th} restoration filter

$$\mathbf{W}(i+1) = \mathbf{R}_{\mathbf{ff}}(i)\mathbf{H}^{\mathbf{T}}(\mathbf{H}\mathbf{R}_{\mathbf{ff}}(i)\mathbf{H}^{\mathbf{T}} + \mathbf{R}_{\mathbf{nn}})^{-1}$$

Step 2: Obtain the $(i+1)^{\text{th}}$ estimate of the restored image

$$\hat{\mathbf{f}}(i+1) = \mathbf{W}(i+1)\mathbf{y}$$

Step 3: Use $\hat{\mathbf{f}}(i+1)$ to compute an improved estimate of $\mathbf{R}_{\mathbf{ff}}$ given by

$$\mathbf{R}_{\mathbf{ff}}(i+1) = E\{\hat{\mathbf{f}}(i+1)\hat{\mathbf{f}}^{\mathbf{T}}(i+1)\}$$

Step 4: Increase i and repeat steps 1,2,3,4.

6. Pseudo inverse – Singular value decomposition

(*Remaining topics clarification please refer Gonzalez-digital image processing*)

UNIT IV

IMAGE COMPRESSION

Lossless compression: Variable length coding – LZW coding – Bit plane coding-predictive coding-DPCM.

Lossy Compression: Transform coding – Wavelet coding – Basics of Image

compression standards: JPEG, MPEG, Basics of Vector quantization.

COMPRESSION FUNDAMENTALS

Introduction

In recent years, there have been significant advancements in algorithms and architectures for the processing of image, video, and audio signals. These advancements have proceeded along several directions. On the algorithmic front, new techniques have led to the development of robust methods to reduce the size of the image, video, or audio data. Such methods are extremely vital in many applications that manipulate and store digital data. Informally, we refer to the process of size reduction as a compression process. We will define this process in a more formal way later. On the architecture front, it is now feasible to put sophisticated compression processes on a relatively low-cost single chip; this has spurred a great deal of activity in developing multimedia systems for the large consumer market.

One of the exciting prospects of such advancements is that multimedia information comprising image, video, and audio has the potential to become just another data type. This usually implies that multimedia information will be digitally encoded so that it can be manipulated, stored, and transmitted along with other digital data types. For such data usage to be pervasive, it is essential that the data encoding is standard across different platforms and applications. This will foster widespread development of applications and will also promote interoperability among systems from different vendors. Furthermore, standardisation can lead to the development of cost-

effective implementations, which in turn will promote the widespread use of multimedia information. This is the primary motivation behind the emergence of image and video compression standards.

Background

Compression is a process intended to yield a compact digital representation of a signal. In the literature, the terms *source coding, data compression, bandwidth compression,* and *signal compression* are all used to refer to the process of compression. In the cases where the signal is defined as an image, a video stream, or an audio signal, the generic problem of compression is to minimise the bit rate of their digital representation. There are many applications that benefit when image, video, and audio signals are available in compressed form. **Without compression, most of these applications would not be feasible!**

Example 1: Let us consider **facsimile image transmission**. In most facsimile machines, the document is scanned and digitised. Typically, an 8.5x11 inches page is scanned at 200 dpi; thus, resulting in 3.74 Mbits. Transmitting this data over a low-cost 14.4 kbits/s modem would require 5.62 minutes. With compression, the transmission time can be reduced to 17 seconds. This results in substantial savings in transmission costs.

Example 2: Let us consider a video-based CD-ROM application. **Full-motion video**, at 30 fps and a 720 x 480 resolution, generates data at 20.736 Mbytes/s. At this rate, only 31 seconds of video can be stored on a 650 MByte CD-ROM. Compression technology can increase the storage capacity to 74 minutes, for VHS-grade video quality.

Image, video, and audio signals are amenable to compression due to the factors below.

g There is considerable statistical redundancy in the signal.

Within a single image or a single video frame, there exists significant correlation among neighbour samples. This correlation is referred to as *spatial correlation.*

For data acquired from multiple sensors (such as satellite images), there exists significant correlation amongst samples from these sensors. This correlation is referred to as *spectral correlation.*

For temporal data (such as video), there is significant correlation amongst samples in different segments of time. This is referred to as *temporal correlation.*

- **There is considerable information in the signal that is irrelevant from a perceptual point of view.**

- **Some data tends to have high-level features that are redundant across space and time; that is, the data is of a fractal nature.**

For a given application, compression schemes may exploit any one or all of the above factors to achieve the desired compression data rate.

There are many applications that benefit from data compression technology. Table 1.1 lists a representative set of such applications for image, video, and audio data, as well as typical data rates of the corresponding compressed bit streams. Typical data rates for the uncompressed bit streams are also shown.

Application	Data Rate	
	Uncompressed	Compressed
Voice 8 ksamples/s, 8 bits/sample	64 kbps	2-4 kbps
Slow motion video (10fps) framesize 176x120, 8bits/pixel	5.07 Mbps	8-16 kbps
Audio conference 8 ksamples/s, 8 bits/sample	64 kbps	16-64 kbps
Video conference (15fps) framesize 352x240, 8bits/pixel	30.41 Mbps	64-768 kbps
Digital audio 44.1 ksamples/s, 16 bits/sample	1.5 Mbps	1.28-1.5 Mbps
Video file transfer (15fps) framesize 352x240, 8bits/pixel	30.41 Mbps	384 kbps
Digital video on CD-ROM (30fps) framesize 352x240, 8bits/pixel	60.83 Mbps	1.5-4 Mbps
Broadcast video (30fps) framesize 720x480, 8bits/pixel	248.83 Mbps	3-8 Mbps

HDTV (59.94 fps)	1.33 Gbps	20 Mbps
framesize 1280x720, 8bits/pixel		

Table 1.1: Applications for image, video, and audio compression.

In the following figure, a systems view of the compression process is depicted.

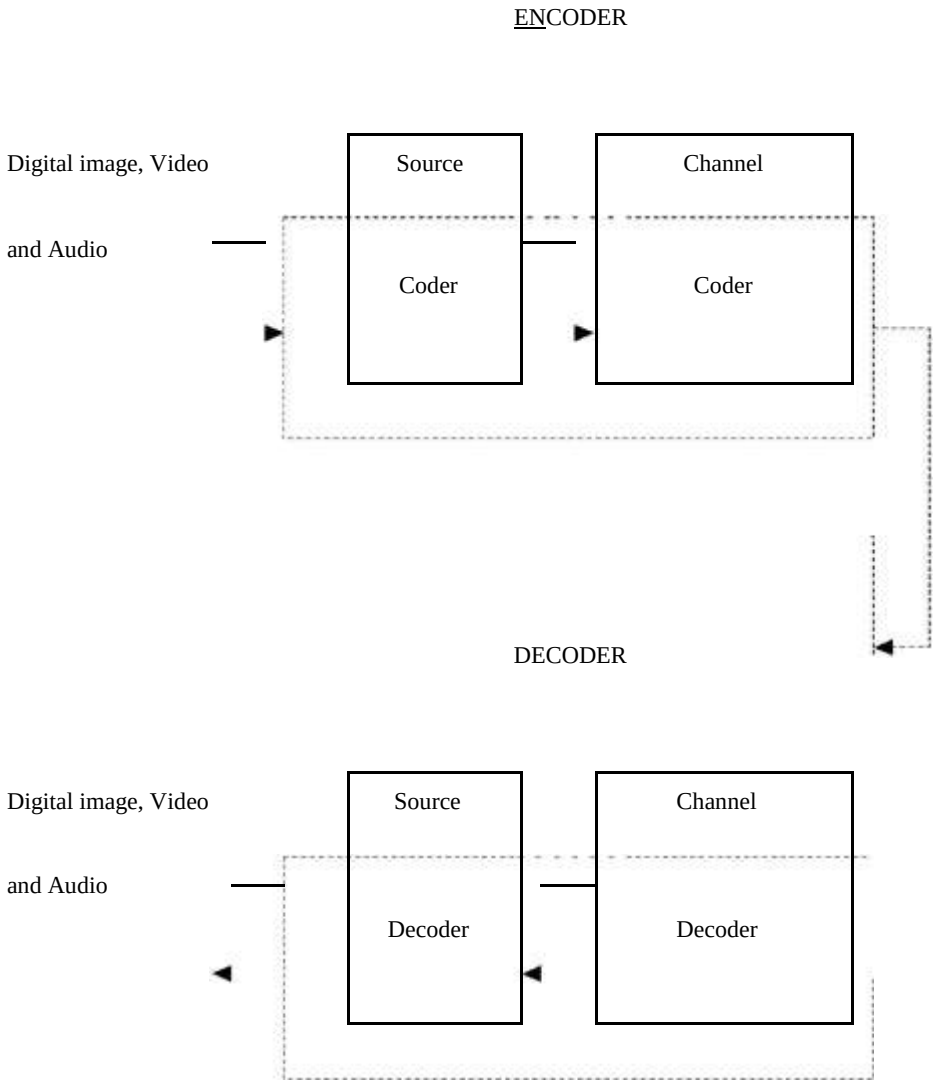


Figure 1.1 Generic compression system

The core of the encoder is the source coder. The source coder performs the compression process by reducing the input data rate to a level that can be supported by the storage or transmission medium. The bit rate output of the encoder is measured in bits per sample or bits per second. For image or video data, a pixel is the basic element; thus, bits per sample is also referred to as bits per pixel or bits per pel. In the literature, the term *compression ratio,* denoted as c_r , is also used instead of *bit rate* to characterise the capability of the compression system. An intuitive definition

of c_r is

$$c_r = \frac{\textit{source coder input size}}{\textit{source coder output size}}$$

This definition is somewhat ambiguous and depends on the data type and the specific compression method that is employed. For a still-image, size could refer to the bits needed to represent the entire image. For video, size could refer to the bits needed to represent one frame of video. Many compression methods for video do not process each frame of video, hence, a more commonly used notion for size is the bits needed to represent one second of video.

In a practical system, the source coder is usually followed by a second level of coding: the channel coder (Figure 1.1). The channel coder translates the compressed bit stream into a signal suitable for either storage or transmission. In most systems, source coding and channel coding are distinct processes. In recent years, methods to perform combined source and channel coding have also been developed. Note that, in order to reconstruct the image, video, or audio signal, one needs to reverse the processes of channel coding and source coding. This is usually performed at the decoder.

From a system design viewpoint, one can restate the compression problem as a bit rate minimisation problem, where several constraints may have to be met, including the following:

- **Specified level of signal quality**. This constraint is usually applied at the decoder.
- **Implementation complexity**. This constraint is often applied at the decoder, and in some instances at both the encoder and the decoder.
- **Communication delay**. This constraint refers to the end to end delay, and is measured from the start of encoding a sample to the complete decoding of that sample.

Note that, these constraints have different importance in different applications. For example, in a two-way teleconferencing system, the communication delay might be the major constraint, whereas, in a television broadcasting system, signal quality and decoder complexity might be the main constraints.

Lossless versus lossy compression

Lossless compression

In many applications, the decoder has to reconstruct without any loss the original data. For a lossless compression process, the reconstructed data and the original data must be identical in value for each and every data sample. This is also referred to as a reversible process. In lossless compression, for a specific application, the choice of a compression method involves a trade-off along the three dimensions depicted in Figure 1.2; that is, coding efficiency, coding complexity, and coding delay.

Coding Efficiency

This is usually measured in bits per sample or bits per second (bps). Coding efficiency is usually limited by the information content or *entropy* of the source. In intuitive terms, the entropy of a source X provides a measure for the "randomness" of X. From a compression theory point of view, sources with large entropy are more difficult to compress (for example, random noise is very hard to compress).

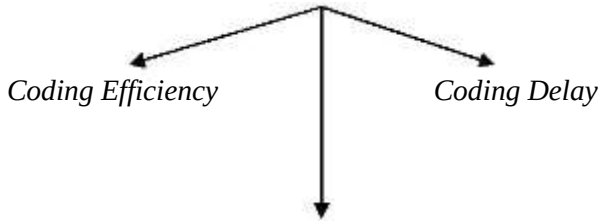
Coding Complexity

The complexity of a compression process is analogous to the computational effort needed to implement the encoder and decoder functions. The computational effort is usually measured in terms of memory requirements and number of arithmetic operations. The operations count is characterised by the term millions of operations per second and is often referred to as MOPS. Here, by operation, we imply a basic arithmetic operation that is supported by the computational engine. In the compression literature, the term MIPS (millions of instructions per second) is sometimes used. This is specific to a computational engine's architecture; thus, in this text we refer to coding complexity in terms of MOPS. In some applications, such as portable devices, coding complexity may be characterised by the power requirements of a hardware implementation.

