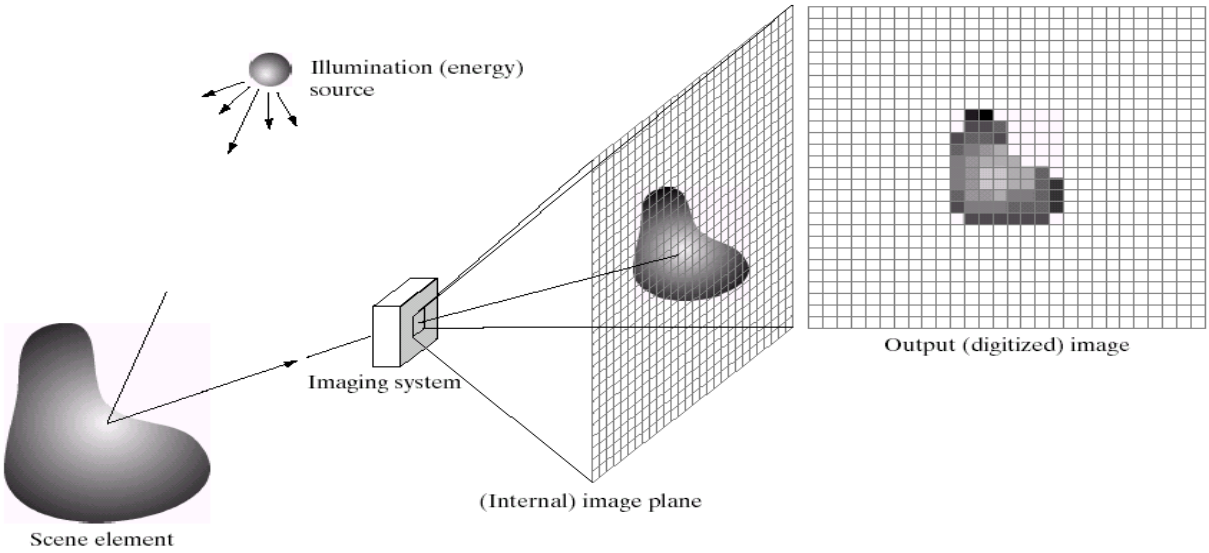
Image Processing

Table of Contents

1. Digital Image

An image may be defined as a two-dimensional function, *f (x, y),* where *x* and *y* are spatial (plane) coordinates, and the amplitude of *f* at any pair of coordinates *(x, y)* is called the *intensity* or *gray level* of the image at that point. When *x, y,* and the amplitude values of *f* are all finite, discrete quantities, we call the image a *Digital Image*. Thus, a digital image is a representation of a two-dimensional image as a finite set of digital values, called picture elements or pixels.



Pixel values typically represent gray levels, colors, height, opacities etc. Digitization implies that a digital image is an approximation of a real scene.

Common image formats include:

- 1 sample per point (B&W or Gray scale)
- 3 samples per point (Red, Green and Blue)
- 4 samples per point (Red, Green, Blue and "Alpha" a.k.a. Opacity)

1.1. Digital Image Processing

Digital image processing focused on two major tasks

- Improvements of pictorial information for human interpretation and
- Processing of image data for storage, transmission, and representation for autonomous machine perception.

Vision is the most advanced of our senses, so it is not surprising that images play the single most important role in human perception. However, unlike humans, who are limited to the visual band of the electromagnetic (EM) spectrum, imaging machines cover almost the entire EM spectrum, ranging from gamma to radio waves. They can operate on images generated by sources that humans are not accustomed to associating with images. These include ultrasound, electron microscopy, and computer-generated images. Thus, digital image processing encompasses a wide and varied field of applications.

There is no general agreement regarding where image processing stops and other related areas, such as image analysis and computer vision start. Sometimes a distinction is made by defining image processing as a discipline in which both the input and output of a process are images. This is limiting and somewhat artificial boundary. For example, under this definition, even the trivial task of computing the average intensity of an image (which yields a single number) would not be considered an image processing operation. On the other hand, there are fields such as computer vision whose ultimate goal is to use computers to emulate human vision, including learning and being able to make inferences and take actions based on visual inputs. This area itself is a branch of artificial intelligence (AI) whose objective is to emulate human intelligence. The field of AI is in its earliest stages of infancy in terms of development, with progress having been much slower that originally anticipated. They are of image analysis (also called image understanding) is in between image processing and computer vision.

There are no clear-cut boundaries in the continuum from image processing at one end to computer vision at the other. However, one useful paradigm is to consider three types of computerized processes in this continuum: low-, mid-, and high-level processes.

Low-level processes involve primitive operations such as image preprocessing to reduce noise, contrast enhancement, and image sharpening. A low-level process is characterized by the fact that both its inputs and outputs are images.

Mid-level processing on images involves tasks such as segmentation (partitioning an image into regions or objects), description of those objects to reduce them to a form suitable for computer processing, and classification (recognition) of individual objects. A mid-level process is characterized by the fact that its inputs generally are images, but its outputs are attributes extracted from those images (e.g., edges, contours, and the identity of individual objects).

Finally, higher-level processing involves "making sense" of an ensemble of recognized objects, as in image analysis, and, at the far end of the continuum, performing the cognitive functions normally associated with vision.

Low Level Process	**Mid Level Process**	**High Level Process**
Input: Image **Output:** Image **Examples:** Noise removal, image sharpening	**Input:** Image **Output:** Attributes **Examples:** Object recognition, segmentation	**Input:** Attributes **Output:** Understanding **Examples:** Scene understanding, autonomous navigation

1.1.1. Application of Digital Image Processing

- **Image Enhancement/Restoration**

One of the most common uses of DIP techniques is to improve quality, remove noise etc. Launched in 1990 the Hubble telescope can take images of very distant objects. However, an incorrect mirror made many of Hubble's images useless Image processing techniques were used to fix this.

- **Artistic Effects**

Artistic effects are used to make images more visually appealing, to add special effects and to make composite images.

- **Medical Visualization**

Find boundaries between types of tissues from slice of MRI scan. Images with gray levels represent tissue density and suitable filter is used to highlight edges.

- **Geographic Information Systems**

Digital image processing techniques are used extensively to manipulate satellite imagery, terrain classification, meteorology etc.

- **Industrial Inspection**

Human operators are expensive, slow and unreliable. Machines are made to do the job instead. Industrial vision systems are used in all kinds of industries.

Printed Circuit Board (PCB) inspection: Machine inspection is used to determine that all components are present and that all solder joints are acceptable.

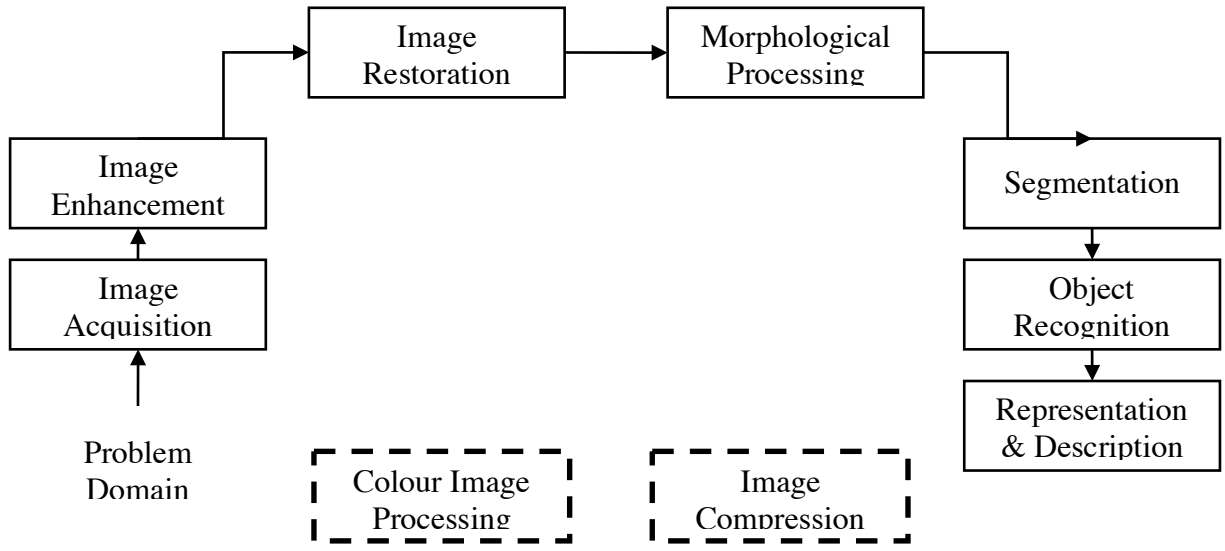
- **Law Enforcement**

Image processing techniques are used extensively by law enforcers for number plate recognition for speed cameras/ automated toll systems, fingerprint recognition, enhancement of CCTV images etc.

- **Human Computer Interfaces**

Try to make human computer interfaces more natural like face recognition, gesture recognition etc.

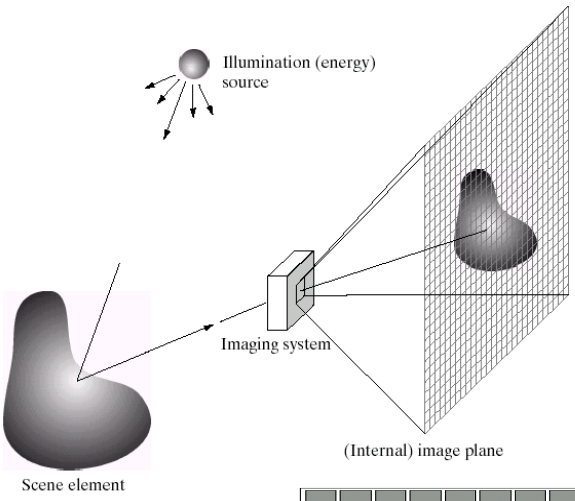
1.1.2. Key Stages in Digital Image Processing



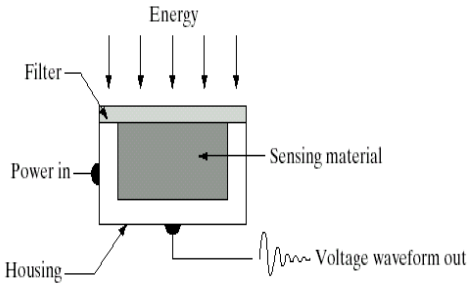
Images Acquisition

Images are typically generated by illuminating a scene and absorbing the energy reflected by the objects in that scene. Typical notions of illumination and scene can be way off:

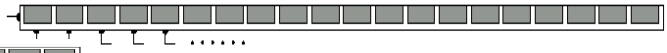
- X-rays of a skeleton
- Ultrasound of an unborn baby
- Electro-microscopic images of molecules



Images Acquisition

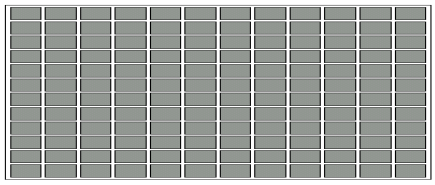


Imaging Sensor



Line of Image Sensors

Image Sensing



Array of Image Sensors

Image Sensing

Incoming energy lands on a sensor material responsive to that type of energy and this generates a voltage. Collections of sensors are arranged to capture images.

Image Enhancement

Image enhancement is among the simplest and most appealing areas of digital image processing. Basically, the idea behind enhancement techniques is to bring out detail that is obscured, or simply to highlight certain features of interest in an image. A familiar example of enhancement is when we increase the contrast of an image because "it looks better." Image enhancement is a very subjective area of image processing.

Image Restoration

Image restoration is an area that also deals with improving the appearance of an image. However, unlike enhancement, which is subjective, image restoration is objective, in the sense that restoration, techniques tend to be based on mathematical or probabilistic models of image degradation. Enhancement, on the other hand, is based on human subjective preferences regarding what constitutes a "good" enhancement result.

Color Image Processing

Color image processing is an area that has been gaining in importance because of the significance increase in the use of digital images over the Internet.

Wavelets

Wavelets are the foundation for representing images in various degrees of resolution.

Compression

Compression, as the name implies, deals with techniques for reducing the storage required to save an image, or the bandwidth required to transmit it. Although storage technology has improved significantly over the past decade, the same cannot be said for transmission capacity. This is true particularly in uses of the Internet, which are characterized by significant pictorial content. E.g. .jpg file extension used in the JPEG (Joint Photographic Experts Group) image compression standard.

Morphological Processing

Morphological processing, deals with tools for extracting image components that are useful in the representation and description of shape.

Segmentation

Segmentation procedures partition an image into its constituent parts or objects. In general, autonomous segmentation is one of the most difficult tasks in digital image processing. A rugged segmentation procedure brings the process a long way toward successful solution of imaging problems that require objects to be identified individually. On the other hand, weak or erratic segmentation algorithms almost always guarantee eventual failure. In general, the more accurate the segmentation, the more likely recognition is to succeed.

Representation and Description

Representation and description almost always follow the output of segmentation stage, which usually is raw pixel data, constituting either the boundary of a region (i.e. the set of pixels separating one image region from another) or all the points in the region itself. In either case, converting the data to a form suitable for computer processing is necessary. The first decision that must be made is whether the data should be represented as a boundary or as a complete region. Boundary representation is appropriate when the focus is on external shape characteristics, such as corners and inflections. Regional representation is appropriate when the focus is on internal properties, such as texture or skeletal shape. In some applications, these representations complement each other. Choosing a representation is only part of the solution for transforming raw data into a form suitable for subsequent computer processing. A method must also be specified for describing the data so that features of interest are highlighted. Description, also called feature selection, deals with extracting attributes that result in some quantitative information of interest or are basic for differentiating one class of objects from another.

Recognition

Recognition is the process that assigns a label (e.g. vehicle) to an object based on its descriptors.

1.1.3. Steps in image processing

The *problem domain* in this example consists of pieces of mail and the objective is to read the address on each piece

Step 1: Image Acquisition

– Acquire a digital image using an image sensor

• a monochrome or color TV camera: produces an entire image of the problem domain every 1/30 second

• a line-scan camera: produces a single image line at a time, motion past the camera produces a 2-dimensional image

– If not digital, an *analog-to-digital* conversion process is required

– The nature of the image sensor (and the produced image) are determined by the application

• Mail reading applications rely greatly on line-scan cameras

• CCD and CMOS imaging sensors are very common in many applications

Step 2: preprocessing

– Key function: improve the image in ways that increase the chance for success of the other processes

– In the mail example, may deal with contrast enhancement, removing noise, and isolating regions whose texture indicates a likelihood of alphanumeric information

Step 3: segmentation

– Broadly defined: breaking an image into its constituent parts

– In general, one of the most difficult tasks in image processing

• Good segmentation simplifies the rest of the problem

• Poor segmentation make the task impossible

– Output is usually raw pixel data: may represent region boundaries, points in the region itself, etc.

• Boundary representation can be useful when the focus is on external shape characteristics (e.g. corners, rounded edges, etc.)

• Region representation is appropriate when the focus is on internal properties (e.g. texture or skeletal shape)

– For the mail problem (character recognition) both representations can be necessary

Step 4: representation & description

– Representation: transforming raw data into a form

suitable for computer processing

– Description (also called feature extraction) deals with extracting features that result in some quantitative information of interest or features which are basic for differentiating one class of objects from another

– In terms of character recognition, *descriptors* such as lakes (holes) and bays help differentiate one part of the alphabet from another

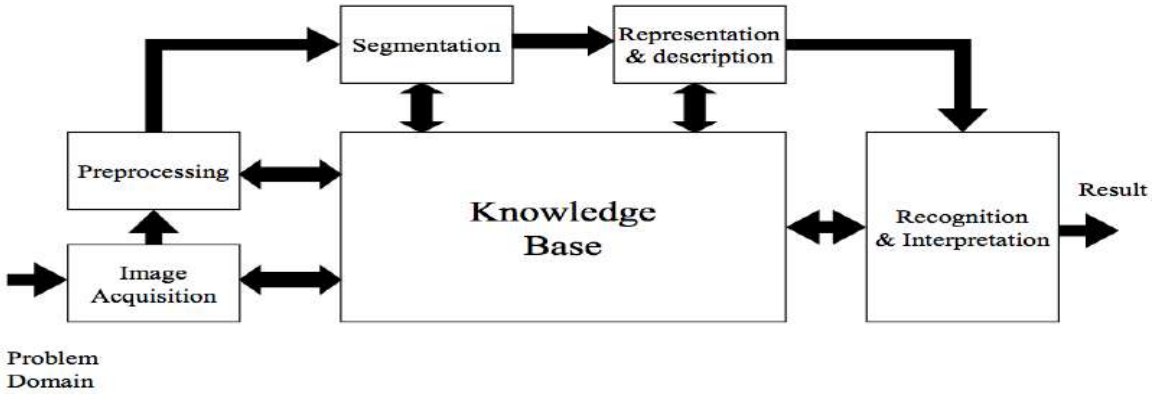
Step 5: recognition & interpretation

– Recognition: The process which assigns a label to an object based on the information provided by its descriptors

A may be the alphanumeric character A

– Interpretation: Assigning meaning to an ensemble of recognized objects

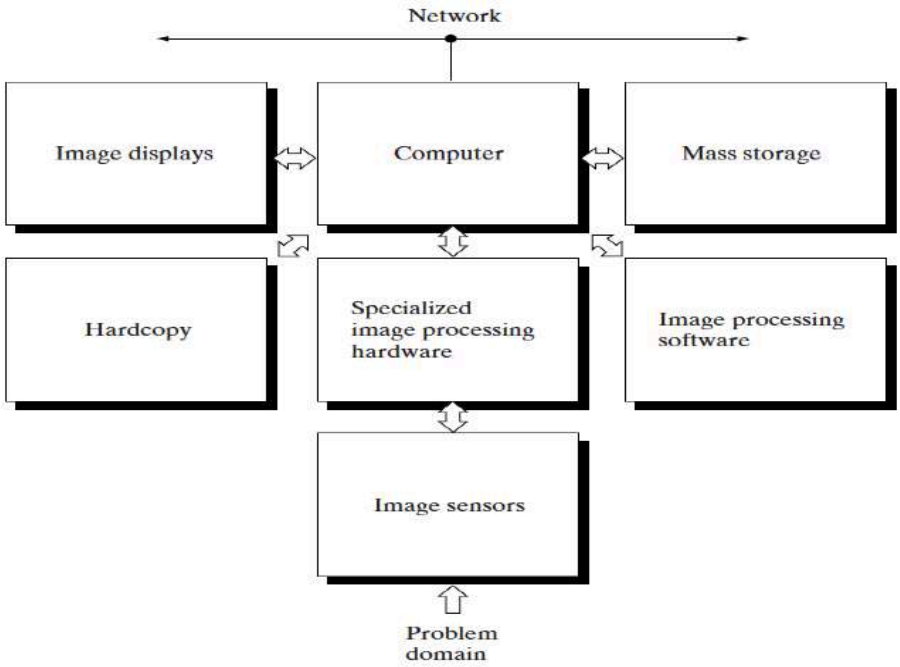
35487-0286 may be a ZIP code



Knowledge about a problem domain is coded into an image processing system in the form of a *knowledge database*

- May be simple: detailing areas of an image expected to be of interest
- May be complex: A list of all possible defects of a material in a vision inspection system
- Guides operation of each processing module
- Controls interaction between modules: Provides feedback through the system

1.2. Components of Image Processing



Components of General Purpose Image Processing System

With reference to *sensing*, two elements are required to acquire digital images. The first is a physical device that is sensitive to the energy radiated by the object we wish to image. The second, called a *digitizer*, is a device for converting the output of the physical sensing device into digital form. For instance, in a digital video camera, the sensors produce an electrical output proportional to light intensity. The digitizer converts these outputs to digital data.

Specialized image processing hardware usually consists of the digitizer just mentioned, plus hardware that performs other primitive operations, such as an arithmetic logic unit (ALU), which performs arithmetic and logical operations in parallel on entire images. One example of how an ALU is used is in averaging images as quickly as they are digitized, for the purpose of noise reduction. This type of hardware sometimes is called a front-end subsystem, and its most distinguishing characteristic is speed. In other words, this unit performs functions that require fast data throughputs (e.g., digitizing and averaging video images at 30 frames/s) that the typical main computer cannot handle.

The *computer* in an image processing system is a general-purpose computer and can range from a PC to a supercomputer. In dedicated applications, sometimes specially designed computers are used to achieve a required level of performance, but our interest here is on general-purpose image processing systems. In these systems, almost any well-equipped PC-type machine is suitable for offline image processing tasks.

Software for image processing consists of specialized modules that perform specific tasks. A well-designed package also includes the capability for the user to write code that, as a minimum, utilizes the specialized modules. More sophisticated software packages allow the integration of those modules and general- purpose software commands from at least one computer language.

Mass storage capability is a must in image processing applications. An image of size 1024*1024 pixels, in which the intensity of each pixel is an 8-bit quantity, requires one megabyte of storage space if the image is not compressed. When dealing with thousands, or even millions, of images, providing adequate storage in an image processing system can be a challenge. Digital storage for image processing applications falls into three principal categories: (1) shortterm storage for use during processing, (2) on-line storage for relatively fast recall, and (3) archival storage, characterized by infrequent access. Storage is measured in bytes (eight bits), Kbytes (one thousand bytes), Mbytes (one million bytes), Gbytes (meaning giga, or one billion, bytes), and Tbytes (meaning tera, or one trillion, bytes).

Image displays in use today are mainly color (preferably flat screen) TV monitors. Monitors are driven by the outputs of image and graphics display cards that are an integral part of the computer system. Seldom are there requirements for image display applications that cannot be met by display cards available commercially as part of the computer system. In some cases, it is necessary to have stereo displays, and these are implemented in the form of headgear containing two small displays embedded in goggles worn by the user.

Hardcopy devices for recording images include laser printers, film cameras, heat-sensitive devices, inkjet units, and digital units, such as optical and CD-ROM disks. Film provides the highest possible resolution, but paper is the obvious medium of choice for written material.For presentations, images are displayed on film transparencies or in a digital medium if image projection equipment is used.The latter approach is gaining acceptance as the standard for image presentations.

Networking is almost a default function in any computer system in use today. Because of the large amount of data inherent in image processing applications, the key consideration in image transmission is bandwidth. In dedicated networks, this typically is not a problem, but communications with remote sites via the Internet are not always as efficient. Fortunately, this situation is improving quickly as a result of optical fiber and other broadband technologies.

2. Digital Image Fundamental

2.1. A Simple Image Formation

An image is a 2-D light intensity function *f(x,y)*. When an image is generated from a physical process, its values are proportional to energy radiated by a physical source (e.g. electromagnetic waves). As light is a form of energy, f(x, y) must be non-zero and finite; that is, $0 < f(x,y) < \infty$

The function f(x,y) may be characterized by two components:

- The amount of source illumination on the scene being viewed, and
- The amount of illumination reflected by the objects in the scene.

These are called illumination and reflectance components and are denoted by *i(x,y)* and

r(x,y) respectively. The two functions combine as a product to for *f(x,y)*:

$$f(x,y)=i(x,y)r(x,y)$$

i(x,y) is the illumination: $0 < i(x,y) < \infty$

- Typical values: 9000 foot-candles sunny day, 100 office room, 0.01 moonlight

r(x,y) is the reflectance: $0 < r(x,y) < 1$

- r(x,y)=0 implies total absorption
- r(x,y)=1 implies total reflectance
- Typical values: 0.01 black velvet, 0.80 flat white paint, 0.93 snow

The nature of *i(x,y)* is determined by the illumination source, and *r(x,y)* is determined by the characteristics of the imaged objects. The intensity of a monochrome image *f* at (*x,y*) is the *gray level* (*l*) of the image at that point

$$L_{min} \leq l \leq L_{max}$$

- In practice $L_{min} = i_{min}\, r_{min}$ and $L_{max} = i_{max}\, r_{max}$
- As a guideline $L_{min} \approx 0.005$ and $L_{max} \approx 100$ for indoor image processing applications
- The interval $[L_{min}, L_{max}]$ is called the *gray scale*
- Common practice is to shift the interval to [0,L] where *l*=0 is considered black and *l*=L is considered white. All intermediate values are shades of gray.

2.2. Sampling and Quantization

To be suitable for computer processing an image, f(x,y) must be digitized both spatially and in amplitude

- Digitizing the spatial coordinates is called *image sampling*
- Amplitude digitization is called gray-level quantization

The result of sampling and quantization is a matrix of real numbers. Assume an image *f(x,y)* is sampled so that the resulting digital image has M rows and N columns. f(x,y) is then approximated by equally spaced samples in the form of an NxM array where each element is a discrete quantity

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & \dots & f(0,M-1) \\ f(1,0) & f(1,1) & \dots & f(1,M-1) \\ f(N-1,0) & f(N-1,1) & \dots & f(N-1,M-1) \end{bmatrix}$$

Common practice is to let N and M be powers of two; $N = 2^n$ and $M = 2^k$. And $G = 2^m$ where G denotes the number of gray levels. The assumption here is that gray levels are equally space in the interval [0,L]. The number of bits, b necessary to store the image is then

$$b = N * M * m$$

$$b = N^2 m; if\ N = M$$

For example, a 128*128 image with 64 gray levels would require 98,304 bits of storage.

2.2.1. Spatial and Gray-Level Resolution

Sampling is the principal factor determining the spatial resolution of an image. Basically, spatial resolution is the smallest discernible detail in an image. Suppose that we construct chart with vertical lines of width W, with the space between the lines also having width W.

A line pair consists of one such line and its adjacent space. Thus, the width of a line pair is 2W, and there are 1/2W line pairs per unit distance. A widely used definition of resolution is simply *the smallest number of discernible line pairs per unit distance*; for example, 100 line pairs per millimeter.

Gray level resolution similarly refers to the smallest discernible change in gray level.

2.2.2. Aliasing and Moiré Patterns

Functions whose area under the curve is finite can be represented in terms of Sines and Cosines of various frequencies. The Sine/Cosine component with the highest frequency determines the highest "frequency content" of the function. Suppose that this highest frequency is finite and that the function is of unlimited duration (these functions are called band-limited functions). Then, the Shannon sampling theorem [Bracewell (1995)] tells us that, if the function is sampled at a rate equal to or greater than twice its highest frequency, it is possible to recover completely the original function from its samples. If the function is under-sampled, then a phenomenon called aliasing corrupts the sampled image. The corruption is in the form of additional frequency components being introduced into the sampled function. These are called aliased frequencies. Note that the sampling rate in images is the number of samples taken (in both spatial directions) per unit distance.