Coding Delay

A complex compression process often leads to increased coding delays at the encoder and the decoder. Coding delays can be alleviated by increasing the processing power of the computational engine; however, this may be impractical in environments where there is a power constraint or when the underlying computational engine cannot be improved. Furthermore, in many applications, coding delays have to be constrained; for example, in interactive communications. The need to constrain the coding delay often forces the compression system designer to use a less sophisticated algorithm for the compression processes.

From this discussion, it can be concluded that these trade-offs in coding complexity, delay, and efficiency are usually limited to a small set of choices along these axes. In a subsequent section, we will briefly describe the trade-offs within the context of specific lossless compression methods.



Coder Complexity

-Compression Ratio?

-Memory requirements?

-Power requirements?

-Operations per second?

Figure 1.2 Trade-offs in lossless compression.

Lossy compression

The majority of the applications in image or video data processing do not require that the reconstructed data and the original data are identical in value. Thus, some amount of loss is permitted in the reconstructed data. A compression process that results in an imperfect reconstruction is referred to as a lossy compression process. This compression process is irreversible. In practice, most irreversible compression processes degrade rapidly the signal quality when they are repeatedly applied on previously decompressed data.

The choice of a specific lossy compression method involves trade-offs along the four dimensions shown in Figure 1.3. Due to the additional degree of freedom, namely, in the signal quality, a lossy compression process can yield higher compression ratios than a lossless compression scheme.

Signal Quality This term is often used to characterise the signal at the output of the decoder.

There is no universally accepted measure for signal quality.

One measure that is often cited is the signal to noise ratio *SNR* , which can be expressed as

$$SNR = 10\log_{10}\frac{\text{encoder input signal energy}}{\text{noise signal energy}}$$

The noise signal energy is defined as the energy measured for a hypothetical signal that is the difference between the encoder input signal and the decoder output signal. Note that, *SNR* as defined here is given in decibels (dB). In the case of images or video, *PSNR* (peak signal-to-noise ratio) is used instead of *SNR* . The calculations are essentially the same as in the case of *SNR* , however, in the numerator, instead of using the encoder input signal one uses a hypothetical signal with a signal strength of 255 (the maximum decimal value of an unsigned 8-bit number, such as in a pixel).

High *SNR* or *PSNR* values do not always correspond to signals with perceptually high quality. Another measure of signal quality is the mean opinion score, where the performance of a

compression process is characterised by the subjective quality of the decoded signal. For instance, a five point scale such as *very annoying, annoying, slightly annoying, perceptible but not annoying,* and *imperceptible* might be used to characterise the impairments in the decoder output.

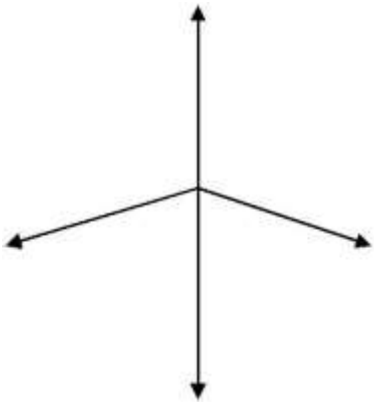
In either lossless or lossy compression schemes, the quality of the input data affects the compression ratio. For instance, acquisition noise, data sampling timing errors, and even the analogue-to-digital conversion process affects the signal quality and reduces the spatial and temporal correlation. Some compression schemes are quite sensitive to the loss in correlation and may yield significantly worse compression in the presence of noise.

Signal Quality

-Bit error probability?

-SNR?

-Mean opinion score?



Coding Efficiency *Coding Delay*

-Compression Ratio?

Coder Complexity

-Memory requirements?

-Power requirements?

-Operations per second?

Figure 1.3 Trade-offs in lossy compression.

Issues in compression method selection

In this chapter, we have introduced some fundamental concepts related to image, video, and audio compression. When choosing a specific compression method, one should consider the following issues:

ω Lossless or lossy. This is usually dictated by the coding efficiency requirements.

ω Coding efficiency. Even in a lossy compression process, the desirable coding efficiency might not be achievable. This is especially the case when there are specific constraints on output signal quality.

ω Variability in coding efficiency. In some applications, large variations in coding efficiency among different data sets may not be acceptable.

ω Resilience to transmission errors. Some compression methods are more robust to transmission errors than others. If retransmissions are not permitted, then this requirement may impact on the overall encoder- decoder design.

ω Complexity trade-offs. In most implementations, it is important to keep the overall encoder-decoder complexity low. However, certain applications may require only a low decoding complexity.

ω Nature of degradations in decoder output. Lossy compression methods introduce artifacts in the decoded signal. The nature of artifacts depends on the compression method that is employed. The degree to which these artifacts are judged also varies from application to application. In communication systems, there is often an interplay between the transmission

errors and the coding artifacts introduced by the coder. Thus, it is important to consider all types of error in a system design.

- ω Data representation. In many applications, there is a need to support two decoding phases. In the first phase, decoding is performed to derive an intelligible signal; this is the case in data browsing. In the second phase, decoding is performed to derive a higher quality signal. One can generalise this notion to suggest that some applications require a hierarchical representation of the data. In the compression context, we refer to such compression schemes as *scalable compression methods.* The notion of scalability has been adopted in the compression standards.

- ω Multiple usage of the encoding-decoding tandem. In many applications, such as video editing, there is a need to perform multiple encode-decode operations using results from a previous encode-decode operation. This is not an issue for lossless compression; however, for lossy schemes, resilience to multiple encoding-decoding cycles is essential.

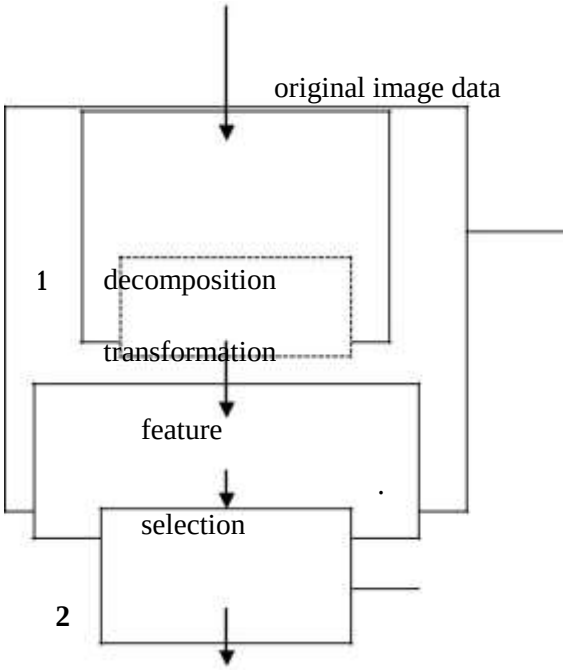
- ω Interplay with other data modalities, such as audio and video. In a system where several data modalities have to be supported, the compression methods for each modality should have some common elements. For instance, in an interactive videophone system, the audio compression method should have a frame structure that is consistent with the video frame structure. Otherwise, there will be unnecessary requirements on buffers at the decoder and a reduced tolerance to timing errors.

- ω Interworking with other systems. In a mass-market environment, there will be multiple data modalities and multiple compression systems. In such an environment, transcoding from one compression method to another may be needed. For instance, video editing might be done on a frame by frame basis; hence, a compression method that does not exploit temporal redundancies might be used here. After video editing, there might be a need to broadcast this video. In this case, temporal redundancies can be exploited to achieve a higher coding efficiency. In such a scenario, it is important to select compression methods that support transcoding from one compressed stream format to another. Interworking is important in many communications environments as well.

THE SOURCE CODER

In this course we are interested in exploring various compression techniques referring to the **source coder** only, where the image compression takes place.

A general procedure for image data compression is shown in the following block diagram (Figure 1.4). **This is the source coder of the previous diagram!**



Compressed image

ELEMENTS OF INFORMATION THEORY

ω Any information generating process can be viewed as a source that emits a sequence of symbols chosen from a finite alphabet (for example, text: ASCII symbols; n -bit images: 2^n symbols).

ω Simplest form of an information source: *discrete memoryless source* (DMS). Successive symbols produced by such a source are statistically independent.

ω A DMS is completely specified by the source alphabet $S = \{s_1, s_2, , s_n\}$ and the associated probabilities $\{p_1, p_2, , p_n\}$.

ω *Self Information:*

$$I(s) = \log \frac{1}{p_i} = -\log_2 p_i$$

the occurrence of a less probable event provides more information

the information of independent events taken as a single event equals the sum of the information

ζ *Average Information* per Symbol or *Entropy* of a DMS:

$$H(S) = \sum_{i=1}^{n} p_i I(s_i) = -\sum_{i=1}^{n} p_i \log_2 p_i \text{ bits/symbol}$$

ξι Interpretation of Entropy:

Average amount of information per symbol provided by the source (definition)

Average amount of information per symbol an observer needs to spend to remove the uncertainty in the source

ξιι N – th extention of the DMS: Given a DMS of size n, group the source into blocks of N symbols. Each block can now be considered as a single source symbol generated by a source S^N with alphabet size n^N. In this case

$$H(S^N) = N \times H(S)$$

Noiseless Source Coding Theorem

Let S be a source with alphabet size n and entropy $H(S)$. Consider coding blocks of N source symbols into binary codewords. For any $\delta > 0$, it is possible by choosing N large enough to construct a code in such a way that the average number of bits per original source symbol l_{avg} satisfies

$$H(s) \le l_{avg} \le H(s) + \delta$$

METHODS FOR LOSSLESS COMPRESSION

1 PRELIMINARIES

Lossless compression refers to compression methods for which the original uncompressed data set can be recovered exactly from the compressed stream. The need for lossless compression arises from the fact that many applications, such as the compression of digitized medical data, require that no loss be introduced from the compression method. Bitonal image transmission via a facsimile device also imposes such requirements. In recent years, several compression standards have been developed for the lossless compression of such images. We discuss these standards later. In general, even when lossy compression is allowed, the overall compression scheme may be a combination of a lossy compression process followed by a lossless compression process. Various image, video, and audio compression standards follow this model, and several of the lossless compression schemes used in these standards are described in this section.

The general model of a lossless compression scheme is as depicted in the following figure.

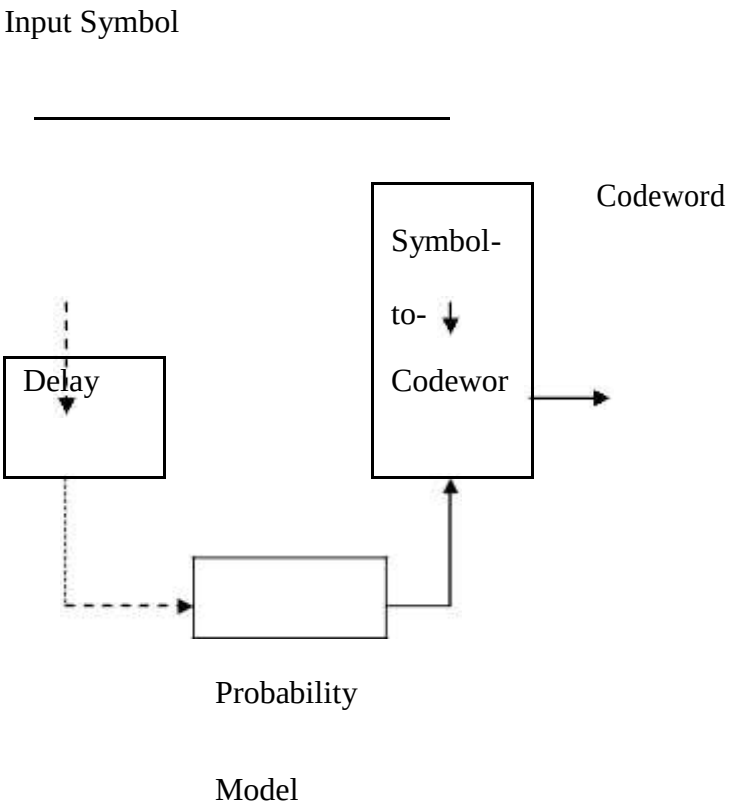


Figure 1.1: A generic model for lossless compression

Given an input set of symbols, a modeler generates an estimate of the probability distribution of the input symbols. This probability model is then used to map symbols into codewords. The combination of the probability modeling and the symbol-to-codeword mapping functions is usually referred to as **entropy coding**. The key idea of entropy coding is to use short codewords

for symbols that occur with high probability and long codewords for symbols that occur with low probability.

The probability model can be derived either from the input data or from a priori assumptions about the data. Note that, for decodability, the same model must also be generated by the decoder. Thus, if the model is dynamically estimated from the input data, causality constraints require a delay function between the input and the modeler. If the model is derived from a priori assumptions, then the delay block is not required; furthermore, the model function need not have access to the input symbols. The probability model does not have to be very accurate, but the more accurate it is, the better the compression will be. Note that, compression is not always guaranteed. If the probability model is wildly inaccurate, then the output size may even expand. However, even then the original input can be recovered without any loss.

Decompression is performed by reversing the flow of operations shown in the above Figure 1.1.

This decompression process is usually referred to as **entropy decoding**.

Message-to-Symbol Partitioning

As noted before, entropy coding is performed on a symbol by symbol basis. Appropriate partitioning of the input messages into symbols is very important for efficient coding. For example, typical images have sizes from 256× 256 pixels to 64000×64000 pixels. One could view one instance of a 256× 256 multi-frame image as a single message, $256^2 = 65536$ long; however, it is very difficult to provide probability models for such long symbols. In practice, we typically view any image as a string of symbols. In the case of a 256× 256 image, if we assume that each pixel takes values between zero and 255, then this image can be viewed as a sequence of symbols drawn from the alphabet 0,1,2, ,255. The modeling problem now reduces to finding a good probability model for the 256 symbols in this alphabet.

For some images, one might partition the data set even further. For instance, if we have an image with 12 bits per pixel, then this image can be viewed as a sequence of symbols drawn from the alphabet 0,1, ,4095. Hardware and/or software implementations of the lossless compression methods may require that data be processed in 8–, 16–, 32–, or 64 – bit units. Thus, one approach

might be to take the stream of 12 – bit pixels and artificially view it as a sequence of 8 – bit symbols. In this case, we have reduced the alphabet size. This reduction compromises the achievable compression ratio; however, the data are matched to the processing capabilities of the computing element.

Other data partitions are also possible; for instance, one may view the data as a stream of 24 – bit symbols. This approach may result in higher compression since we are combining two pixels into one symbol. In general, the partitioning of the data into blocks, where a block is composed of several input units, may result in higher compression ratios, but also increases the coding complexity.

Differential Coding

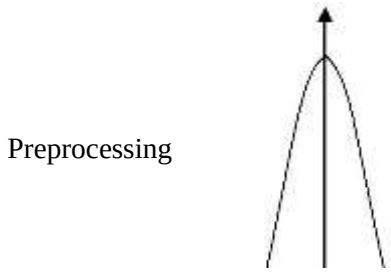
Another preprocessing technique that improves the compression ratio is differential coding. Differential coding skews the symbol statistics so that the resulting distribution is more amenable to compression. Image data tend to have strong inter-pixel correlation. If, say, the pixels in the

image are in the order $x_1, x_2, x_3, \ , x_N$, then instead of compressing these pixels, one might process the sequence of differentials $y_i = x_i - x_{i-1}$, where $i = 1,2, \ , N$, and $x_0 = 0$. In compression terminology, y_i is referred to as the **prediction residual** of x_i. The notion of

compressing the prediction residual instead of x_i is used in all the image and video compression standards. For images, a typical probability distribution for x_i and the resulting distribution for y_i are shown in Figure 1.2.

Let symbol s_i have a probability of occurrence p_i. From coding theory, the ideal symbol-to-codeword mapping function will produce a codeword requiring $\log_2 (1/ p_i)$ bits. A distribution close to uniform for p_i ($p_i \approx 1/ 255$) , such as the one shown in the left plot of Figure 1.2, will result in codewords that on the average require eight bits; thus, no compression is achieved. On the other hand, for a skewed probability distribution, such as the one shown in the right plot of Figure 1.2, the **symbol-to-codeword mapping function** can on the average yield codewords requiring less than eight bits per symbol and thereby achieve compression.

We will understand these concepts better in the following Huffman encoding section.



$s_1, s_2, s_3, \ , s_N$

Figure 1.2: Typical distribution of pixel values for x_i and y_i . Here, the pixel values are shown on the horizontal axis and the corresponding probability of occurrence is shown on the vertical axis.

2 HUFFMAN ENCODING

In 1952, D. A. Huffman developed a code construction method that can be used to perform lossless compression. In Huffman coding, the modeling and the symbol-to-codeword mapping functions of Figure 1.1 are combined into a single process. As discussed earlier, **the input data are partitioned into a sequence of symbols** so as to facilitate the modeling process. In most image and video compression applications, the size of the alphabet composing these symbols is restricted to at most 64000 symbols. The Huffman code construction procedure evolves along the following parts:

- Order the symbols according to their probabilities.

  For Huffman code construction, the frequency of occurrence of each symbol must be known a priori. In practice, the frequency of occurrence can be estimated from a training set of data that is representative of the data to be compressed in a lossless manner. If, say, the alphabet is composed of N distinct symbols $s_1, s_2, s_3, , , s_N$ and the probabilities of occurrence are p_1, $p_2, p_3, , , p_N$, then the symbols are rearranged so that $p_1 \geq p_2 \geq p_3 \geq p_N$.

- Apply a contraction process to the two symbols with the smallest probabilities.

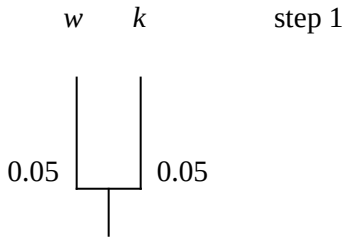
  Suppose the two symbols are s_{N-1} and s_N . We replace these two symbols by a hypothetical symbol, say, $H_{N-1} = (s_{N-1}, s_N)$ that has a probability of occurrence $p_{N-1} + p_N$. Thus, the

new set of symbols has $N-1$ members $s_1, s_2, s_3, \ , s_{N-2}, H_{N-1}$.

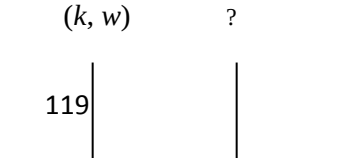
7. We repeat the previous part 2 until the final set has only one member.

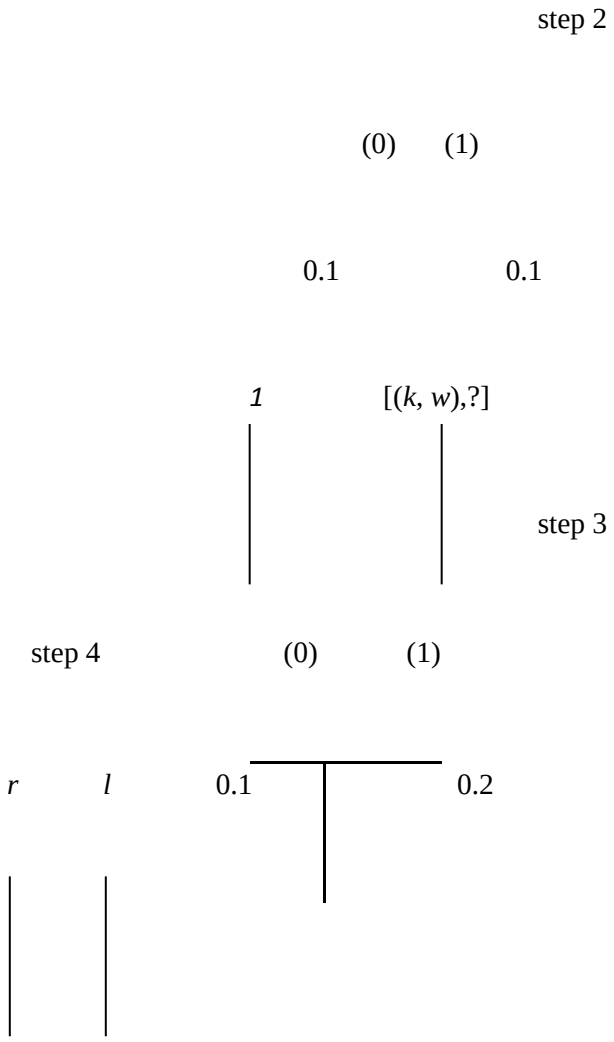
The recursive procedure in part 2 can be viewed as the construction of a binary tree, since at each step we are merging two symbols. At the end of the recursion process all the symbols will be leaf nodes of this tree. The codeword for each symbol s_i is obtained by traversing the binary tree from its root to the leaf node corresponding to s_i.

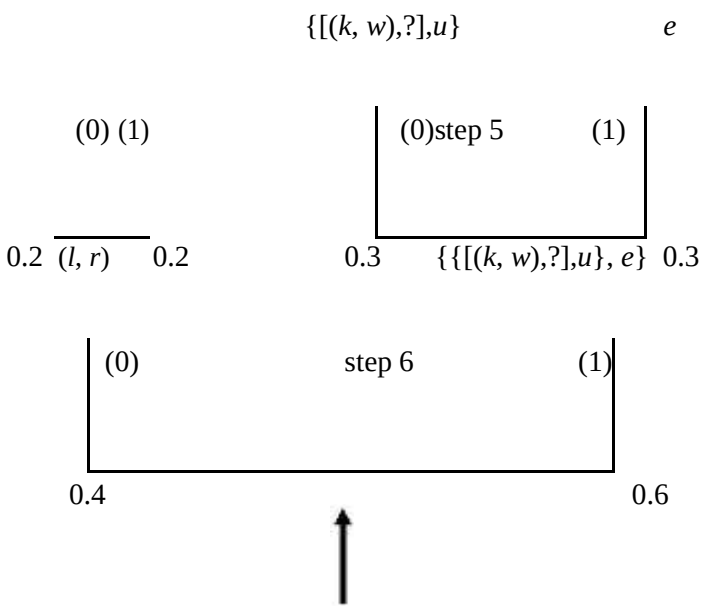
We illustrate the code construction process with the following example depicted in Figure 2.1. The input data to be compressed is composed of symbols in the alphabet *k, l, u, w, e, r,* ? . First we sort the probabilities. In Step 1, we merge the two symbols *k* and *w* to form the new symbol (*k, w*) . The probability of occurrence for the new symbol is the sum of the probabilities of occurrence for *k* and *w* . We sort the probabilities again and perform the merge on the pair of least frequently occurring symbols which are now the symbols (*k, w*) and ? . We repeat this process through Step 6. By visualizing this process as a binary tree as shown in this figure and traversing the process from the bottom of the tree to the top, one can determine the codewords for each symbol. For example, to reach the symbol *u* from the root of the tree, one traverses nodes that were assigned the bits 1,0 and 0 . Thus, the codeword for *u* is 100.



7. (1)




step 2
(0)
(1)
0.1
0.1
1
[(k, w),?]
step 3
step 4
(0)
(1)
r
l
0.1
0.2



Generate Codewords

	Step 1	**Step 2**	**Step 3**	**Step 4**	**Step 5**	**Step 6**
k 0.05	*e* 0.3	*e* 0.3	*e* 0.3	*e* 0.3	(*l*, *r*) 0.4	{{[(*k*, *w*), ?],*u*},*e*} 0.6
l 0.2	*l* 0.2	*l* 0.2	*l* 0.2	{[(*k*, *w*), ?],*u*} 0.3	*e* 0.3	(*l*, *r*) 0.4
u 0.1	*r* 0.2	*r* 0.2	*r* 0.2	*l* 0.2	{[(*k*, *w*),?],*u*} 0.3	
w 0.05	*u* 0.1	*u* 0.1	[(*k*, *w*),?] 0.2	*r* 0.2		
e 0.3	? 0.1	? 0.1	*u* 0.1			
r 0.2	*k* 0.05	(*k*, *w*) 0.1				
? 0.1	*w* 0.05					

Symbol	Probability	Codeword
k	0.05	10101
l	0.2	01
u	0.1	100
w	0.05	10100
e	0.3	11
r	0.2	00
?	0.1	1011

Figure 2.1: An example of Huffman codeword construction

In this example, the average codeword length is 2.6 bits per symbol. In general, the average codeword length is defined as

$$l_{avg} = \sum l_i \, p_i$$

(2.1) where l_i is the codeword length (in bits) for the codeword corresponding to symbol s_i . The average codeword length is a measure of the compression ratio. Since our alphabet has seven symbols, a fixed-length coder would require at least three bits per codeword. In this example, we have reduced the representation from three bits per symbol to 2.6 bits per symbol; thus, the

corresponding compression ratio can be stated as 3/2.6=1.15. For the lossless compression of typical image or video data, compression ratios in excess of two are hard to come by.