Except for a special case it is impossible to satisfy the sampling theorem in practice. We can only work with sampled data that are finite in duration. We can model the process of converting a function of unlimited duration into a function of finite duration simply by multiplying the unlimited function by a "gating function" that is valued 1 for some interval and 0 elsewhere. Unfortunately, this function itself has frequency components that extend to infinity. Thus, the very act of limiting the duration of a band-limited function causes it to cease being band limited, which causes it to violate the key condition of the sampling theorem. The principal approach for reducing the aliasing effects on an image is to reduce its high-frequency components by blurring the image prior to sampling. However, aliasing is always present in a sampled image. The effect of aliased frequencies can be seen under the right conditions in the form of so-called *Moiré patterns*.

There is one special case of significant importance in which a function of infinite duration can be sampled over a finite interval without violating the sampling theorem. When a function is periodic, it may be sampled at a rate equal to or exceeding twice its highest frequency, and it is possible to recover the function from its samples provided that the sampling captures exactly an integer number of periods of the function. This special case allows us to illustrate vividly the Moiré effect. Figure shows two identical periodic patterns of equally spaced vertical bars, rotated in opposite directions and then superimposed on each other by multiplying the two images. A Moiré pattern, caused by a breakup of the periodicity, is seen in Fig. as a 2-D sinusoidal (aliased) waveform (which looks like a corrugated tin roof) running in a vertical direction. A similar pattern can appear when images are digitized (e.g., scanned) from a printed page, which consists of periodic ink dots.

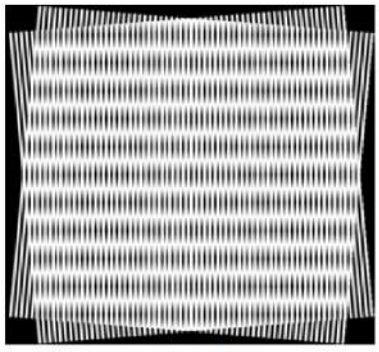


Illustration of the Moiré pattern effect.

2.2.3. Zooming and Shrinking Digital Images

This topic is related to image sampling and quantization because zooming may be viewed

as oversampling, while shrinking may be viewed as under-sampling. The key difference between these two operations and sampling and quantizing an original continuous image is that zooming and shrinking are applied to a digital image.

Zooming requires two steps: the creation of new pixel locations, and the assignment of gray levels to those new locations. Let us start with a simple example. Suppose that we have an image of size 500*500 pixels and we want to enlarge it 1.5 times to 750*750 pixels. Conceptually, one of the easiest ways to visualize zooming is laying an imaginary 750*750 grid over the original image. Obviously, the spacing in the grid would be less than one pixel because we are fitting it over a smaller image. In order to perform gray-level assignment for any point in the overlay, we look for the closest pixel in the original image and assign its gray level to the new pixel in the grid. When we are done with all points in the overlay grid, we simply expand it to the original specified size to obtain the zoomed image. This method of gray-level assignment is called nearest neighbor interpolation.

Pixel replication is a special case of nearest neighbor interpolation. Pixel replication is applicable when we want to increase the size of an image an integer number of times. For instance, to double the size of an image, we can duplicate each column. This doubles the image size in the horizontal direction. Then, we duplicate each row of the enlarged image to double the size in the vertical direction. The same procedure is used to enlarge the image by any integer number of times (triple, quadruple, and so on). Duplication is just done the required number of times to achieve the desired size. The gray-level assignment of each pixel is predetermined by the fact that new locations are exact duplicates of old locations.

Image shrinking is done in a similar manner as just described for zooming. The equivalent process of pixel replication is row-column deletion. For example, to shrink an image by one-half, we delete every other row and column. We can use the zooming grid analogy to visualize the concept of shrinking by a non-integer factor, except that we now expand the grid to fit over the original image, do gray-level nearest neighbor or bilinear interpolation, and then shrink the grid back to its original specified size. To reduce possible aliasing effects, it is a good idea to blur an image slightly before shrinking it.

2.3. Basic Relationships Between Pixels

2.3.1. Neighbors of a Pixel

An image is denoted by: f(x,y). When referring, lowercase letters (e.g. *p, q*) will denote individual pixels. A subset of f(x,y) is denoted by *S*

- Neighbors of a pixel:

– A pixel *p* at (x,y) has 4 horizontal/vertical neighbors at
  - (x+1,y),(x-1,y),(x,y+1)and(x,y-1)
  - called the *4-neighbors* of *p*: $N_4(p)$

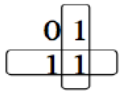
– A pixel *p* at (x,y) has 4 diagonal neighbors at
  - (x+1,y+1),(x+1,y-1),(x-1,y+1)and(x-1,y-1)
  - called the *diagonal–neighbors* of *p*:$N_D(p)$

– The *4-neighbors* and the *diagonal-neighbors* of *p* are called the *8-neighbors* of *p*: $N_8(p)$

2.3.2. Adjacency, Connectivity, Regions, and Boundaries

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

Let V be the set of values used to determine connectivity



– For example, in a binary image, V={1} for the connectivity of pixels with a value of 1

– In a gray scale image, for the connectivity of pixels with a range of intensity values of, say, 32 to 64, it follows that V={32,33,...,63,64}

We consider three types of connectivity

• *4-connectivity*: Pixels p and q with values from V are *4-connected* if q is in the set $N_4(p)$

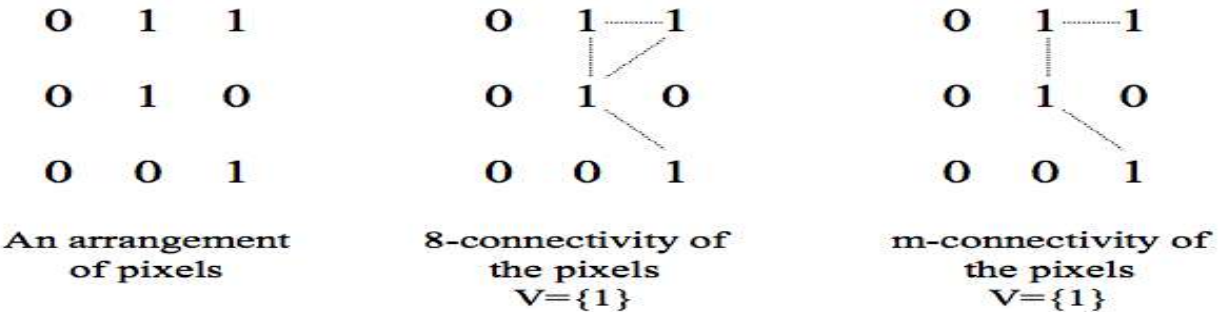
• *8-connectivity*: Pixels p and q with values from V are *8-connected* if q is in the set $N_8(p)$

• *m-connectivity (mixed)*: Pixels p and q with values from V are

m-connected if

– q is in the set $N_4(p)$,*or*

– q is in the set $N_D(p)$ *and* the set $N_4(p) \cap N_4(q)$ is empty (This is the set of pixels that are 4-neighbors of p and q and whose values are from V)



Pixel p is adjacent to q if they are connected and we can define 4-, 8-, or m-adjacency depending on the specified type of connectivity

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

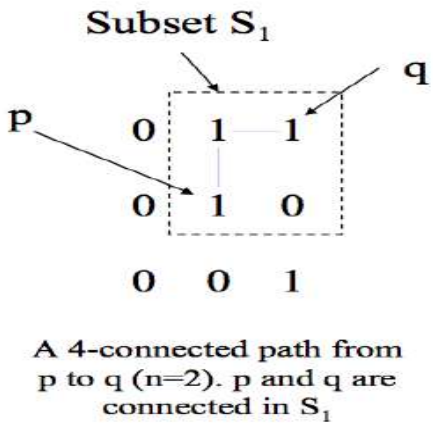
• A path from p at (x,y) to q at (s,t) is a sequence of distinct pixels with coordinates (x_0,y_0), (x_1,y_1),....., (x_n,y_n)

– Where (x_0,y_0)=(x,y) and (x_n,y_n)=(s,t) and

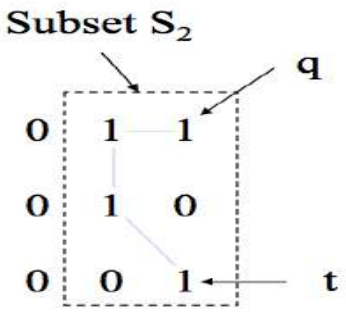
– (x_i,y_i) is adjacent to (x_{i-1},y_{i-1}) for 1<= i <= n

– n is the *length* of the path

• If p and q are in S, then p is connected to q in S if there is a path from p to q consisting entirely of pixels in S

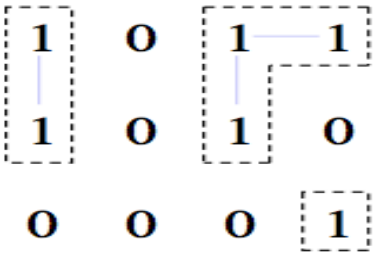


A 4-connected path from p to q (n=2). p and q are connected in S_1

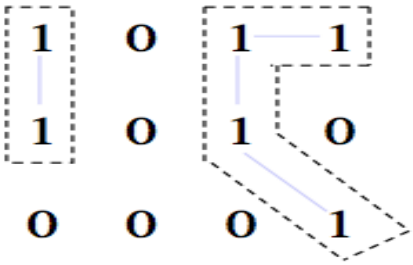


An m-connected path from t to q (n=3). t and q are connected in S_2

For any pixel *p* in *S*, the set of pixels connected to *p* form a connected component of *S*. Distinct connected components in *S* are said to be *disjoint.*



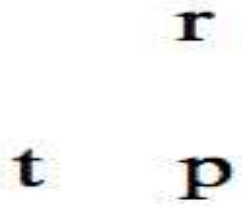
3 4-connected components of S



2 m-connected components of S

2.3.3. Labeling 4-Connected Components

Consider scanning an image pixel by pixel from left to right and top to bottom



– Assume, for the moment, we are interested in 4-connected components. Let *p* denote the pixel of interest, and *r* and *t* denote the upper and left neighbors of *p*, respectively. The nature of the scanning process assures that *r* and *t* have been encountered (and labeled if 1) by the time *p* is encountered. Consider the following procedure

if p=0 continue to the next position

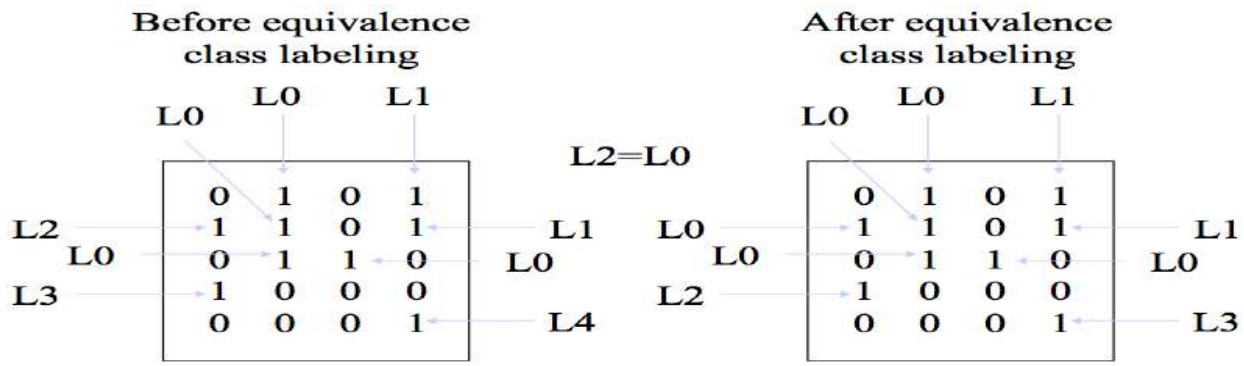
if r=t=0 assign a new label to p (Ln)

if r=t=1 and they have the same label, assign that label to p

if only one of r and t are 1, assign its label to p

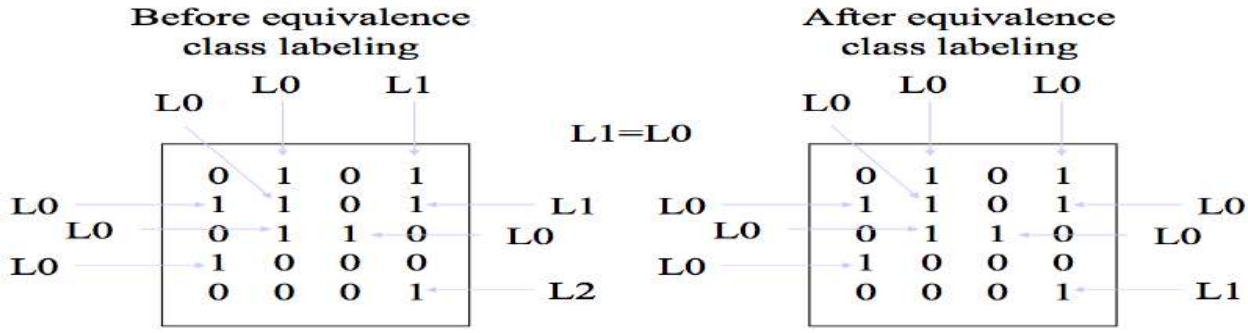
if r=t=1 and they have different labels, assign one label to p and note that the two labels are equivalent (that is r and t are connected through p)

At the end of the scan, sort pairs of equivalent labels into equivalence classes and assign a different label to each class



2.3.4. Labeling 8-Connected Components

Proceed as in the 4-connected component labeling case, but also examine two upper diagonal neighbors (*q* and *s*) of *p*



2.3.5. Distance Measures

Given pixels *p*, *q*, and *z* at (*x,y*), (*s,t*) and (*u,v*) respectively. *D* is a *distance function* (or *metric*) if:

– D(p,q) ≥ 0 (D(p,q)=0 iff p=q),

– D(p,q) = D(q,p), and

– D(p,z) ≤ D(p,q) + D(q,z).

The *Euclidean distance* between *p* and *q* is given by:

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

The pixels having distance less than or equal to some value *r* from (x,y) are the points contained in a disk of radius *r* centered at (x,y). The D4 distance (also called the *city block distance*) between *p* and *q* is given by:

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

In this case, the pixels having a D4 distance less than some *r* from (x,y) form a diamond centered at (x,y). For example: pixels where D4 ≤ 2

```
    2
  2 1 2
2 1 0 1 2
  2 1 2
    2
```

Note: Pixels with D_4=1 are the 4-neighbors of (x,y)

```
2 2 2 2 2
2 1 1 1 2
2 1 0 1 2
2 1 1 1 2
2 2 2 2 2
```

Note: Pixels with D_8=1 are the 8-neighbors of (x,y)

The D8 distance (also called the *chessboard distance*) between *p* and *q* is given by:

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

The pixels having a D8 distance less than some *r* from (x,y) form a square centered at (x,y). For example: pixels where $D_8 \leq 2$ (above).

Distance measure and connectivity

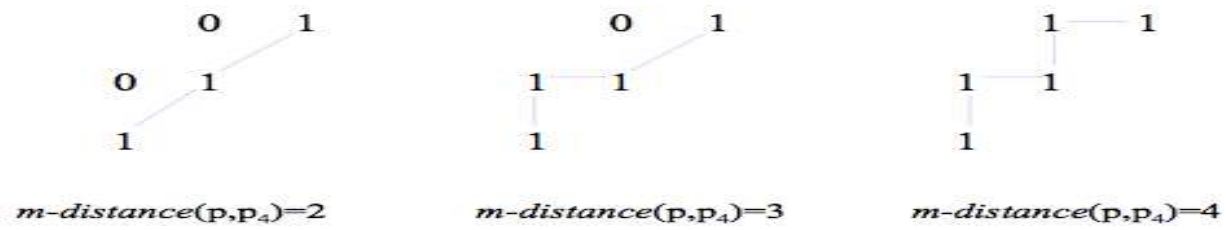
- The D4 distance between two points *p* and *q* is the shortest 4-path between the two points
- The D8 distance between two points *p* and *q* is the shortest 8-path between the two points
- D4 and D8 may be considered, regardless of whether a connected path exists between them, because the definition of these distances involves only the pixel coordinates
- For *m-connectivity*, the value of the distance (the length of the path) between two points depends on the values of the pixels along the path

Consider the given arrangements of pixels and assume

p_3 p_4
p_1 p_2
p

- p, p_2 and p_4 =1
- p_1 and p_3 can be 0 or 1

If *V*={1} and p_1 and p_3 are 0, the m-distance (p, p_4) is 2 If either p_1 or p3 are $_1$, the m-distance (p, p_4) is 3 If p_1 and p_3 are 1, the m-distance (p, p_4) is 4



2.3.6. Arithmetic and Logic Operations

Arithmetic & logic operations on images used extensively in most image processing applications. This may cover the entire image or a subset.

Arithmetic operation between pixels *p* and *q* are defined as:

– Addition: (*p*+*q*); Used often for image averaging to reduce noise

– Subtraction: (*p*-*q*); Used often for static background removal

– Multiplication: (*p***q*) (or *pq*, $p \times q$); Used to correct gray-level shading

– Division: ($p \div q$) (or *p*/*q*); As in multiplication

Logical operation between pixels *p* and *q* are defined as:

– AND: *p* AND *q* (also p·q); OR: *p* OR *q* (also p+q); COMPLEMENT: NOT*q* (also *q*′)

- Form a *functionally complete* set
- Applicable to binary images

- Basic tools in binary image processing, used for:
  - Masking , Feature detection and Shape analysis

2.3.7. Neighborhood-Oriented Operations

Arithmetic and logical operations may take place on a subset of the image. It is typically neighborhood oriented and formulated in the context of *mask* operations (also called *template*, *window* or *filter* operations).

Basic concept:

- let the value of a pixel be a function of its (current) gray level and the gray level of its neighbors (in some sense)
- Consider the following subset of pixels in an image
- Suppose we want to filter the image by replacing the value at Z_5 with the average value of the pixels in a 3x3 region centered around Z_5
- Perform an operation of the form: $Z = \frac{1}{9}(Z_1 + Z_2 + \cdots + Z_9) = \frac{1}{9}\sum_{i=1}^{9} Z_i$
- And assign to Z_5 the value of z

	Z_1	Z_2	Z_3
	Z_4	Z_5	Z_6
	Z_7	Z_8	Z_9

In more general form, the operation may look like:

$$Z = (W_1Z_1 + W_2Z_2 + \cdots + W_9Z_9) = \sum_{i=1}^{9} W_iZ_i$$

This equation is widely used in image processing

	W_1	W_2	W_3
	W_4	W_5	W_6
	W_7	W_8	W_9

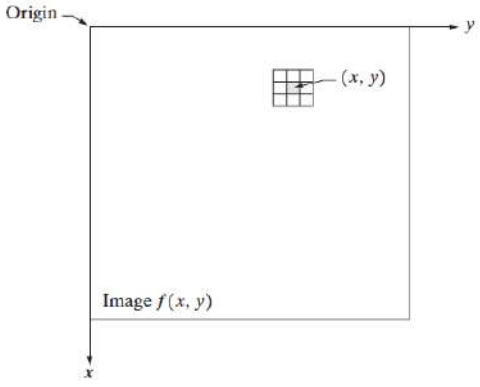
- Proper selection of coefficients (weights) allows for operations such as
  - noise reduction
  - region thinning
  - edge detection

3. Image Enhancement in Spatial Domain

The principal objective of enhancement is to process an image so that the result is more suitable than the original image for a specific application. There is no general theory of image enhancement but it is the process of making images more useful. The reasons for doing this includes:

- Highlighting interesting detail in images
- Removing noise from images
- Making images more visually appealing

Image enhancement approaches falls into two broad categories: spatial domain methods and frequency domain methods. The term *spatial domain* refers to the image plane itself, and approaches in this category are based on the direct manipulation of pixels in an image. *Frequency domain* processing techniques are based on modifying the Fourier transform of an image.



3.1. Background

The term *spatial domain* refers to the aggregate of pixels composing an image. Spatial domain methods are procedures that operate directly on these pixels.

Most spatial domain enhancement operations can be reduced to the form

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

Where $f(x, y)$ is the input image, $g(x, y)$ is the processed image and T is some operator defined over some neighborhood of (x, y).

The principal approach in defining a neighborhood about a point (x, y) is to use a square or rectangular sub-image area centered at (x, y). The center of the sub-image is moved from pixel to pixel starting, say, at the top left corner. The operator T is applied at each location (x, y) to yield the output, g, at that location. The process utilizes only the pixels in the area of the image spanned by the neighborhood. Although other neighborhood shapes, such as approximations to a circle, sometimes are used, square and rectangular arrays are by far the most predominant because of their ease of implementation.

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

$$s = T(r)$$

Where, for simplicity in notation, r and s are variables denoting, respectively, the gray level of f(x, y) and g(x, y) at any point (x, y). For example, if T(r) has the form shown in Fig. (a), the effect of this transformation would be to produce an image of higher contrast than the original by darkening the levels below m and brightening the levels above m in the original image. In this technique, known as *contrast stretching*, the values of r below m are compressed by the transformation function into a narrow range of s, toward black. The opposite effect takes place for values of r above m. In the limiting case shown in Fig. (b), T(r) produces a two-level (binary) image. A mapping of this form is called a thresholding function. Some fairly simple, yet powerful, processing approaches can be formulated with gray-level transformations. Because enhancement at any point in an image depends only on the gray level at that point, techniques in this category often are referred to as *point processing.*

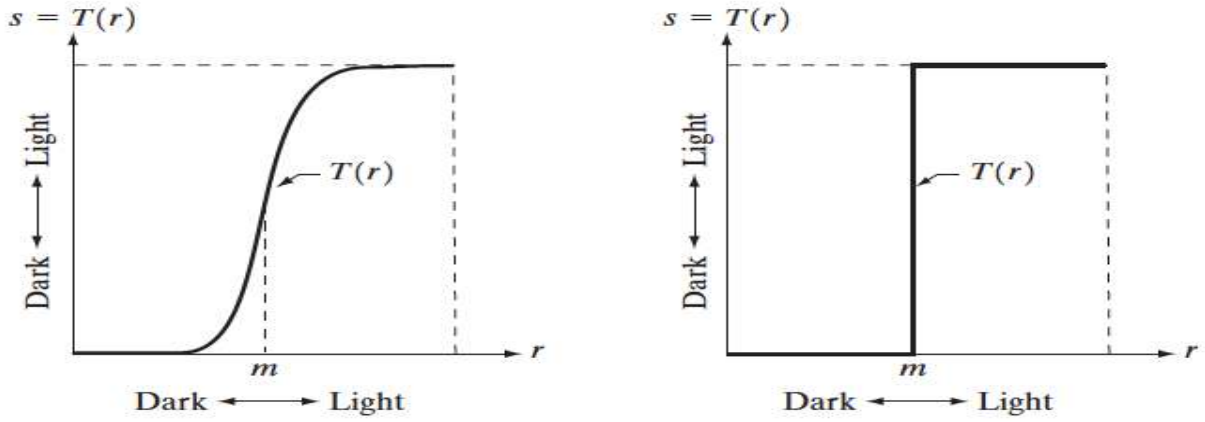
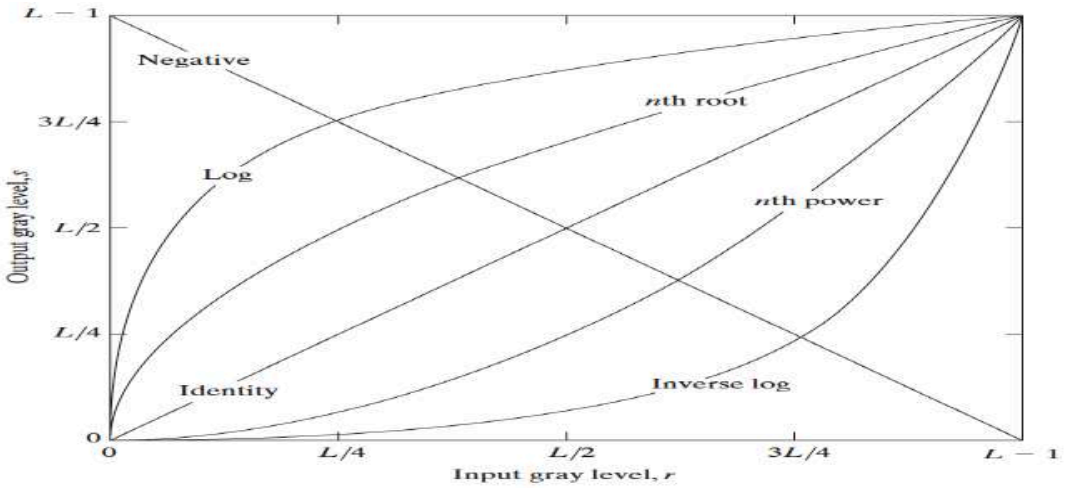


Fig a and b - Gray Level Transformation Functions for Contrast Enhancement

Larger neighborhoods allow considerably more flexibility. The general approach is to use a function of the values of f in a predefined neighborhood of **(x, y)** to determine the value of g at **(x, y)** . One of the principal approaches in this formulation is based on the use of so-called masks (also referred to as filters, kernels, templates, or windows). Basically, a mask is a small (say, 3* 3) 2-D array, in which the values of the mask coefficients determine the nature of the process, such as image sharpening. Enhancement techniques based on this type of approach often are referred to as mask processing or filtering.

3.2. Some Basic Gray Level Transformation



Some Basic Gray Level Transformation Used for Enhancement

3.2.1. Image Negatives

Negative images are useful for enhancing white or grey detail embedded in dark regions of an image. The negative of an image with gray levels in the range [0,L-1] is obtained by using the negative transformation shown above, which is given by the expression

$$s = L - 1 - r$$

Reversing the intensity levels of an image produces the equivalent of a photographic negative. This type of processing is particularly suited for enhancing white or gray detail embedded in dark regions of an image, especially when the black areas are dominant in size.

3.2.2. Logarithmic Transformation

The general form of the log transformation 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 figure above 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.