Properties of Huffman Codes

According to Shannon, the entropy of a source S is defined as

$$H(s) = n = \sum p_i \log_2 (1/ p_i)$$

(2.2) where, as before, p_i denotes the probability that symbol s_i from S will occur. From information theory, if the symbols are distinct, then the average number of bits needed to encode them is always bounded from below by their entropy. For example, for the alphabet used in the previous section, the average length is bounded by 2.6 bits per symbol. It can be shown that Huffman codewords satisfy the constraints $n \leq l_{avg} < n+1$; that is, the average length is very close to the optimum. A tighter bound is $n \leq l_{avg} < p + 0.086$, where p is the probability of the most frequently occurring symbol**. The equality is achieved when all symbol probabilities are inverse powers of two**.

The Huffman code table construction process, as was described here, is referred to as a **bottom-up** method, since we perform the contraction process on the two least frequently occurring symbols. In recent years, **top-down** construction methods have also been published in the literature.

The code construction process has a complexity of $O(N \log_2 N)$. With presorting of the input symbol probabilities, code construction methods with complexity $O(N)$ are presently known.

In the example, one can observe that no codeword is a **prefix** for another codeword. Such a code is referred to as a **prefix-condition** code. Huffman codes satisfy always the prefix-condition.

Due to the prefix-condition property, Huffman codes are **uniquely decodable**. Not every uniquely decodable code satisfies the prefix-condition. A code such as 0, 01, 011, 0111 does not satisfy the prefix-condition, since zero is a prefix for all of the codewords; however, every codeword is uniquely decodable, since a zero signifies the start of a new codeword.

If we have a binary representation for the codewords, the complement of this representation is also a valid set of Huffman codewords. The choice of using the codeword set or the corresponding complement set depends on the application. For instance, if the Huffman codewords are to be transmitted over a noisy channel where the probability of error of a one being received as a zero is higher than the probability of error of a zero being received as a one, then one would choose the codeword set for which the bit zero has a higher probability of occurrence. This will improve the performance of the Huffman coder in this noisy channel.

In Huffman coding, fixed-length input symbols are mapped into variable-length codewords. Since there are no fixed-size boundaries between codewords, if some of the bits in the compressed stream are received incorrectly or if they are not received at all due to dropouts, all the data are lost. This potential loss can be prevented by using special markers within the compressed bit stream to designate the start or end of a compressed stream packet.

Extended Huffman Codes

Suppose we have three symbols with probabilities as shown in the following table. The Huffman codeword for each symbol is also shown.

Symbol	Probability	Code
s_1	0.8	0
s_2	0.02	11
s_3	0.18	10

For the above set of symbols we have:

Entropy $H(s) = n = 0.816$ bits/symbol.

Average number of bits per symbol $l_{avg} = 1.2$ bits/symbol.

Redundancy $l_{avg} - n = 1.2 - 0.816 = 0.384$ or $\frac{l_{avg} - n}{n}$ % = 47% of entropy.

For this particular example Huffman code gives a poor compression. This is because one of the symbols (s_1) has significantly higher probability of occurrence compared to the others. Suppose

we merge the symbols in groups of two symbols. In the next table the extended alphabet and corresponding probabilities and Huffman codewords are shown.

Symbol	Probability	Code
s_1s_1	0.64	0
s_1s_2	0.016	10101
s_1s_3	0.144	11
$s_2\ s_1$	0.016	101000
$s_2\ s_2$	0.0004	10100101
$s_2\ s_3$	0.0036	1010011
s_3s_1	0.144	100

s_3s_2	0.0036	10100100
s_3s_3	0.0324	1011

Table: The extended alphabet and corresponding Huffman code

For the new extended alphabet we have

l_{avg} = 1.7516 bits/**new symbol** or l_{avg} = 0.8758 bits/original symbol.

Redundancy $l_{avg} - n$ = 0.8758– 0.816 = 0.0598 or $\frac{l_{avg} - n}{n}$ % = 7% of entropy.

We see that by coding the extended alphabet a significantly better compression is achieved. The above process is called **Extended Huffman Coding**.

Main Limitations of Huffman Coding

11. To achieve the entropy of a DMS (Discrete Memoryless SourCe), the symbol probabilities should be negative powers of 2 (i.e. log p_i is an integer).
12. Can not assign fractional codelengths.
13. Can not efficiently adapt to changing source statistics.

1. To improve coding efficiency $H(s) / l_{avg}$ we can encode the symbols of an extended source.
   However number of entries in Huffman table grows exponentially with block size.
   There are also cases where even the extended Huffman does not work. Suppose we have the following case:

Symbol	Probability	Code
s_1	0.95	0
s_2	0.02	11
s_3	0.03	10

Table: Huffman code for three symbol alphabet

Entropy $H(s) = n = 0.335$ bits/symbol.

Average number of bits per symbol $l_{avg} = 1.05$ bits/symbol.

Redundancy $l_{avg} - n = 1.05 - 0.335 = 0.715$ or $\frac{l_{avg} - n}{n}\% = 213\%$ of entropy!

Suppose we merge the symbols in groups of two symbols. In the next table the extended alphabet and corresponding probabilities and Huffman codewords are shown.

Symbol	Probability	Code
s_1s_1	0.9025	0
s_1s_2	0.019	111
s_1s_3	0.0285	100

$s_2 s_1$	0.019	1101
$s_2 s_2$	0.0004	110011
$s_2 s_3$	0.0006	110001
$s_3 s_1$	0.0285	101
$s_3 s_2$	0.0006	110010
$s_3 s_3$	0.0009	110000

Table: The extended alphabet and corresponding Huffman code

For the new extended alphabet we have

l_{avg} = 1.222 bits/**new symbol** or l_{avg} = 0.611 bits/original symbol.

Redundancy $\frac{l_{avg} - n}{n}$ % = 72% of entropy.

For this example it is proven that redundancy drops to acceptable values by merging the original symbols in groups of 8 symbols! and in that case the alphabet size is 6561 new symbols!

Arithmetic coding solves many limitations of Huffman coding. Arithmetic coding is out of the scope of this course.

3 HUFFMAN DECODING

The Huffman encoding process is relatively straightforward. The symbol-to-codeword mapping table provided by the modeler is used to generate the codewords for each input symbol. On the other hand, the Huffman decoding process is somewhat more complex.

Bit-Serial Decoding

Let us assume that the binary coding tree is also available to the decoder. In practice, this tree can be reconstructed from the symbol-to-codeword mapping table that is known to both the encoder and the decoder. The decoding process consists of the following steps:

4. Read the input compressed stream bit by bit and traverse the tree until a leaf node is reached.

5. As each bit in the input stream is used, it is discarded. When the leaf node is reached, the Huffman decoder outputs the symbol at the leaf node. This completes the decoding for this symbol.

We repeat these steps until all of the input is consumed. For the example discussed in the previous section, since the longest codeword is five bits and the shortest codeword is two bits, the decoding bit rate is not the same for all symbols. Hence, this scheme has a fixed input bit rate but a variable output symbol rate.

Lookup-Table-Based Decoding

Lookup-table-based methods yield a constant decoding symbol rate. The lookup table is constructed at the decoder from the symbol-to-codeword mapping table. If the longest codeword in this table is L bits, then a 2^L entry lookup table is needed. Recall the first example that we presented in that section where $L = 5$. Specifically, the lookup table construction for each symbol

s_i is as follows:

6. Let c_i be the codeword that corresponds to symbol s_i. Assume that c_i has l_i bits. We form an L – bit address in which the first l_i bits are c_i and the remaining $L - l_i$ bits take on all possible combinations of zero and one. Thus, for the symbol s_i there will be 2^{L-l_i} addresses.

7. At each entry we form the two-tuple (s_i, l_i).

Decoding using the lookup-table approach is relatively easy:

8. From the compressed input bit stream, we read in L bits into a buffer.

9. We use the L – bit word in the buffer as an address into the lookup table and obtain the corresponding symbol, say s_k . Let the codeword length be l_k . We have now decoded one symbol.

10. We discard the first l_k bits from the buffer and we append to the buffer, the next l_k bits from the input, so that the buffer has again L bits.
11. We repeat Steps 2 and 3 until all of the symbols have been decoded.

The primary advantages of lookup-table-based decoding are that it is fast and that the decoding rate is constant for all symbols, regardless of the corresponding codeword length. However, the input bit rate is now variable. For image or video data, the longest codeword could be around 16 to 20 bits. Thus, in some applications, the lookup table approach may be impractical due to space constraints. Variants on the basic theme of lookup-table-based decoding include using hierarchical lookup tables and combinations of lookup table and bit-by-bit decoding.

There are codeword construction methods that facilitate lookup-table-based decoding by constraining the maximum codeword length to a fixed-size L , but these are out of the scope of this course.

UNIT V

IMAGE SEGMENTATION AND REPRESENTATION

Edge detection – Thresholding - Region Based segmentation – Boundary representation: chair codes- Polygonal approximation – Boundary segments – boundary descriptors: Simple descriptors-Fourier descriptors - Regional descriptors – Simple descriptors- Texture

---

Edge detection

Edge detection is a fundamental tool in image processing and computer vision, particularly in the areas of feature detection and feature extraction, which aim at identifying points in a digital image at which the image brightness changes sharply or more formally has discontinuities.

Motivations

Canny edge detection applied to a photograph

The purpose of detecting sharp changes in image brightness is to capture important events and changes in properties of the world. It can be shown that under rather general assumptions for an image formation model, discontinuities in image brightness are likely to correspond to:

* discontinuities in depth,
* discontinuities in surface orientation,
* changes in material properties and
* variations in scene illumination.

In the ideal case, the result of applying an edge detector to an image may lead to a set of connected curves that indicate the boundaries of objects, the boundaries of surface markings as well as curves that correspond to discontinuities in surface orientation. Thus, applying an edge detection algorithm to an image may significantly reduce the amount of data to be processed and may therefore filter out information that may be regarded as less relevant, while preserving the important structural properties of an image. If the edge detection step is successful, the subsequent task of interpreting the information contents in the original image may therefore be substantially simplified. However, it is not always possible to obtain such ideal edges from real life images of moderate complexity. Edges extracted from non-trivial images are often hampered by *fragmentation*, meaning that the edge curves are not connected, missing edge segments as well as *false edges* not

corresponding to interesting phenomena in the image – thus complicating the subsequent task of interpreting the image data.

Edge detection is one of the fundamental steps in image processing, image analysis, image pattern recognition, and computer vision techniques. During recent years, however, substantial (and successful) research has also been made on computer vision methods that do not explicitly rely on edge detection as a pre-processing step.

Edge properties

The edges extracted from a two-dimensional image of a three-dimensional scene can be classified as either viewpoint dependent or viewpoint independent. A *viewpoint independent edge* typically reflects inherent properties of the three-dimensional objects, such as surface markings and surface shape. A *viewpoint dependent edge* may change as the viewpoint changes, and typically reflects the geometry of the scene, such as objects occluding one another.

A typical edge might for instance be the border between a block of red color and a block of yellow. In contrast a **line** (as can be extracted by a ridge detector) can be a small number of pixels of a different color on an otherwise unchanging background. For a line, there may therefore usually be one edge on each side of the line.

A simple edge model

Although certain literature has considered the detection of ideal step edges, the edges obtained from natural images are usually not at all ideal step edges. Instead they are normally affected by one or several of the following effects:

* focal blur caused by a finite depth-of-field and finite point spread function.
* penumbral blur caused by shadows created by light sources of non-zero radius.
* shading at a smooth object

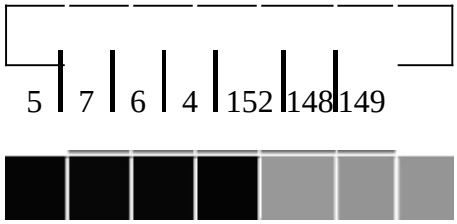
A number of researchers have used a Gaussian smoothed step edge (an error function) as the simplest extension of the ideal step edge model for modeling the effects of edge blur in practical applications.[5][3] Thus, a one-dimensional image f which has exactly one edge placed at $x = 0$ may be modeled as:

$$f(x) = \frac{I_r - I_l}{2}\left(\operatorname{erf}\left(\frac{x}{\sqrt{2}\sigma}\right) + 1\right) + I_l.$$

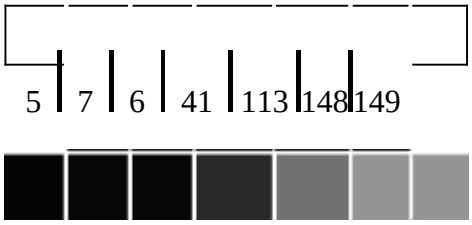
At the left side of the edge, the intensity is $I_l = \lim_{x \to -\infty} f(x)$, and right of the edge it is $I_r = \lim_{x \to \infty} f(x)$. The scale parameter σ is called the blur scale of the edge.