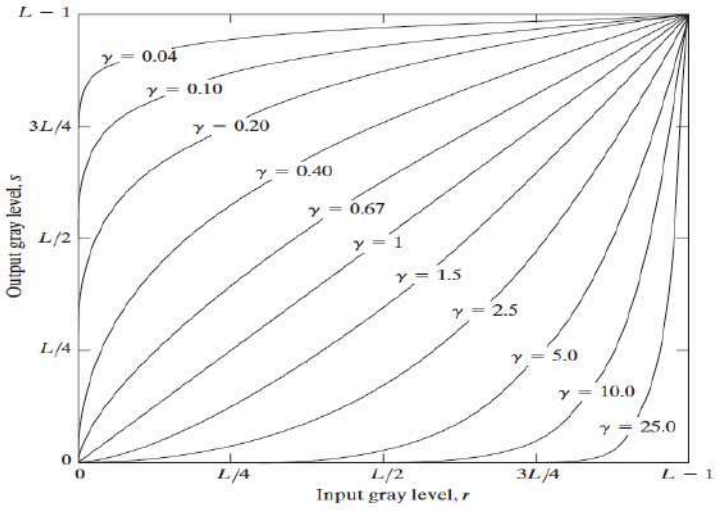
Log functions are particularly useful when the input grey level values may have an extremely large range of values

3.2.3. Power Law Transformation

Power law transformations have the following form: $s = c * r^{\gamma}$

Where c and $\gamma$are positive constants.

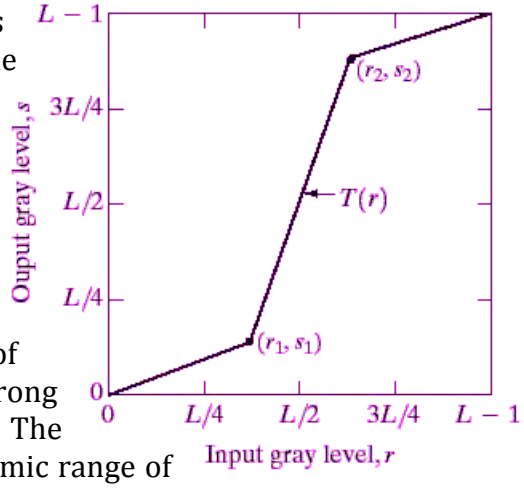
It maps a narrow range of dark input values into a wider range of output values or vice versa. Varying γ gives a whole family of curves. Many devices used for image capture, display and printing



respond according to a power law. The exponent in the power-law equation is referred to as gamma. The process of correcting for the power-law response is referred to as gamma correction. Example: – CRT devices have an intensity-to-voltage response that is; a power function (exponents typically range from 1.8-2.5). Gamma correction in this case could be achieved by applying the transformation s=r1/2.5=r^0.4.

3.2.4. Piecewise Linear Transformation Functions

A complementary approach to the methods discussed in the previous three sections is to use piecewise linear functions. The principal advantage of piecewise linear functions over the types of functions we have discussed thus far is that the form of piecewise functions can be arbitrarily complex. In fact, a practical implementation of some important transformations can be formulated only as piecewise functions. The principal disadvantage of piecewise functions is that their specification requires considerably more user input.



Contrast Stretching

One of the simplest piecewise linear-functions is a contrast-stretching transformation. Low-contrast images can result from poor illumination, lack of dynamic range in the imaging sensor, or even wrong setting of a lens aperture during image acquisition. The idea behind contrast stretching is to increase the dynamic range of the gray levels in the image being processed.

Rather than using a well defined mathematical function we can use arbitrary user-defined transforms. Contrast stretching expands the range of intensity levels in an image so it spans a given (full) intensity range.

Control points (r_1,s_1) and (r_2,s_2) control the shape of the transform T(r)

• r_1=r_2, s_1=0 and s_2=L-1 yields a thresholding function

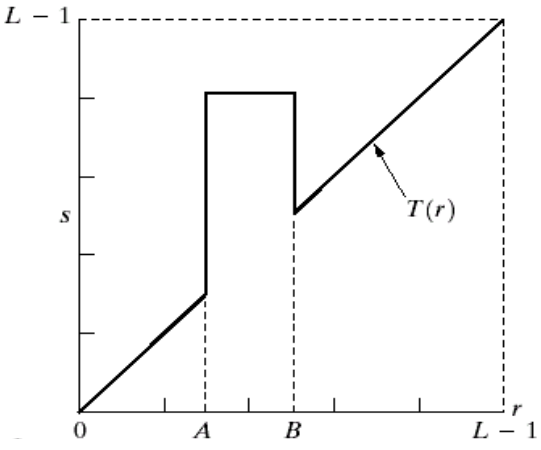
The contrast stretched image is obtained using the transformation obtained from the equation of the line having following points

• (r_1,s_1)=(r_{min},0) and (r_2, s_2)=(r_{max},L-1)

Gray Level Slicing

Used to highlight a specific range of intensities in an image that might be of interest. There are two common approaches

- Set all pixel values within a range of interest to one value (white) and all others to another value (black). Produces a binary image
- Brighten (or darken) pixel values in a range of interest and leave all others unchanged

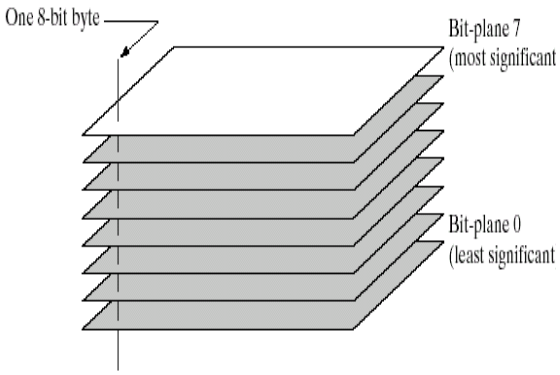


Highlights a specific range of grey levels

- Similar to thresholding
- Other levels can be suppressed or maintained
- Useful for highlighting features in an image

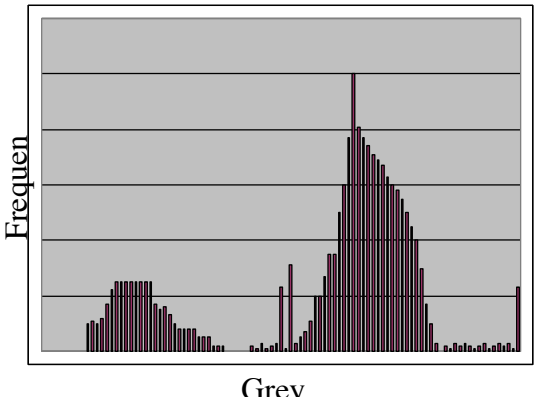
Bit Plane Slicing

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



Often by isolating particular bits of the pixel values in an image we can highlight interesting aspects of that image

- Higher-order bits usually contain most of the significant visual information
- Lower-order bits containsubtle details



3.3. Image Histograms

The histogram of an image shows us the distribution of grey levels in the image. It is massively useful in image processing, especially in segmentation.

The histogram of a digital image with gray levels in the range [0, L-1] is a discrete function $h(r_k)=n_k$, where r_k is the k th gray level and nk is the number of pixels in the image having gray level r_k . It is common practice to normalize a histogram by dividing each of its values by the total number of pixels in the image, denoted by n. Thus, a normalized histogram is given by $p(r_k)=n_k/n$, for k=0 , 1,…,L-1 . Loosely speaking, $p(r_k)$ gives an estimate of the probability of occurrence of gray level r_k . Note that the sum of all components of a normalized histogram is equal to 1.

Uses of Histogram Processing

- Image enhancements
- Image statistics
- Image compression
- Image segmentation
- Simple to calculate in software
- Economic hardware implementations
  - Popular tool in real-time image processing

A plot of this function for all values of k provides a global description of the appearance of the image (gives useful information for contrast enhancement). Histogram is commonly viewed in plots as

$h(r_k) = n_k$ versus r_k

$p(r_k) = n_k / MN$ versus r_k

3.3.1. Histogram Equalization

Histogram equalization is a process for increasing the contrast in an image by spreading

the histogram out to be approximately uniformly distributed. The gray levels of an image that has been subjected to histogram equalization are spread out and always reach white. The increase of dynamic range produces an increase in contrast. For images with low contrast, histogram equalization has the adverse effect of increasing visual graininess.

The intensity transformation function we are constructing is of the form

$$s = T(r);\ 0 \le r \le L-1$$

An output intensity level s is produced for every pixel in the input image having intensity r.

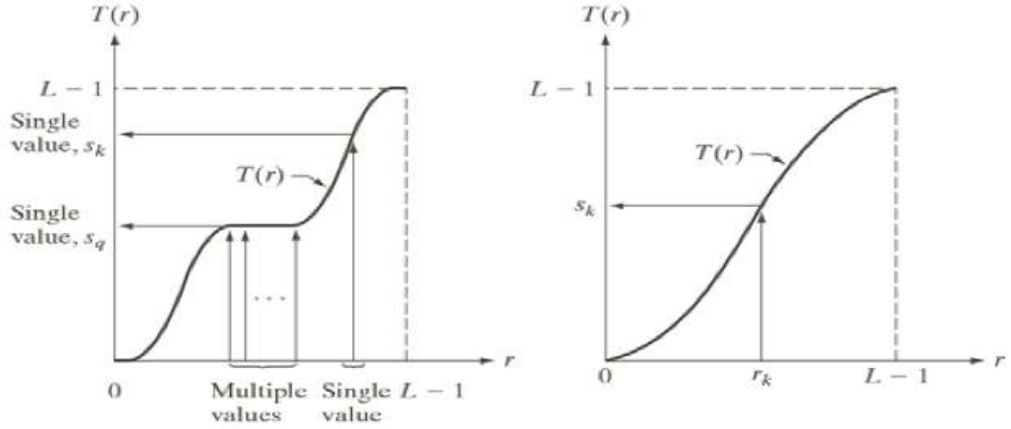
We assume $T(r)$ is monotonically increasing in the interval $0 \le r \le L-1$

$$\text{i.e. } 0 \le T(r) \le L-1; for 0 \le r \le L-1$$

If we define the inverse

$$r = T^{-1}(s); 0 \le s \le L-1$$

Then $T(r)$ should be strictly monotonically increasing.



a b

FIGURE 3.17 (a) Monotonically increasing function, showing how multiple values can map to a single value. (b) Strictly monotonically increasing function. This is a one-to-one mapping, both ways.

Histogram equalization requires construction of a transformation function s_k

$$s_k = T(r_k) = \sum_{j=0}^{k} \frac{n_j}{M \times N} \qquad s_k = T(r_k) = \frac{(L-1)}{M \times N} \sum_{j=0}^{k} n_j$$

where r_k is the kth gray level, n_k is the number of pixels with that gray level, MxN is the number of pixels in the image, and k=0,1,...,L-1. This yields an s with as many elements as the original image's histogram (normally 256 for our test images). The values of s will be in the range [0,1]. For constructing a new image, s would be scaled to the range [1,256].

Spreading out the frequencies in an image (or equalising the image) is a simple way to improve dark or washed out images

The formula for histogram equalisation is given as: $s_k = T(t_k) = \sum_{j=1}^{k} p_r(r_j) = \sum_{j=1}^{k} \frac{n_j}{n}$ where

- r_k: input intensity
- s_k: processed intensity
- k: the intensity range (e.g 0.0 – 1.0)
- n_j: the frequency of intensity j
- n: the sum of all frequencies

3.4. Enhancement Using Arithmetic/Logic Operations

Arithmetic/logic operations involving images are performed on a pixel-by-pixel basis between two or more images (this excludes the logic operation NOT, which is performed on a single image). As an example, subtraction of two images results in a new image whose pixel at coordinates (x, y) is the difference between the pixels in that same location in the two images being subtracted. Depending on the hardware and/or software being used, the actual mechanics of implementing arithmetic/logic operations can be done sequentially, one pixel at a time, or in parallel, where all operations are performed simultaneously.

Logic operations similarly operate on a pixel-by-pixel basis† .We need only be concerned with the ability to implement the AND, OR, and NOT logic operators because these three operators are *functionally complete*. In other words, any other logic operator can be implemented by using only these three basic functions. When dealing with logic operations on gray-scale images, pixel values are processed as strings of binary numbers. For example, performing the NOT operation on a black, 8-bit pixel (a string of eight 0' s) produces a white pixel (a string of eight 1's). Intermediate values are processed the same way, changing all 1' s to 0' s and vice versa. The AND and OR operations are used for masking; that is, for selecting subimages in an image. In the AND and OR image masks, light represents a binary 1 and dark represents a binary 0. Masking sometimes is referred to as *region of interest* (ROI) processing. In terms of enhancement, masking is used primarily to isolate an area for processing. This is done to highlight that area and differentiate it from the rest of the image. Logic operations also are used frequently in conjunction with morphological operations.

3.4.1. Image Subtraction

The difference between tow images $f(x, y)$and $h(x, y)$, expressed as

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

is obtained by computing the difference between all pairs of corresponding pixels from f and h. The key usefulness of subtraction is the enhancement of *differences* between images.

One of the most commercially successful and beneficial uses of image subtraction is in the area of medical imaging called *mask mode radiography*.

3.4.2. Image Averaging

Consider a noisy image $g(x, y)$ formed by the addition of noise $\eta(x, y)$ to an original image $f(x, y)$; that is,

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

where the assumption is that at every pair of coordinates (x, y) the noise is un-correlated and has zero average value. The objective of the following procedure is to reduce the noise content by adding set of noisy images, $\{g_i(x, y)\}$.

If the noise satisfies the constraints just sated, it can be shown that if an image $\bar{g}(x, y)$ is formed by averaging K different noisy images,

$$\bar{g}(x, y) = \frac{1}{K}\sum_{i=1}^{K} g_i(x, y)$$

then it follows that

$$E\{\bar{g}(x,y)\} = f(x,y)$$

and

$$\sigma^2_{\bar{g}(x,y)} = \frac{1}{K}\sigma^2_{\eta(x,y)}$$

where $E\{\bar{g}(x,y)\}$is the expected value of $\bar{g}$, and $\sigma^2_{\bar{g}(x,y)}$and $\sigma^2_{\eta(x,y)}$are the variances of $\bar{g}$and η, all at coordinates (x,y). The standard deviation at any point in the average image is

$$\sigma_{\bar{g}(x,y)} = \frac{1}{\sqrt{K}}\sigma_{\eta(x,y)}$$

As K increases, above equations indicate that the variability (noise) of the pixel values at each location (x,y) decreases. Because $E\{\bar{g}(x,y)\} = f(x,y)$, this means that $\bar{g}(x,y)$ approaches f(x,y) as the number of noisy images used in the averaging process increases. In practice, the images $g_i(x,y)$ must be registered (aligned) in order to avoid the introduction of blurring and other artifacts in the output image.

An important application of image averaging is in the field of astronomy, where imaging with very low light levels is routine, causing sensor noise frequently to render single images virtually useless for analysis.

3.5. Basics of Spatial Filtering

Some neighborhood operations work with the values of the image pixels in the neighborhood and the corresponding values of a subimage that has the same dimensions as the neighborhood. The subimage is called a *filter, mask, kernel, template, or window*, with the first three terms being the most prevalent terminology. The values in a filter subimage are referred to as coefficients, rather than pixels.

Use of spatial masks for filtering is called *spatial filtering*. This may be linear or nonlinear

3.5.1. Linear filters

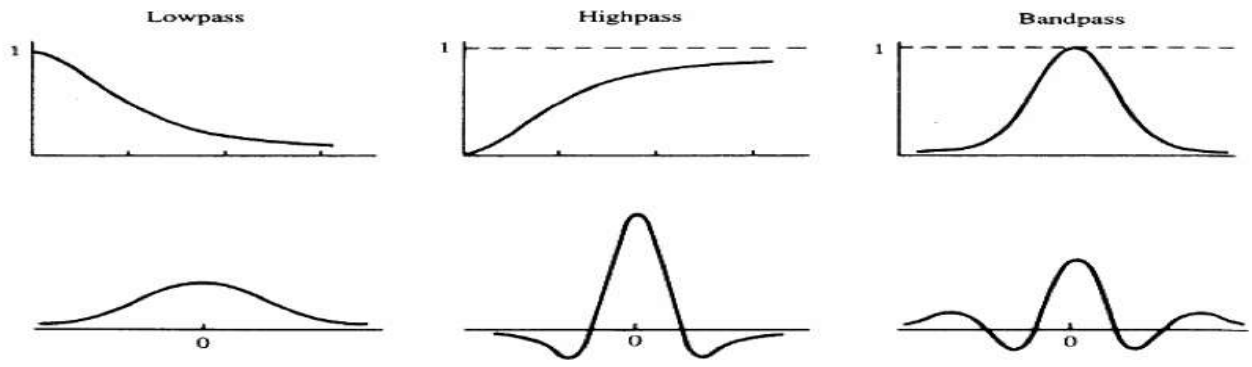
– Lowpass: attenuate (or eliminate) high frequency components such as characterized by edges and sharp details in an image. Net effect is image blurring

– Highpass: attenuate (or eliminate) low frequency components such as slowly varying characteristics. Net effect is a sharpening of edges and other details

- Bandpass: attenuate (or eliminate) a given frequency range. It is used primarily for image restoration (are of little interest for image enhancement).

Basic approach is to sum products between mask coefficients and pixel values.

$$R = w_1 z_1 + w_2 z_2 + \cdots + w_9 z_9$$



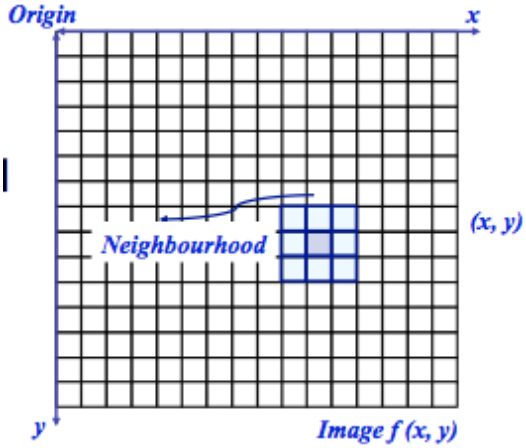
3.5.2. Non-Linear Filter

Nonlinear spatial filters also operate on neighborhoods. Their operation is based directly on pixel values in the neighborhood under consideration. They do not explicitly use coefficient values as in the linear spatial filter. Example: nonlinear spatial filters

- Median filter: Computes the median gray-level value of the neighborhood. Used for noise reduction.
- Max filter: Used to find the brightest points in an image: R=max{z_k |k=1,2,...,9}
- Min filter: Used to find the dimmest points in an image: R=min{z_k |k=1,2,...,9}

3.5.3. Neighborhood Operations

Neighbourhood operations simply operate on a larger neighbourhood of pixels than point operations. Neighbourhoods are mostly a rectangle around a central pixel. Any size rectangle and any shape filter are possible. Some simple neighbourhood operations include:

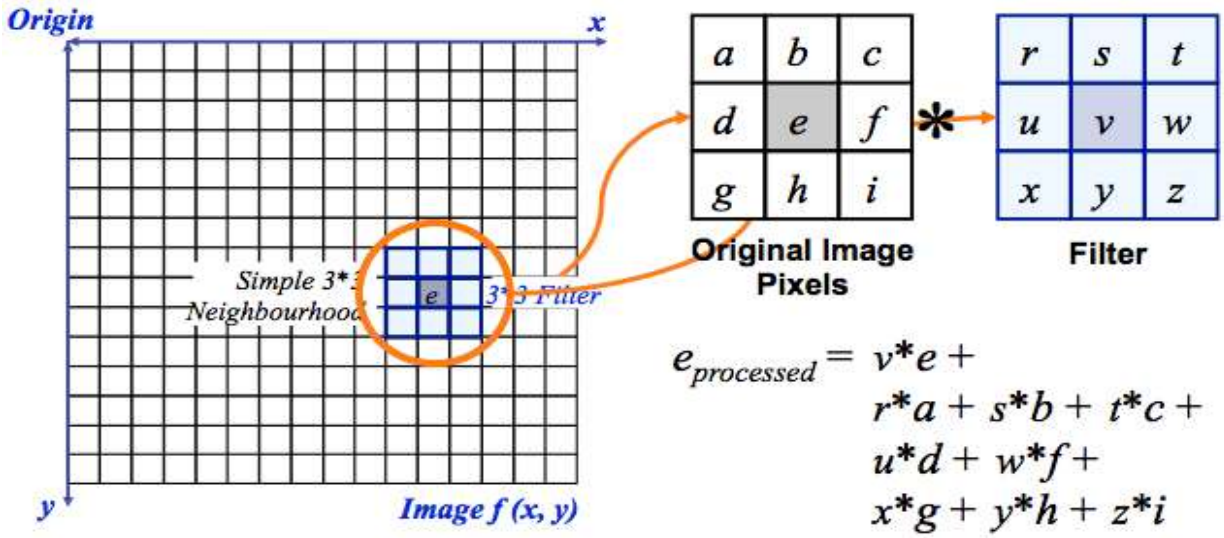


Min: Set the pixel value to the minimum in the neighbourhood

Max: Set the pixel value to the maximum in the neighbourhood

Median: The median value of a set of numbers is the midpoint value in that set (e.g. from the set [1, 7, 15, 18, 24] 15 is the median). Sometimes the median works better than the average.

3.6. Spatial Filtering Process



$$e_{processed} = v*e + r*a + s*b + t*c + u*d + w*f + x*g + y*h + z*i$$

The above is repeated for every pixel in the original image to generate the filtered image.

$$g(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x+s, y+t)$$

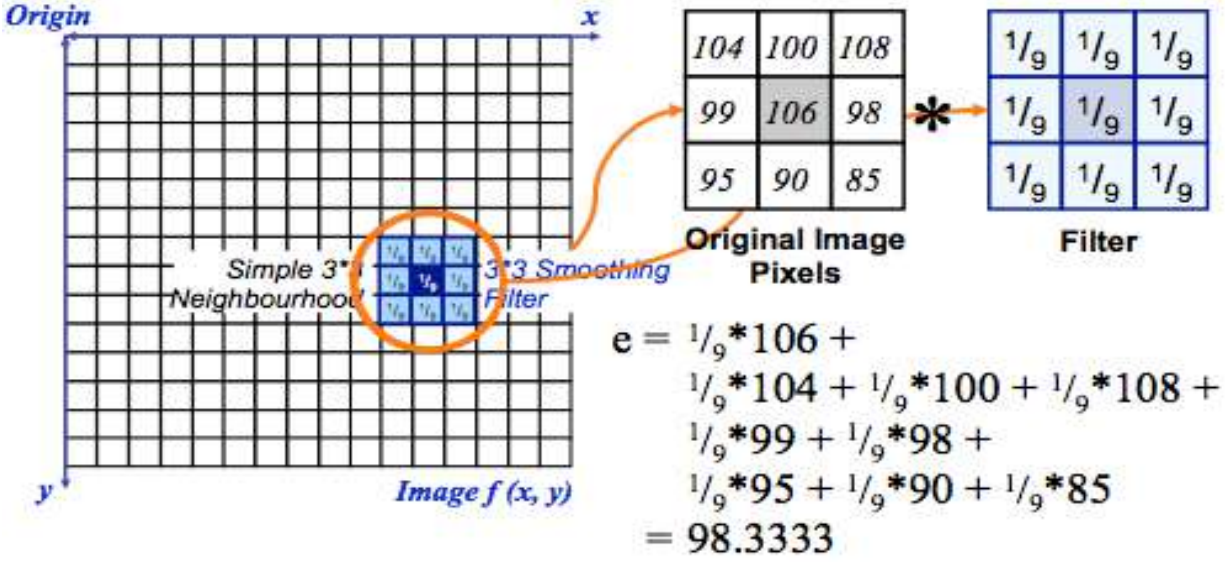
3.6.1. Smoothing Spatial Filters (Low Pass)

The shape of the impulse response needed to implement a lowpass (smoothing) filter indicates the filter should have all positive coefficients. For a 3x3 mask, the simplest arrangement is to have all

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

the coefficient values equal to one (neighborhood averaging). The response would be the sum of all gray levels for the nine pixels in the mask. This could cause the value of R to be out of the valid gray-level range. The solution is to scale the result by dividing by 9.

One of the simplest spatial filtering operations we can perform is a smoothing operation. Simply average all of the pixels in a neighbourhood around a central value. Especially useful in removing noise from images. Also useful for highlighting gross detail.



3.6.2. Weighted Smoothing Filters (Low Pass)

1/16	2/16	1/16
2/16	4/16	2/16
1/16	2/16	1/16

Allowing different pixels in the neighborhood different weights in the averaging function can generate more effective smoothing filters. Pixels closer to the central pixel are more important. Often this process is referred to as a *weighted averaging*. By smoothing the original image we get rid of lots of the finer detail which leaves only the gross features for thresholding.

One problem with the lowpass filter is it blurs edges and other sharp details. If the intent is to achieve noise reduction, one approach can be to use median filtering. The value of each pixel is replaced by the median pixel value in the neighborhood (as opposed to the average). It is particularly effective when the noise consists of strong, spike like components and edge sharpness is to be preserved. The median m of a set of values is such that half of the values are greater than m and half are less than m. To implement, sort the pixel values in the neighborhood, choose the median and assign this value to the pixel of interest. This process forces pixels with distinct intensities to be more like their neighbors.

Note: Filtering is often used to remove noise from images. Sometimes a median filter works better than an averaging filter.

3.6.3. Sharpening Filters (High Pass or High Boost))

$$\frac{1}{9}\begin{array}{|c|c|c|}\hline -1 & -1 & -1 \\ \hline -1 & 8 & -1 \\ \hline -1 & -1 & -1 \\ \hline \end{array}$$

The shape of the impulse response needed to implement a high-pass (sharpening) filter indicates the filter should have positive coefficients near its center and negative coefficients in the outer periphery. For a 3x3 mask, the simplest arrangement is to have the center coefficient positive and all others negative.

Note the sum of the coefficients is zero. When the mask is over a

constant or slowly varying region the output is zero or very small. This filter eliminates the zero-frequency term. Eliminating this term reduces the average gray-level value in the image to zero (will reduce the global contrast of the image). Thus, the result will be a somewhat edge-enhanced image over a dark background. Reducing the average gray-level value to zero implies some negative gray levels. The output should be scaled back into an appropriate range [0, L-1].