Why edge detection is a non-trivial task

To illustrate why edge detection is not a trivial task, let us consider the problem of detecting edges in the following one-dimensional signal. Here, we may intuitively say that there should be an edge between the 4th and 5th pixels.



If the intensity difference were smaller between the 4th and the 5th pixels and if the intensity differences between the adjacent neighboring pixels were higher, it would not be as easy to say that there should be an edge in the corresponding region. Moreover, one could argue that this case is one in which there are several edges.



Hence, to firmly state a specific threshold on how large the intensity change between two neighbouring pixels must be for us to say that there should be an edge between these pixels is not always simple. Indeed, this is one of the reasons why edge detection may be a non-trivial problem unless the objects in the scene are particularly simple and the illumination conditions can be well controlled (see for example, the edges extracted from the image with the girl above).

Approaches to edge detection

There are many methods for edge detection, but most of them can be grouped into two categories, search-based and zero-crossing based. The search-based methods detect edges by first computing a measure of edge strength, usually a first-order derivative expression such as the gradient magnitude, and then searching for local directional maxima of the gradient magnitude using a computed estimate of the local orientation of the edge, usually the gradient direction. The zero-crossing based methods search for zero crossings in a second-order derivative expression computed from the image in order to find edges, usually the zero-crossings of the Laplacian or the zero-crossings of a non-linear differential expression. As a pre-processing step to edge detection, a smoothing stage, typically Gaussian smoothing, is almost always applied (see also noise reduction).

The edge detection methods that have been published mainly differ in the types of smoothing filters that are applied and the way the measures of edge strength are computed. As many edge detection methods rely on the computation of image gradients, they also differ in the types of filters used for computing gradient estimates in the x- and y-directions.

A survey of a number of different edge detection methods can be found in (Ziou and Tabbone 1998); see also the encyclopedia articles on edge detection in Encyclopedia of Mathematics and Encyclopedia of Computer Science and Engineering.

Canny edge detection

John Canny considered the mathematical problem of deriving an optimal smoothing filter given the criteria of detection, localization and minimizing multiple responses to a single edge. He showed that the optimal filter given these assumptions is a sum of four exponential terms. He also showed that this filter can be well approximated by first-order derivatives of Gaussians. Canny also introduced the notion of non-maximum suppression, which means that given the presmoothing filters, edge points are defined as points where the gradient magnitude assumes a local maximum in the gradient direction. Looking for the zero crossing of the 2nd derivative along the gradient direction was first proposed by Haralick. It took less than two decades to find a modern geometric variational meaning for that operator that links it to the Marr-Hildreth (zero crossing of the Laplacian) edge detector. That observation was presented by Ron Kimmel and Alfred Bruckstein.

Although his work was done in the early days of computer vision, the Canny edge detector (including its variations) is still a state-of-the-art edge detector. Unless the preconditions are particularly suitable, it is hard to find an edge detector that performs significantly better than the Canny edge detector.

The Canny-Deriche detector was derived from similar mathematical criteria as the Canny edge detector, although starting from a discrete viewpoint and then leading to a set of recursive filters for image smoothing instead of exponential filters or Gaussian filters.

The differential edge detector described below can be seen as a reformulation of Canny's method from the viewpoint of differential invariants computed from a scale-space representation leading to a number of advantages in terms of both theoretical analysis and sub-pixel implementation.

Other first-order methods

For estimating image gradients from the input image or a smoothed version of it, different gradient operators can be applied. The simplest approach is to use central differences:

$$L_x(x,y) = -1/2 \cdot L(x-1,y) + 0 \cdot L(x,y) + 1/2 \cdot L(x+1,y)$$
$$L_y(x,y) = -1/2 \cdot L(x,y-1) + 0 \cdot L(x,y) + 1/2 \cdot L(x,y+1),$$

corresponding to the application of the following filter masks to the image data:

$$L_x = \begin{bmatrix} -1/2 & 0 & 1/2 \end{bmatrix} * L \quad \text{and} \quad L_y = \begin{bmatrix} +1/2 \\ 0 \\ -1/2 \end{bmatrix} * L.$$

The well-known and earlier Sobel operator is based on the following filters:

$$L_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * L \quad \text{and} \quad L_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * L.$$

Given such estimates of first- order derivatives, the gradient magnitude is then computed as:

$$|\nabla L| = \sqrt{L_x^2 + L_y^2}$$

while the gradient orientation can be estimated as

$$\theta = \operatorname{atan2}(L_y, L_x).$$

Other first-order difference operators for estimating image gradient have been proposed in the Prewitt operator and Roberts cross.

Thresholding and linking

Once we have computed a measure of edge strength (typically the gradient magnitude), the next stage is to apply a threshold, to decide whether edges are present or not at an image point. The lower the threshold, the more edges will be detected, and the result will be increasingly susceptible to noise and detecting edges of irrelevant features in the image. Conversely a high threshold may miss subtle edges, or result in fragmented edges.

If the edge thresholding is applied to just the gradient magnitude image, the resulting edges will in general be thick and some type of edge thinning post-processing is necessary. For edges detected with non-maximum suppression however, the edge curves are thin by definition and the edge pixels can be linked into edge polygon by an edge linking (edge tracking) procedure. On a discrete grid, the non-maximum suppression stage can be implemented by estimating the gradient direction using first-order derivatives, then rounding off the gradient direction to multiples of 45 degrees, and finally comparing the values of the gradient magnitude in the estimated gradient direction.

A commonly used approach to handle the problem of appropriate thresholds for thresholding is by using thresholding with hysteresis. This method uses multiple thresholds to find edges. We begin by using the upper threshold to find the start of an

edge. Once we have a start point, we then trace the path of the edge through the image pixel by pixel, marking an edge whenever we are above the lower threshold. We stop marking our edge only when the value falls below our lower threshold. This approach makes the assumption that edges are likely to be in continuous curves, and allows us to follow a faint section of an edge we have previously seen, without meaning that every noisy pixel in the image is marked down as an edge. Still, however, we have the problem of choosing appropriate thresholding parameters, and suitable thresholding values may vary over the image.

Edge Thinning

Edge thinning is a technique used to remove the unwanted spurious points on the edge of an image. This technique is employed after the image has been filtered for noise (using median, Gaussian filter etc.), the edge operator has been applied (like the ones described above) to detect the edges and after the edges have been smoothed using an appropriate threshold value. This removes all the unwanted points and if applied carefully, results in one pixel thick edge elements.

Advantages: 1) Sharp and thin edges lead to greater efficiency in object recognition. 2) If you are using Hough transforms to detect lines and ellipses then thinning could give much better results. 3) If the edge happens to be boundary of a region then, thinning could easily give the image parameters like perimeter without much algebra.

There are many popular algorithms used to do this, one such is described below:

1) Choose a type of connectivity, like 8, 6 or 4.

2) 8 connectivity is preferred, where all the immediate pixels surrounding a particular pixel are considered.

3) Remove points from North, south, east and west.

4) Do this in multiple passes, i.e. after the north pass, use the same semi processed image in the other passes and so on.

5) Remove a point if:

The point has no neighbors in the North (if you are in the north pass, and respective directions for other passes.)

The point is not the end of a line.
The point is isolated.

Removing the points will not cause to disconnect its neighbors in any way.

6) Else keep the point. The number of passes across direction should be chosen according to the level of accuracy desired.

Second-order approaches to edge detection

Some edge-detection operators are instead based upon second-order derivatives of the intensity. This essentially captures the rate of change in the intensity gradient. Thus, in the ideal continuous case, detection of zero-crossings in the second derivative captures local maxima in the gradient.

The early Marr-Hildreth operator is based on the detection of zero-crossings of the Laplacian operator applied to a Gaussian-smoothed image. It can be shown, however, that this operator will also return false edges corresponding to local minima of the gradient magnitude. Moreover, this operator will give poor localization at curved edges. Hence, this operator is today mainly of historical interest.

Differential edge detection

A more refined second-order edge detection approach which automatically detects edges with sub-pixel accuracy, uses the following *differential approach* of detecting zero-crossings of the second-order directional derivative in the gradient direction:

Following the differential geometric way of expressing the requirement of non-maximum suppression proposed by Lindeberg, let us introduce at every image point a local

coordinate system (u,v), with the v-direction parallel to the gradient direction. Assuming that the image has been presmoothed by Gaussian smoothing and a scale-space

representation $L(x,y;t)$ at scale t has been computed, we can require that the gradient magnitude of the scale-space representation, which is equal to the first-order directional derivative in the v-direction L_v, should have its first order directional derivative in the v-direction equal to zero

$$\partial_v(L_v) = 0$$

while the second-order directional derivative in the v-direction of L_v should be negative, i.e.,

$$\partial_{vv}(L_v) \leq 0.$$

Written out as an explicit expression in terms of local partial derivatives L_x, L_y ... L_{yyy}, this edge definition can be expressed as the zero-crossing curves of the differential invariant

$$L_v^2 L_{vv} = L_x^2\, L_{xx} + 2\, L_x\, L_y\, L_{xy} + L_y^2\, L_{yy} = 0,$$

that satisfy a sign-condition on the following differential invariant

$$L_v^3 L_{vvv} = L_x^3\, L_{xxx} + 3\, L_x^2\, L_y\, L_{xxy} + 3\, L_x\, L_y^2\, L_{xyy} + L_y^3\, L_{yyy} \leq 0$$

where L_x, L_y ... L_{yyy} denote partial derivatives computed from a scale-space representation L obtained by smoothing the original image with a Gaussian kernel. In this way, the edges will be automatically obtained as continuous curves with subpixel accuracy. Hysteresis thresholding can also be applied to these differential and subpixel edge segments.

In practice, first-order derivative approximations can be computed by central differences as described above, while second-order derivatives can be computed from the scale-space representation L according to:

$$\begin{aligned}
L_{xx}(x,y) &= L(x-1,y) - 2L(x,y) + L(x+1,y).\\
L_{xy}(x,y) &= (L(x-1,y-1) - L(x-1,y+1) - L(x+1,y-1) + L(x+1,y+1))/4,\\
L_{yy}(x,y) &= L(x,y-1) - 2L(x,y) + L(x,y+1).
\end{aligned}$$

corresponding to the following filter masks:

$$L_{xx} = \begin{bmatrix} 1 & -2 & 1 \end{bmatrix} * L \quad \text{and} \quad L_{xy} = \begin{bmatrix} -1/4 & 0 & 1/4 \\ 0 & 0 & 0 \\ 1/4 & 0 & -1/4 \end{bmatrix} * L \quad \text{and} \quad L_{yy} = \begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix}$$

Higher-order derivatives for the third-order sign condition can be obtained in an analogous fashion.

Phase congruency based edge detection

A recent development in edge detection techniques takes a frequency domain approach to finding edge locations. Phase congruency (also known as phase coherence) methods attempt to find locations in an image where all sinusoids in the frequency domain are in phase. These locations will generally correspond to the location of a perceived edge, regardless of whether the edge is represented by a large change in intensity in the spatial domain. A key benefit of this technique is that it responds strongly to Mach bands, and avoids false positives typically found around roof edges. A roof edge, is a discontinuity in the first order derivative of a grey-level profile]

Thresholding

Thresholding is the simplest method of image segmentation. From a grayscale image, thresholding can be used to create binary images

Method

During the thresholding process, individual pixels in an image are marked as ―object‖ pixels if their value is greater than some threshold value (assuming an object to be brighter than the background) and as ―background‖ pixels otherwise. This convention is

known as *threshold above*. Variants include *threshold below*, which is opposite of threshold above; *threshold inside*, where a pixel is labeled "object" if its value is between two thresholds; and *threshold outside*, which is the opposite of threshold inside (Shapiro, et al. 2001:83). Typically, an object pixel is given a value of ―1‖ while a background pixel is given a value of ―0.‖ Finally, a binary image is created by coloring each pixel white or black, depending on a pixel's label's.

Threshold selection

The key parameter in the thresholding process is the choice of the threshold value (or *values*, as mentioned earlier). Several different methods for choosing a threshold exist; users can manually choose a threshold value, or a thresholding algorithm can compute a value automatically, which is known as *automatic thresholding* (Shapiro, et al. 2001:83). A simple method would be to choose the mean or median value, the rationale being that if the object pixels are brighter than the background, they should also be brighter than the average. In a noiseless image with uniform background and object values, the mean or median will work well as the threshold, however, this will generally not be the case. A more sophisticated approach might be to create a histogram of the image pixel intensities

and use the valley point as the threshold. The histogram approach assumes that there is some average value for the background and object pixels, but that the actual pixel values have some variation around these average values. However, this may be computationally expensive, and image histograms may not have clearly defined valley points, often making the selection of an accurate threshold difficult. One method that is relatively simple, does not require much specific knowledge of the image, and is robust against image noise, is the following iterative method:

1. An initial threshold (T) is chosen, this can be done randomly or according to any other method desired.
2. The image is segmented into object and background pixels as described above, creating two sets:
    1. G_1 = {f(m,n):f(m,n)>T} (object pixels)
    2. G_2 = {f(m,n):f(m,n)$\leq$T} (background pixels) (note, f(m,n) is the value of the pixel located in the m^{th} column, n^{th} row)
3. The average of each set is computed.
    1. m_1 = average value of G_1
    2. m_2 = average value of G_2
4. A new threshold is created that is the average of m_1 and m_2
    1. T‘ = (m_1 + m_2)/2
5. Go back to step two, now using the new threshold computed in step four, keep repeating until the new threshold matches the one before it (i.e. until convergence has been reached).