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

Sharpening spatial filters seek to highlight fine detail

- Remove blurring from images
- Highlight edges

Sharpening filters are based on *spatial differentiation*

A high-pass filter may be computed as:

High-pass = Original - Lowpass

• Multiplying the original by an amplification factor yields a high-boost or high-frequency-emphasis filter

High-boost = A(Original) – Lowpass

= (A – 1)(Original) + Original – Lowpass

= (A – 1)(Original) + High-pass

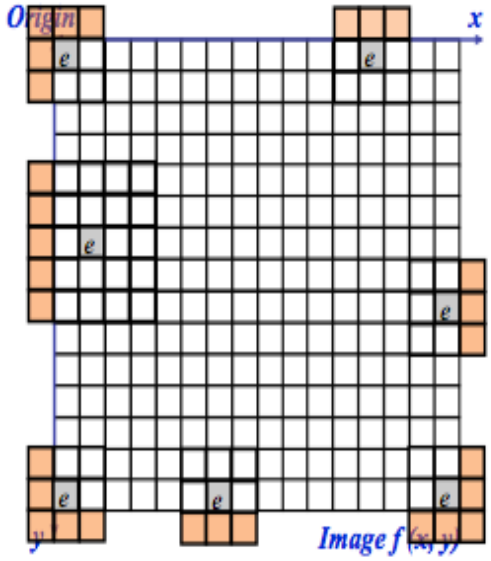
– If A>1, part of the original image is added to the high-pass result (partially restoring low frequency components)

– Result looks more like the original image with a relative degree of edge enhancement that depends on the value of A

- May be implemented with the center coefficient value w=9A-1 (A≥1)

3.6.4. Strange Things at the Edges

At the edges of an image we are missing pixels to form a neighbourhood



There are a few approaches to dealing with missing edge pixels:

Omit missing pixels

- Only works with some filters
- Can add extra code and slow down processing

Pad the image

- Typically with either all white or all black pixels

Replicate border pixels

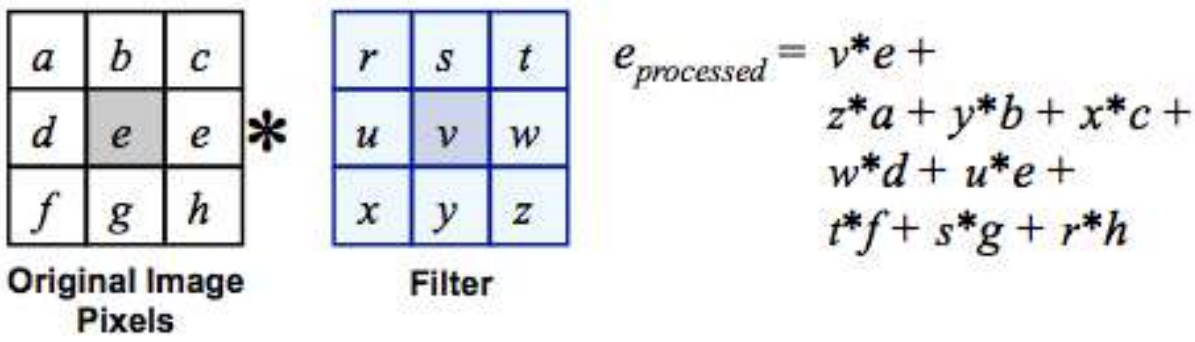
Truncate the image

Allow pixels *wrap around* the image

- Can cause some strange image artefacts

3.6.5. Correlation and Convolution

The filtering we have been talking about so far is referred to as *correlation* with the filter itself referred to as the *correlation kernel. Convolution* is a similar operation, with just one subtle difference.



For symmetric filters it makes no difference.

3.7. Spatial Differentiation

Differentiation measures the *rate of change* of a function. The formula for the 1st derivative of a function is as follows:

$$\frac{\partial f}{\partial x} = f(x+1) - f(x)$$

It's just the difference between subsequent values and measures the rate of change of the function.

The formula for the 2nd derivative of a function is as follows:

$$\frac{\delta^2 f}{\delta x^2} = f(x+1) + f(x-1) - 2f(x,y)$$

The 2nd derivative is more useful for image enhancement than the 1st derivative

- Stronger response to fine detail
- Simpler implementation

0	1	0
1	-4	1
0	1	0

The first sharpening filter we will look at is the *Laplacian*. It is isotropic and one of the simplest sharpening filters. The Laplacian is defined as follows:

$$\nabla^2 f = \frac{\delta^2 f}{\delta x^2} + \frac{\delta^2 f}{\delta y^2}$$

where the partial 2nd order derivative in the x direction is defined as follows:

$$\frac{\delta^2 f}{\delta x^2} = f(x+1,y) + f(x-1,y) - 2f(x,y)$$

and in the y direction as follows:

$$\frac{\delta^2 f}{\delta y^2} = f(x,y+1) + f(x,y-1) - 2f(x,y)$$

So, the Laplacian can be given as follows:

$$\nabla^2 f = f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1) - 4f(x,y)$$

0	-1	0
-1	5	-1
0	-1	0

Applying the Laplacian to an image we get a new image that highlights edges and other discontinuities. The result of a Laplacian filtering is not an enhanced image. We have to do more work in order to get our final image. Subtract the Laplacian result from the original image to generate our final sharpened enhanced image:

$$g(x,y) = f(x,y) - \nabla^2 f$$

$$or, g(x,y) = 5f(x,y) - f(x+1,y) - f(x-1,y) - f(x,y+1) - f(x,y-1)$$

This gives us a new filter which does the whole job for us in one step

3.7.1. 1st Derivative Filtering

Implementing 1st derivative filters is difficult in practice. For a function *f(x, y)* the gradient of *f* at coordinates *(x, y)* is given as the column vector:

$$\nabla f = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

The magnitude of this vector is given by:

$$\nabla f = mag(\nabla f)$$

$$= \sqrt{G_x^2 + G_x^2}$$

$$= \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

For practical reasons this can be simplified as:

$$\nabla f = |G_x| + |G_y|$$

The magnitude of the gradient at z_5 can be approximated in 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

$$\nabla f = \sqrt{(z_5 - z_8)^2 + (z_5 - z_6)^2}$$

or approximated as

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

A cross difference may also be used to approximate the magnitude of the gradient.

0	1
-1	0

$$\nabla f = \sqrt{(z_5 - z_9)^2 + (z_6 - z_8)^2}$$

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

This can be implemented by taking the absolute value of the response of the following two masks (the Roberts cross-gradient operators) and summing the results

1	0
0	-1

Extension to a 3*3 mask yields the following

$$\nabla f \approx |(z_7 + z_8 + z_9) - (z_1 + z_2 + z_3)| + |(z_3 + z_6 + z_9) - (z_1 + z_4 + z_7)|$$

The difference between the first and third rows & first and third columns approximates the derivative in the x-direction and y-direction respectively. The Prewitt operator masks may be used to implement the above approximation.

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

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

Prewitt

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

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

Sobel

The *Sobel operator* masks may also be used to implement the derivative approximation. The Sobel operators are widely used for edge detection. Note: All the mask coefficients for all the derivative filters sum to zero, indicating a 0 response in a constant area (as expected of a derivative operator).

After application of a derivative filter, generally the output will be scaled (as in the low pass and high pass cases). The result is then thresholded by setting any values above threshold to white (256 in MATLAB with a 256 gray-level color-map) and all below the threshold are set to black (1 in MATLAB) or left with their initial values in f(x,y). the derivative filters may be applied:

- Only in the x-direction
- Only in the y-direction
- In both directions (taking either the sum or maximum of the responses from the filter masks)

Comparison between 1st and 2nd Derivative

Comparing the 1st and 2nd derivatives we can conclude the following:

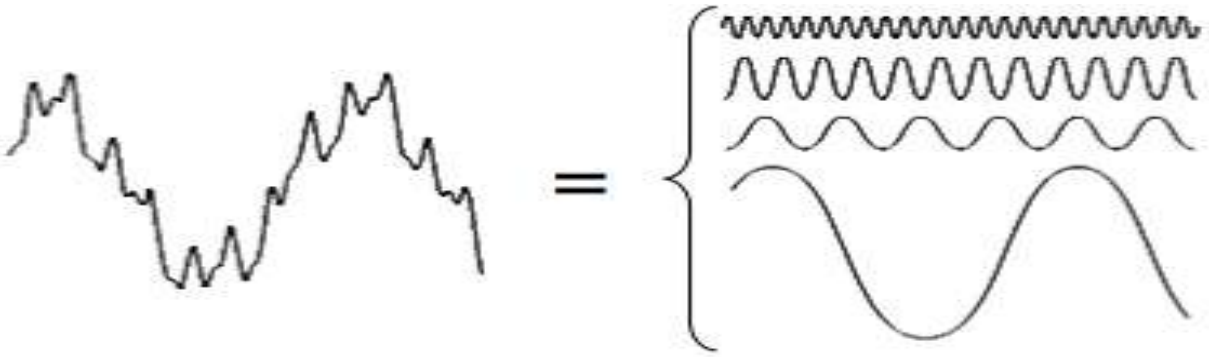
- 1st order derivatives generally produce thicker edges
- 2nd order derivatives have a stronger response to fine detail e.g. thin lines
- 1st order derivatives have stronger response to grey level step
- 2nd order derivatives produce a double response at step changes in grey level

Successful image enhancement is typically not achieved using a single operation, rather we combine a range of techniques in order to achieve a final result.

4. Image Enhancement in the Frequency Domain

4.1. Background

Any function that periodically repeats itself can be expressed as a sum of sines and cosines of different frequencies each multiplied by a different coefficient – a *Fourier series*.



It does not matter how complicated the function is; as long as it is periodic and meets some mild mathematical conditions, it can be represented by such a sum.

Even functions that are not periodic (but whose area under the curve is finite) can be expressed as the integral of sines and/or cosines multiplied by a weighing function. The formulation in this case is the *Fourier Transform*, and its utility is even greater than the Fourier series in most practical problem.

Both representations share the important characteristic that a function, expressed in either a Fourier series or transform, can be reconstructed completely via an inverse process with no loss of information.

4.2. Introduction to the Fourier Transform and the Frequency Domain

The Fourier transforms, $F(u)$, of a single variable. Continuous function, $F(x)$, is defined by the equation

$$F(u) = \int_{-\infty}^{\infty} f(x)e^{-j2\pi ux}\,dx$$

where $j = \sqrt{-1}$. Conversely, given, $F(u)$, we can obtain, $F(x)$, by means of *inverse* Fourier transform

$$F(x) = \int_{-\infty}^{\infty} f(u)e^{j2\pi ux}\,du$$

These two equations comprise the *Fourier transform pair*. These two equations exist if $F(x)$ is continuous and integrable and $F(u)$ is integrable. These conditions are almost always satisfied in practice. We are concerned with functions $F(x)$, which are real, however the Fourier transform of a real function is, generally, complex. So,

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

where $R(u)$ and $I(u)$denote the real and imaginary components of $F(u)$respectively.

Expressed in exponential form, $F(u)$is: $F(u) = |F(u)|e^{j\varphi(u)}$

Where, $|F(u)| = \sqrt{R^2(u) + I^2(u)}$ and $\varphi(u) = \tan^{-1}\frac{I(u)}{R(u)}$

The magnitude function $|F(u)|$is called the *Fourier spectrum* of $F(x)$ and $\varphi(u)$is the phase angle.

The square of the spectrum, $P(u) = |F(u)|^2 = R^2(u) + I^2(u)$, is commonly called the *power spectrum* (or the spectral density) of f(x).

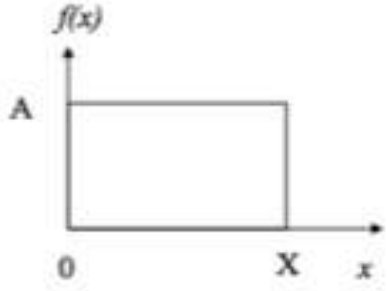
The variable u is often called the frequency variable. This name arises from the expression of the exponential term $e^{-j2\pi ux}$in terms of sines and cosines (from Euler's formula):

$$e^{-j2\pi ux} = \cos(2\pi ux) - j\sin(2\pi ux)$$

Interpreting the integral in the Fourier transform equation as a limit summation of discrete terms make it obvious that:

- $F(u)$is composed of an infinite sum of sine and cosine terms.
- Each value of u determines the frequency of its corresponding sine-cosine pair.