This iterative algorithm is a special one-dimensional case of the k-means clustering algorithm, which has been proven to converge at a *local* minimum—meaning that a different initial threshold *may* give a different final result.

Adaptive thresholding

Thresholding is called **adaptive thresholding** when a different threshold is used for different regions in the image. This may also be known as *local* or *dynamic* thresholding (Shapiro, et al. 2001:89).

Categorizing thresholding Methods

Sezgin and Sankur (2004) categorize thresholding methods into the following six groups based on the information the algorithm manipulates (Sezgin et al., 2004):

- "**histogram shape**-based methods, where, for example, the peaks, valleys and curvatures of the smoothed histogram are analyzed

- **clustering**-based methods, where the gray-level samples are clustered in two parts as background and foreground (object), or alternately are modeled as a mixture of two Gaussians
- **entropy**-based methods result in algorithms that use the entropy of the foreground and background regions, the cross-entropy between the original and binarized image, etc.
- **object attribute**-based methods search a measure of similarity between the gray-level and the binarized images, such as fuzzy shape similarity, edge coincidence, etc.
- **spatial** methods [that] use higher-order probability distribution and/or correlation between pixels
- **local** methods adapt the threshold value on each pixel to the local image characteristics."

Multiband thresholding

Colour images can also be thresholded. One approach is to designate a separate threshold for each of the RGB components of the image and then combine them with an AND operation. This reflects the way the camera works and how the data is stored in the computer, but it does not correspond to the way that people recognize colour. Therefore, the HSL and HSV colour models are more often used. It is also possible to use the CMYK colour model (Pham et al., 2007).

Region growing

Region growing is a simple region-based image segmentation method. It is also classified as a pixel-based image segmentation method since it involves the selection of initial seed points.

This approach to segmentation examines neighboring pixels of initial ―seed points‖ and determines whether the pixel neighbors should be added to the region. The process is iterated on, in the same manner as general data clustering algorithms.

Region-based segmentation

The main goal of segmentation is to partition an image into regions. Some segmentation methods such as "Thresholding" achieve this goal by looking for the boundaries between regions based on discontinuities in gray levels or color properties. Region-based

segmentation is a technique for determining the region directly. The basic formulation for Region-Based Segmentation is:

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

(*b*)R_i is a connected region, i = 1, 2, ...,n

$$(c)\ R_i \bigcap R_j = \varnothing \text{ for all } i = 1, 2, ..., n.$$

(*d*)$P(R_i)$ = *TRUE* for i = 1,2,...,n.

$$(e) P(R_i \bigcup R_j) = FALSE \text{ for any adjacent region } R_i \text{ and } R_j.$$

$P(R_i)$ is a logical predicate defined over the points in set $P(R_k)$ and $\varnothing$is the null set.

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

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

(c) indicates that the regions must be disjoint.

(d) deals with the properties that must be satisfied by the pixels in a segmented region. For example $P(R_i)$ = TRUE if all pixels in R_i have the same gray level.

(e) indicates that region R_i and R_j are different in the sense of predicate P.

Basic concept of seedpoints

The first step in region growing is to select a set of seed points. Seed point selection is based on some user criterion (for example, pixels in a certain gray-level range, pixels evenly spaced on a grid, etc.). The initial region begins as the exact location of these seeds.

The regions are then grown from these seed points to adjacent points depending on a region membership criterion. The criterion could be, for example, pixel intensity, gray level texture, or color.

Since the regions are grown on the basis of the criterion, the image information itself is important. For example, if the criterion were a pixel intensity threshold value, knowledge of the histogram of the image would be of use, as one could use it to determine a suitable threshold value for the region membership criterion.

There is a very simple example followed below. Here we use 4-connected neighborhood to grow from the seed points. We can also choose 8-connected neighborhood for our pixels adjacent relationship. And the criteria we make here is the same pixel value. That is, we keep examining the adjacent pixels of seed points. If they have the same intensity value with the seed points, we classify them into the seed points. It is an iterated process until there are no change in two successive iterative stages. Of course, we can make other criteria, but the main goal is to classify the similarity of the image into regions.

Some important issues

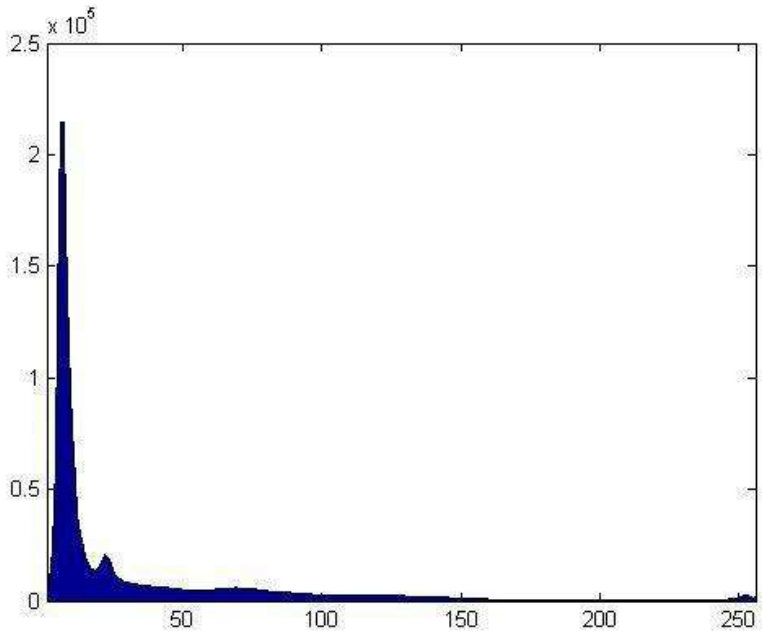
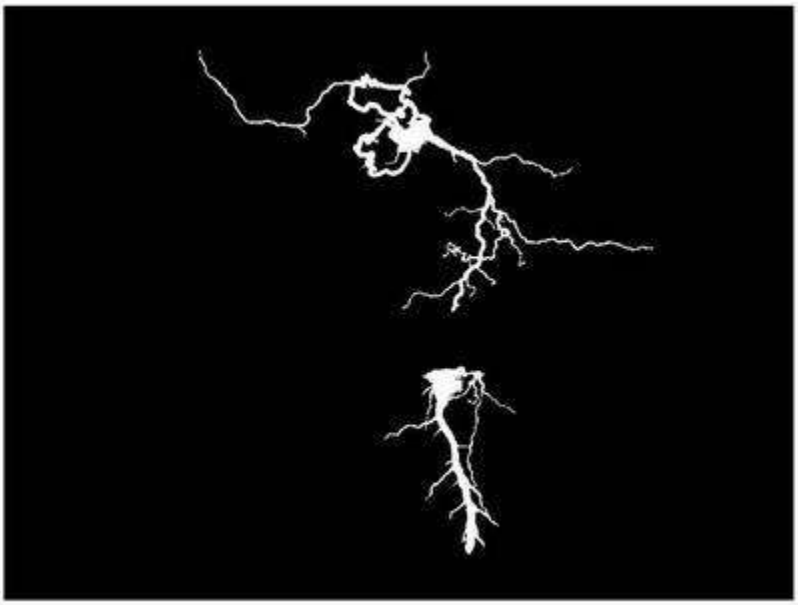


Fig. 0 The histogram of Fig. 1

Fig. 1 The Original Figure

Fig. 2

Fig. 3 Threshold：225~255



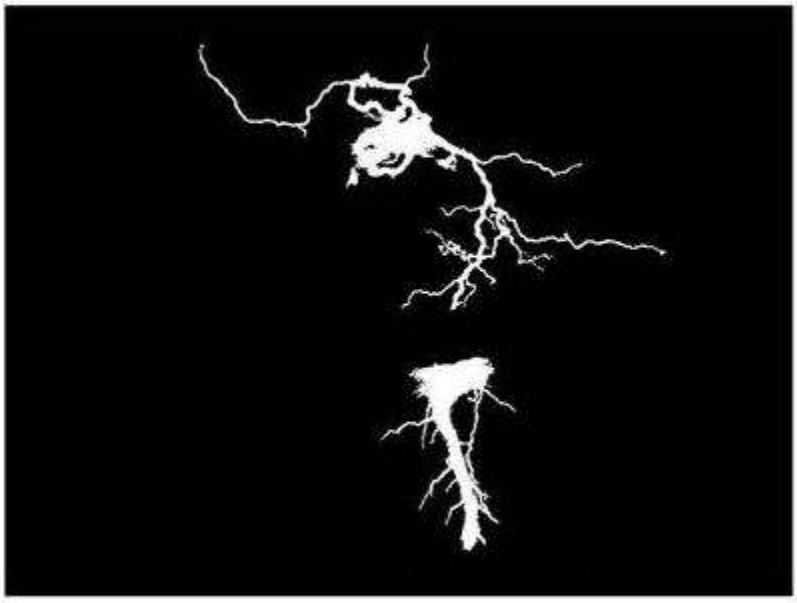


Fig. 5 Threshold：155~255

Then we can conclude several important issues about region growing：

1.The suitable selection of seed points is important.

The selection of seed points is depending on the users. For example, in a gray-level lightning image, we may want to segment the lightning from the background. Then probably, we can examine the histogram and choose the seed points from the highest range of it.

2.More information of the image is better.

Obviously, the connectivity or pixel adjacent information is helpful for us to determine the threshold and seed points.

3.The value, "minimum area threshold".

No region in region growing method result will be smaller than this threshold in the segmented image.

4.The value, "Similarity threshold value".

If the difference of pixel-value or the difference value of average gray level of a set of pixels less than ―Similarity threshold value‖, the regions will be considered as a same region.

The criteria of similarities or so called homogeneity we choose are also important. It usually depends on the original image and the segmentation result we want.

Here are some **criteria** we often use : *Gray level*(average intensity or variance), *color*, and *texture* or *shape*.

Simulation examples

Here we show a simple example for region growing.

Figure. 1 is the original image which is a gray-scale lightning image. The gray-scale value of this image is from 0 to 255. The purpose we apply region growing on this image is that we want to mark the strongest lightning part of the image and we also want the result is connected without being split apart. Therefore, we choose the points having the highest gray-scale value which is 255 as the seed points showed in the Figure. 2.

After determining the seed points, we have to determine the range of threshold. Always keeps in mind that the purpose we want to do is to mark the strongest light in the image. The third figure is the region growing result from choosing the threshold between 225 and the value of seed points (which is 255). It means we only want to mark out the points whose gray-scale values are above 225.

If we make the range of threshold wider, we will get a result having a bigger area of the lightning region show as the Figure. 3 and the Figure. 4.

We can observe the difference between the last two figures which have different threshold value showed above. Region growing provides the ability for us to separate the part we want connected.

As we can see in Figure. 3 to Figure. 5, the segmented result in this example are seed-oriented connected. That means the result grew from the same seed points are the same regions. And the points will not be grown without connected with the seed points in the beginning. Therefore, we can mention that there are still lots of points in the original image having the gray-scale value above 155 which are not marked in Figure. 5. This characteristic ensures the reliability of the segmentation and provides the ability to resist noise. For this example, this characteristic prevents us marking out the non-lightning part in the image because the lightning is always connected as one part.

The advantages and disadvantages of region growing

We briefly conclude the advantages and disadvantages of region growing.

Advantages：

1. Region growing methods can correctly separate the regions that have the same properties we define.

2. Region growing methods can provide the original images which have clear edges the good segmentation results.

3. The concept is simple. We only need a small numbers of seed point to represent the property we want, then grow the region.

4. We can determine the seed points and the criteria we want to make.

5. We can choose the multiple criteria at the same time.

6. It performs well with respect to noise.

Disadvantages :

1. The computation is consuming, no matter the time or power.
2. Noise or variation of intensity may result in holes or oversegmentation.
3. This method may not distinguish the shading of the real images.

We can conquer the noise problem easily by using some mask to filter the holes or outlier. Therefore, the problem or noise actually does not exist. In conclusion, it is obvious that the most serious problem of region growing is the power and time consuming.

Boundary Representation

- Models are a more explicit representation than CSG.
- The object is represented by a complicated data structure giving information about each of the object's faces, edges and vertices and how they are joined together.
- Appears to be a more natural representation for Vision since surface information is readily available.
- The description of the object can be into two parts:

Topology

-- records the connectivity of the faces, edges and vertices by means of pointers in the data structure.

Geometry

-- describes the exact shape and position of each of the edges, faces and vertices.

- The geometry of a vertex is just its position in space as given by its (*x*,*y*,*z*) coordinates.
- Edges may be straight lines, circular arcs, *etc.*.
- A face is represented by some description of its surface (algebraic or parametric forms used).

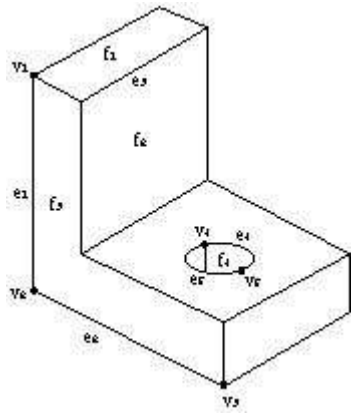


Fig. 50 Faces, edges and vertices

Chain code

A **chain code** is a lossless compression algorithm for monochrome images. The basic principle of chain codes is to separately encode each connected component, or "blot", in the image. For each such region, a point on the boundary is selected and its coordinates are transmitted. The encoder then moves along the boundary of the image and, at each step, transmits a symbol representing the direction of this movement. This continues until the encoder returns to the starting position, at which point the blot has been completely described, and encoding continues with the next blot in the image.

This encoding method is particularly effective for images consisting of a reasonable number of large connected components.

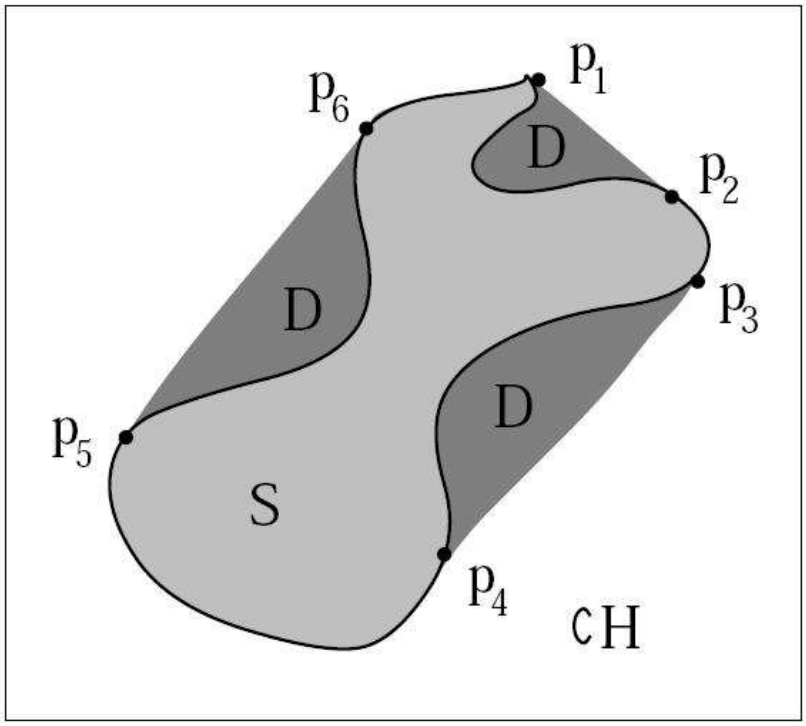
Some popular chain codes include the Freeman Chain Code of Eight Directions (FCCE), Vertex Chain Code (VCC), Three OrThogonal symbol chain code (3OT) and Directional Freeman Chain Code of Eight Directions (DFCCE).

A related blot encoding method is crack code Algorithms exist to convert between chain code, crack code, and run-length encoding.

---

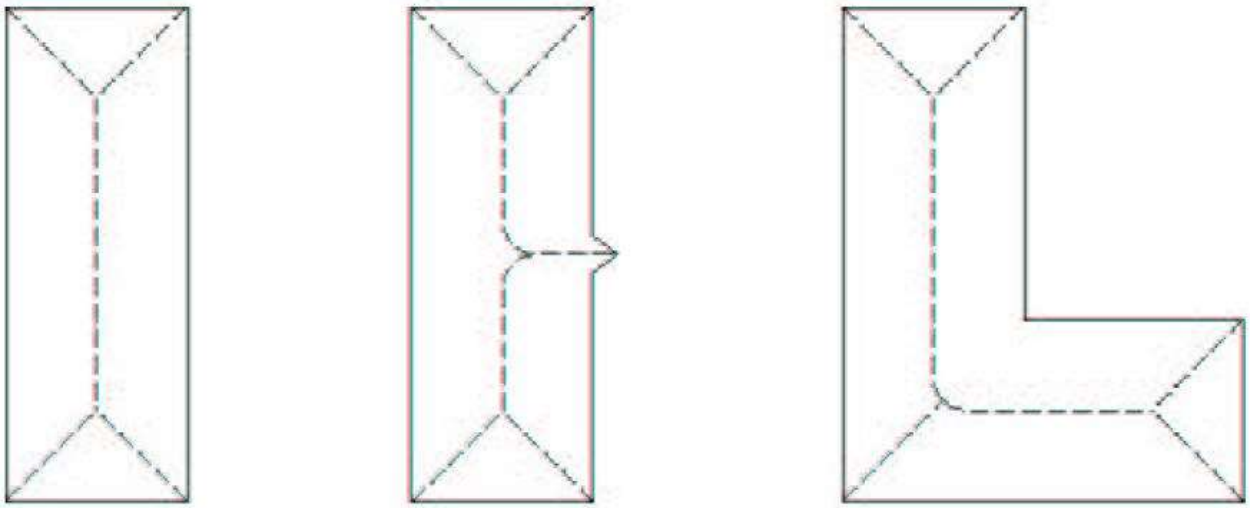
1. Let $i = 3$
2. Mark the points in the object that have furthest distance from each other with p and p_2.
3. Connect the points in the order they are numbered with lines
4. For each segment in the polygon find the point on the perimeter between the poin that have the furthest distance to the polygonal line-segment. If this distance i larger than a threshold mark the point with a label p_i
5. Renumber the points so that they are consecutive
6. Increase i
7. If no points have been added break, otherwise go to 3.

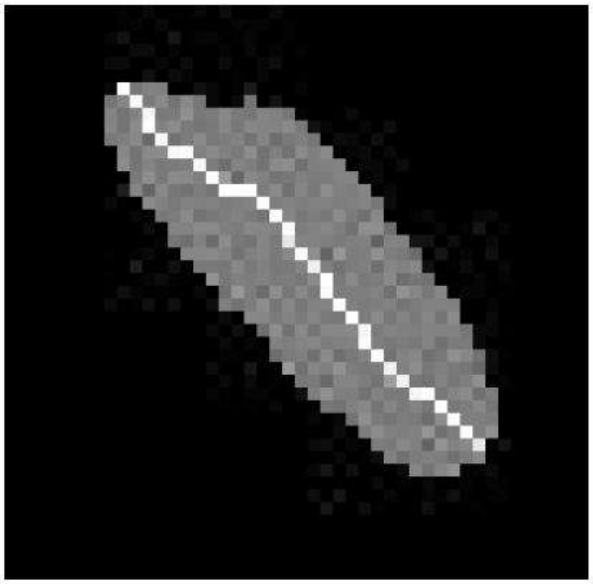
- The **convex hull** H of a set S is defined as the smallest convex set that contains S
- We define the set $D = H \setminus S$.
- The points that have neighbors from the sets D, H and $\complement H$ is called p_i. These points are used for representation of the object

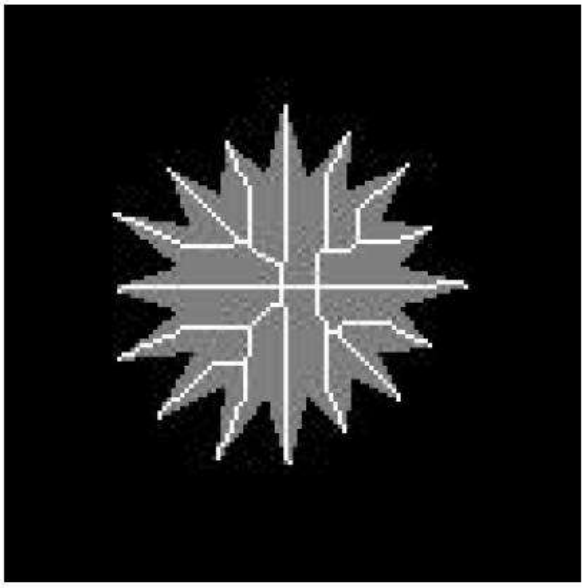


- To limit the number of points found it is possible to smooth the edge of the object

- We define an objects **skeleton** or medial-axel as the points in the object that have several nearest neighbors on the edge of the object



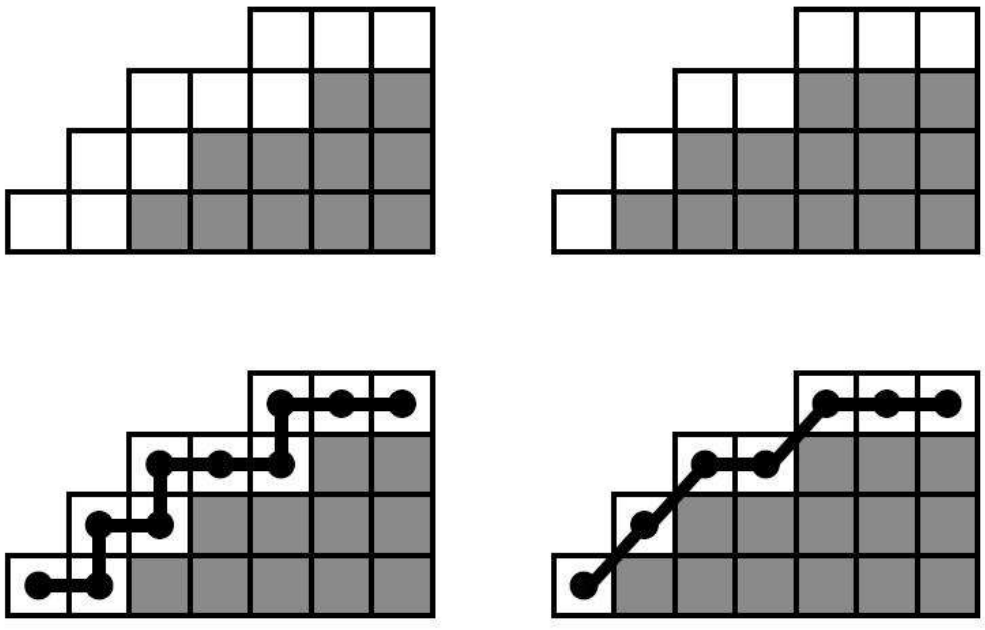




- To find the skeleton by using the definition is very costly, since it involves calculating the distance between all points in the object to all points on the perimeter
- Usually some iterative method is used that removes perimeter pixels until only the skeleton remains
- In order to achieve this, the following rules must be adhered to:
  (a) erosion of the object must be kept to a minima
  (b) connected components must not be broken
  (c) end points must not be removed
- The purpose with the **description** is to **quantify** a representation of an object.
- This implies that we instead of talking about areas in the image we can talk about their **properties**, such as length, curviness, and so on

- One of the simplest descriptions is the **length** P of the **perimeter** of an object
- The obvious measure of perimeter-length is the number of edge pixels. That is, pixel that do belong to the object, but have a neighbor that belong to the background.
- A more precise measure is to assume that each pixel-center is a corner in a polygon
- The the length of the perimeter is given by

$$P = a \cdot N_e + b \cdot N_o$$



- Intuitively we would like to set $a = 1$ and $b = \sqrt{2}$
- It is, however, possible to show that the length of the perimeter will be slightly overestimated with these values
- The optimal weights for a and b (in least square sense) will depend on the curviness of the perimeter
- If the perimeter is a straight line (!?) $a = 0.948$ and $b = 1.343$ will be optimal
- If it is assumed that the direction of two consecutive line-segments is uncorrelated $a = 0.9445$ and $b = 1.3459$ will be optimal

- The **diameter** of an object is defined as

$$Diam(B) = \max_{i,j}[D(p_i, p_j)]$$

  where D is a metric

- The line that passes through the points p_i and p_j that defines the diameter is called the main axis of the object

- The curviness of the perimeter can be obtained by calculating the angle between two consecutive line-segments of the polygonal approximation

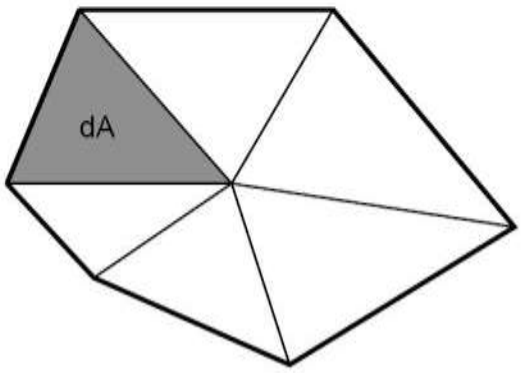
- The curvature at point p_j of the curve $\{p_i\}_{i=1}^{N}$ is also given by

$$c = ||p_{j-1} - 2p_j + p_{j+1}||^2$$

  where $p_i \in \mathbb{R}^2$

- We can approximate the area of an object with the number of pixels belonging to the object
- More accurate measures is, however, obtained by using a polygonal approximation
- The area of a polygon segment (with one corner in the origin) is given by

$$\begin{aligned} dA &= x_1 y_1 - \frac{1}{2} x_1 y_1 - \frac{1}{2} x_2 y_2 + \frac{1}{2}(x_2 - x_1)(y_1 - y_2) \\ &= \frac{1}{2}(x_1 y_2 - x_2 y_1) \end{aligned}$$



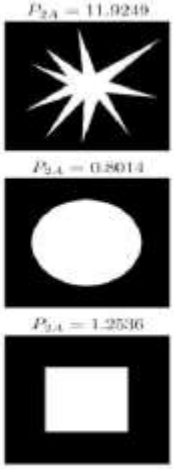
- The area of the entire polygon is then

$$A = \frac{1}{2}\sum_{i=1}^{N_b} x_i y_{i+1} - x_{i+1} y_i$$

- A circle of radius r has the area $A = \pi r^2$ and the length of the perimeter is $P = 2r\pi$ so by defining the quotient

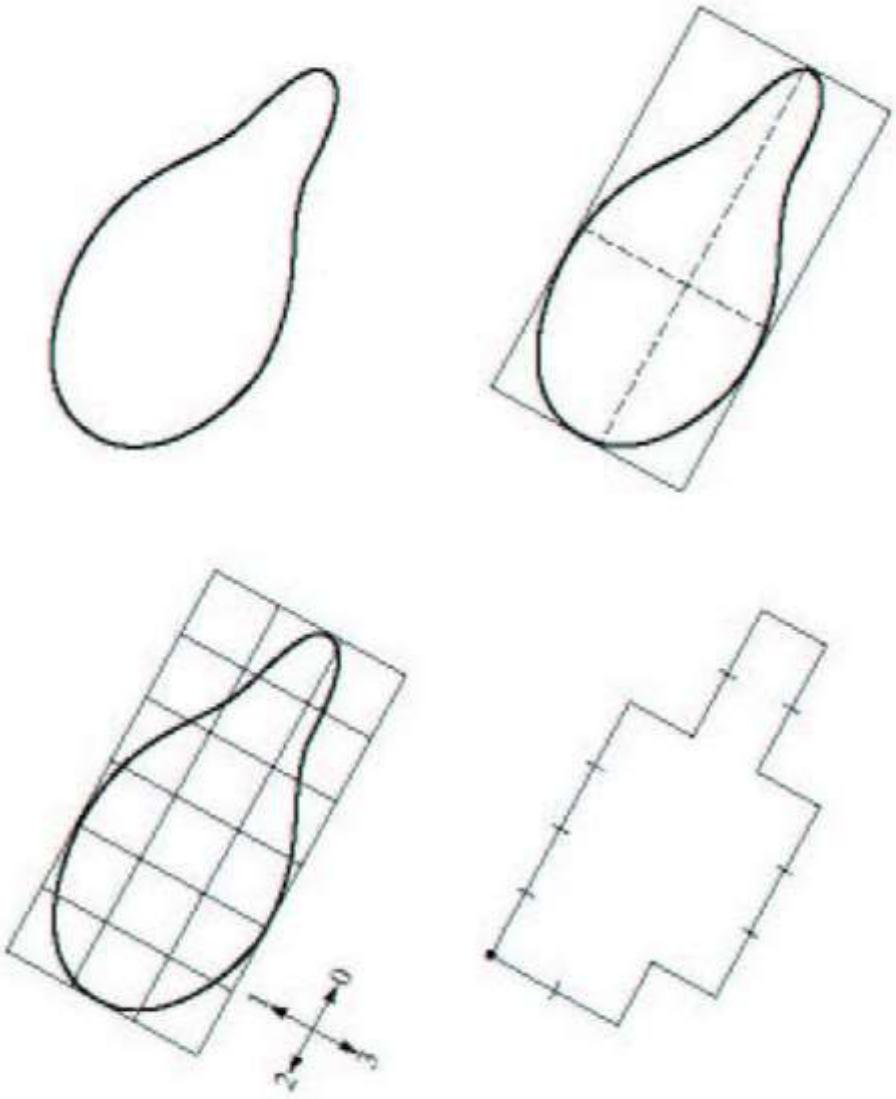
$$P_{2A} = \frac{P^2}{4\pi A}$$

we have a measurement that is 1 if the object is a circle. The larger the measurement the less circle-like is the object.



- A measure of the shape of the object can be obtained according to:
  1. Calculate the chain code for the object
  2. Calculate the difference code for the chain code
  3. Rotate the code so that it is minimal
  4. This number is called the shape number of the object
  5. The length of the number is called the order of the shape

- The quotient between the length of the main axis and the largest width orthogonal to the main axis is called the **eccentricity** of the object.
- Given a order of the shape we can find the rectangle that best approximates the eccentricity of an object
- The rectangle is rotated so that it's main axis coincide with the one of the object
- This rectangle defines a re-sampling grid
- The shape number is then calculated on the re-sampled object



Chain code: 0 0 0 0 3 0 0 3 2 2 3 2 2 2 1 2 1 1

Difference: 3 0 0 0 3 1 0 3 3 0 1 3 0 0 3 1 3 0

Shape no.: 0 0 0 3 1 0 3 3 0 1 3 0 0 3 1 3 0 3

- Suppose we have a perimeter of length N made out of coordinates (x_i, y_i).
- We then define the functions $x(k) = x_k$, $y(k) = y_k$ and

$$s(k) = x(k) + jy(k)$$

for $k = 0, 1, \ldots, N-1$.

- The discrete Fourier transform of $s(k)$ is

$$a(u) = \frac{1}{N} \sum_{k=0}^{N-1} s(k) e^{-j2\pi uk/N}$$

- The complex coefficients $a(u)$ is called the **Fourier descriptors** of the object.

- By using $M \leq N$ coefficients in the reconstruction of the object an approximation is obtained:

$$\hat{s}(k) = \sum_{u=0}^{M-1} a(u) e^{j2\pi uk/N}$$

- Observe that it is still N points in the contour but that we use M frequencies to construct the object
- The information that is lost is the one

Textures

"Texture" is an ambiguous word and in the context of texture synthesis may have one of the following meanings:

1. In common speech, "texture" used as a synonym for "surface structure". Texture has been described by five different properties in the psychology of perception: *coarseness*, *contrast*, *directionality*, *line-likeness* and *roughness* [1].

2. In 3D computer graphics, a texture is a digital image applied to the surface of a three-dimensional model by texture mapping to give the model a more realistic appearance. Often, the image is a photograph of a "real" texture, such as wood grain.

3. In image processing, every digital image composed of repeated elements is called a "texture." For example, see the images below.

Texture can be arranged along a spectrum going from stochastic to regular:

- **Stochastic textures.** Texture images of stochastic textures look like noise: colour dots that are randomly scattered over the image, barely specified by the attributes minimum and maximum brightness and average colour. Many textures look like stochastic textures when viewed from a distance. An example of a stochastic texture is roughcast.

- **Structured textures.** These textures look like somewhat regular patterns. An example of a structured texture is a stonewall or a floor tiled with paving stones.

Visual descriptors

In computer vision, **visual descriptors** or **image descriptors** are descriptions of the visual features of the contents in images, videos, algorithms, or applications that produce such descriptions. They describe elementary characteristics such as the shape, the color, the texture or the motion, among others.

Introduction

As a result of the new communication technologies and the massive use of Internet in our society, the amount of audio-visual information available in digital format is increasing considerably. Therefore, it has been necessary to design some systems that allow us to describe the content of several types of multimedia information in order to search and classify them.

The audio-visual descriptors are in charge of the contents description. These descriptors have a good knowledge of the objects and events found in a video, image or audio and they allow the quick and efficient searches of the audio-visual content.

This system can be compared to the search engines for textual contents. Although it is certain, that it is relatively easy to find text with a computer, is much more difficult to find concrete audio and video parts. For instance, imagine somebody searching a scene of

a happy person. The happiness is a feeling and it is not evident its shape, color and texture description in images.

The description of the audio-visual content is not a superficial task and it is essential for the effective use of this type of archives. The standardization system that deals with audio-visual descriptors is the MPEG-7 (*Motion Picture Expert Group - 7*).

Types of visual descriptors

Descriptors are the first step to find out the connection between pixels contained in a digital image and what humans recall after having observed an image or a group of images after some minutes.

Visual descriptors are divided in two main groups:

1. **General information descriptors**: they contain low level descriptors which give a description about color, shape, regions, textures and motion.

2. **Specific domain information descriptors**: they give information about objects and events in the scene. A concrete example would be face recognition.

General information descriptors

General information descriptors consist of a set of descriptors that covers different basic and elementary features like: color, texture, shape, motion, location and others. This description is automatically generated by means of signal processing.

- **COLOR**: the most basic quality of visual content. Five tools are defined to describe color. The three first tools represent the color distribution and the last ones describe the color relation between sequences or group of images:

    - *Dominant Color Descriptor (DCD)*
    - *Scalable Color Descriptor (SCD)*

    - *Color Structure Descriptor (CSD)*
    - *Color Layout Descriptor (CLD)*

    - *Group of frame (GoF)* or *Group-of-pictures (GoP)*

- **TEXTURE**: also, an important quality in order to describe an image. The texture descriptors characterize image textures or regions. They observe the region homogeneity and the histograms of these region borders. The set of descriptors is formed by:

    - *Homogeneous Texture Descriptor (HTD)*
    - *Texture Browsing Descriptor (TBD)*

    - *Edge Histogram Descriptor (EHD)*

- **SHAPE**: contains important semantic information due to human‘s ability to recognize objects through their shape. However, this information can only be extracted by means of a segmentation similar to the one that the human visual system implements. Nowadays, such a segmentation system is not available yet, however there exists a serial of algorithms which are considered to be a good

approximation. These descriptors describe regions, contours and shapes for 2D images and for 3D volumes. The shape descriptors are the following ones:

- *Region-based Shape Descriptor (RSD)*
- *Contour-based Shape Descriptor (CSD)*
- *3-D Shape Descriptor (3-D SD)*

- **MOTION**: defined by four different descriptors which describe motion in video sequence. Motion is related to the objects motion in the sequence and to the camera motion. This last information is provided by the capture device, whereas the rest is implemented by means of image processing. The descriptor set is the following one:
  - *Motion Activity Descriptor (MAD)*
  - *Camera Motion Descriptor (CMD)*
  - *Motion Trajectory Descriptor (MTD)*
  - *Warping and Parametric Motion Descriptor (WMD and PMD)*

- **LOCATION**: elements location in the image is used to describe elements in the spatial domain. In addition, elements can also be located in the temporal domain:
  - *Region Locator Descriptor (RLD)*
  - *Spatio Temporal Locator Descriptor (STLD)*

Specific domain information descriptors

These descriptors, which give information about objects and events in the scene, are not easily extractable, even more when the extraction is to be automatically done. Nevertheless they can be manually processed.

As mentioned before, face recognition is a concrete example of an application that tries to automatically obtain this information.

Descriptors applications

Among all applications, the most important ones are:

- Multimedia documents search engines and classifiers.
- Digital library: visual descriptors allow a very detailed and concrete search of any video or image by means of different search parameters. For instance, the search of films where a known actor appears, the search of videos containing the Everest mountain, etc.
- Personalized electronic news service.
- Possibility of an automatic connection to a TV channel broadcasting a soccer match, for example, whenever a player approaches the goal area.
- Control and filtering of concrete audio-visual contents, like violent or pornographic material. Also, authorization for some multimedia contents.