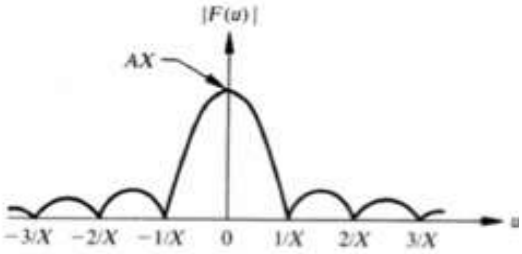
Consider the following simple function. The Fourier transform is:



$$\begin{aligned} F(u) &= \int_{-\infty}^{+\infty} f(x)\exp[-j2\pi ux]dx \\ &= \int_0^X A\exp[-j2\pi ux]dx \\ &= \frac{-A}{j2\pi u}[e^{-j2\pi ux}]_0^X = \frac{-A}{j2\pi u}[e^{-j2\pi uX} - 1] \\ &= \frac{A}{j2\pi u}[e^{j\pi uX} - e^{-j\pi uX}]e^{-j\pi uX} \\ &= \frac{A}{\pi u}\sin(\pi uX)e^{-j\pi uX} \end{aligned}$$

This is a complex function. The Fourier spectrum is:

$$\begin{aligned} |F(u)| &= \left|\frac{A}{\pi u}\right| |\sin(\pi uX)||e^{-j\pi ux}| \\ &= AX\left|\frac{\sin(\pi uX)}{(\pi uX)}\right| \end{aligned}$$



4.2.1. The 2-D Fourier Transform

The Fourier transform can be extended to 2 dimensions:

$$F(u,v) = \iint_{-\infty}^{\infty} f(x,y)e^{-j2\pi(ux+vy)}\,dx\,dy$$

and the inverse transform

$$f(x,y) = \iint_{-\infty}^{\infty} f(u,v)e^{j2\pi(ux+vy)}\,dx\,dy$$

The 2-D Fourier spectrum is:

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

The phase angle is: $\varphi(u,v) = \tan^{-1}\left[\frac{I(u,v)}{R(u,v)}\right]$

The power spectrum is: $P(u,v) = |F(u,v)|^2 = R^2(u,v) + I^2(u,v)$

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

$$= A\int_0^X \exp[-j2\pi ux]dx \int_0^Y \exp[-j2\pi vy]dy$$

$$= A[\frac{e^{-j2\pi ux}}{-j2\pi u}]_0^X [\frac{e^{-j2\pi vy}}{-j2\pi v}]_0^Y$$

$$= \frac{A}{-j2\pi u}[e^{-j2\pi uX} - 1]\frac{1}{-j2\pi y}[e^{-j2\pi vY} - 1]$$

$$= AXY[\frac{\sin(\pi uX)e^{-j\pi uX}}{(\pi uX)}][\frac{\sin(\pi vY)e^{-j\pi vY}}{(\pi vY)}]$$

The spectrum is $|F(u,v)| = AXY[\frac{\sin(\pi uX)}{(\pi uX)}][\frac{\sin(\pi vY)}{(\pi vY)}]$

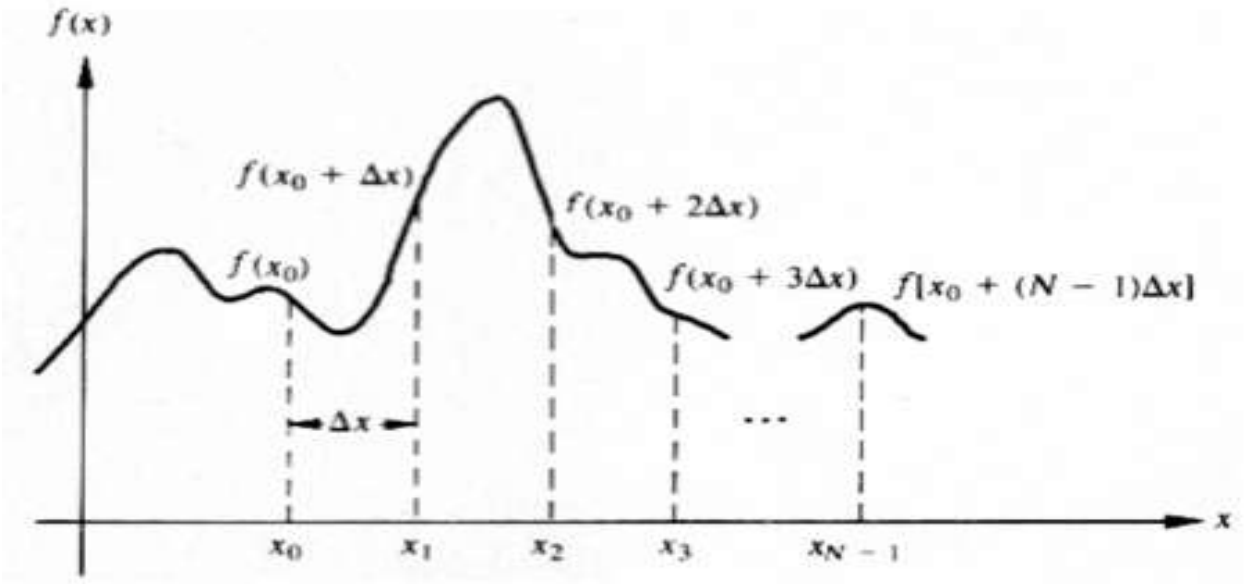
4.3. The Discrete Fourier Transform

Suppose a continuous function, f(x), is discretized into a sequence by taking N samples Δx units apart.

$$\{f(x_0), f(x_0 + \Delta x), f(x_0 + 2\Delta x), \ldots, f(x_0 + [N-1]\Delta x)\}$$

Let x refer to either a continuous or discrete value by saying $f(x) = f(x_0 + x\Delta x)$

Where x assumes the discrete values 0, 1, ... , N-1 and $f(0), f(1), \ldots, f(N-1)$denotes any N uniformly spaced samples from a corresponding continuous function.



The discrete Fourier transform is given by:

$$F(u) = \frac{1}{N}\sum_{x=0}^{N-1} f(x)e^{-j2\pi ux/N}\,;\mathrm{u} = 0, 1, \ldots, \mathrm{N}-1$$

The discrete inverse Fourier transform is given by:

$$f(x) = \sum_{x=0}^{N-1} f(u)e^{j2\pi ux/N}\,; x = 0, 1, \dots, N-1$$

The values of u $= 0, 1, \dots,$ N $- 1$in the discrete case correspond to samples of the continuous transform at 0, Δu, 2Δu, … , (N $- 1)\Delta u$ and Δu $= \frac{1}{N\Delta x}$

In the 2-D case:

$$\mathrm{F(u,v)} = \frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)e^{-j2\pi\left(\frac{ux}{M}+\frac{vy}{N}\right)}\,; u = 0 \to M-1 \& v = 0 \to N-1$$

$$\mathrm{f(x,y)} = \sum_{u=0}^{M-1}\sum_{v=0}^{N-1} F(u,v)e^{j2\pi\left(\frac{ux}{M}+\frac{vy}{N}\right)}\,; x = 0 \to M-1 \& y = 0 \to N-1$$

The discrete function f(x,y) represents samples of the continuous function at

$$f(x_0 + x\Delta x, y_0 + y\Delta y)$$

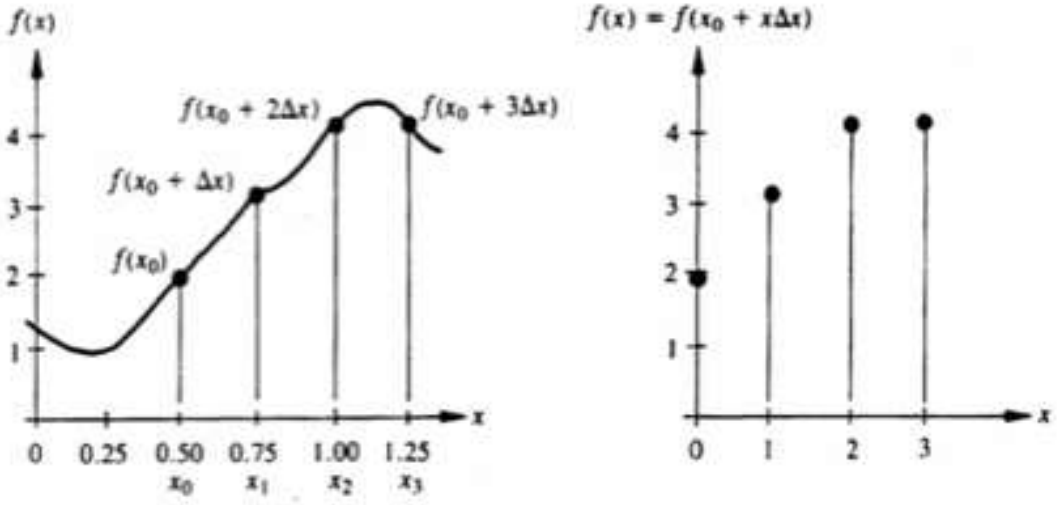
$$\Delta u = \frac{1}{M\Delta x} and \Delta v = \frac{1}{N\Delta y}$$

when N=M (such as in a square image)

$$\mathrm{F(u,v)} = \frac{1}{N}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} f(x,y)e^{-j2\pi(us+vy)/N}$$

And

$$\mathrm{f(x,y)} = \frac{1}{N}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} F(u,v)e^{j2\pi(us+vy)/N}$$



- Consider sampling at x_0=.5, x_1=.75, x_2=1.0, and x_3=1.25
- Here Δx=.25 and x ranges from 0 → 3

The four corresponding Fourier transform terms are

$$F(0)=\frac{1}{4}\sum_{x=0}^{3} f(x)\exp[0]$$
$$=\frac{1}{4}[f(0)+f(1)+f(2)+f(3)]$$
$$=\frac{1}{4}[2+3+4+4]$$
$$=3.25$$

$$F(1)=\frac{1}{4}\sum_{x=0}^{3} f(x)\exp[-j2\pi/4]$$
$$=\frac{1}{4}[2e^{0}+3e^{-j\pi/2}+4e^{-j\pi}+4e^{-j3\pi/2}]$$
$$=\frac{1}{4}[-2+j]$$

$$F(2)=-\frac{1}{4}[1+0j]$$

$$F(3)=-\frac{1}{4}[2+j]$$

The Fourier spectrum is then

$$|F(0)|=3.25$$
$$|F(1)|=[(2/4)^2+(1/4)^2]^{1/2}=\sqrt{5}/4$$
$$|F(2)|=[(1/4)^2+(0/4)^2]^{1/2}=1/4$$
$$|F(3)|=[(2/4)^2+(1/4)^2]^{1/2}=\sqrt{5}/4$$

4.3.1. Properties of 2-D Fourier Transform

The dynamic range of the Fourier spectra is generally higher than can be displayed. A common technique is to display the function

$$D(u,v)=c\ log[1+|F(u,v)|]$$

where c is a scaling factor and the logarithm function performs a "compression" of the data. 'c' is usually chosen to scale the data into the range of the display device, [0.255] typically ([1-256 for 256 gray-level MATLAB image)

Separability

The discrete transform pair can be written in separable forms

$$F(u,v)=\frac{1}{N}\sum_{x=0}^{N-1} e^{-j2\pi ux/N}\sum_{x=0}^{N-1} f(x,y)e^{-j2\pi ux/N}\ ;u,v=0,1,\dots N-1$$

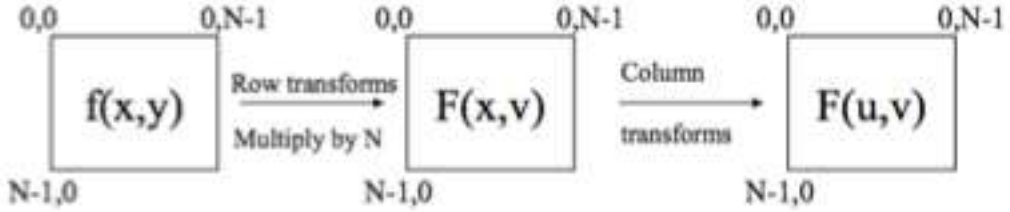
$$f(x,y)=\frac{1}{N}\sum_{u=0}^{N-1} e^{j2\pi ux/N}\sum_{x=0}^{N-1} F(u,v)e^{j2\pi uy/N}\ ;x,y=0,1,\dots N-1$$

So, F(u,v) or f(x,y) can be obtained in 2 steps by successive applications of the 1-D Fourier transform or its inverse. The 2-D transform can be expressed as

$$F(u,v)=\frac{1}{N}\sum_{x=0}^{N-1} F(x,v)e^{-j2\pi ux/N}$$

where

$$F(x,v)=N[\tfrac{1}{N}\textstyle\sum_{y=0}^{N-1} f(x,y)e^{-\frac{j2\pi uy}{N}}]$$



Translation

The translation properties of the Fourier transform pair are

$$f(x,y)e^{\frac{j2\pi(u_0x+v_0y)}{N}} \Leftrightarrow F(u-u_0, v-v_0)$$

and

$$f(x-x_0, y-y_0) \Leftrightarrow F(u,v)e^{-j2\pi(ux_0+vy_0)/N}$$

where the double arrow indicates a correspondence between a function and its Fourier transform (or vice versa). Multiplying f(x,y) by the exponential and taking the transform results in a shift of the origin of the frequency plane to the point (u_0, v_0).

For our purposes, $u_0 = v_0 = N/2$. Therefore,

$$e^{j2\pi(u_0x+v_0y)/N} = e^{j\pi(x+y)} = -1^{x+y}$$

and

$$f(x,y)(-1)^{x+y} \Leftrightarrow F(u-\frac{N}{1}, v-\frac{N}{2})$$

So, the origin of the Fourier transform of f(x,y) can be moved to the center of the corresponding N*N simply by multiplying f(x,y) by $(-1)^{x+y}$ before taking the transform. This does not affect the magnitude of the Fourier transform.

Periodicity of the Fourier Transform

The discrete Fourier transform (and its inverse) are periodic with period N.

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

Although F(u,v) repeats itself infinitely for many values of u and v, only N values of each variable are required to obtain f(x,y) from F(u,v).

i.e. Only one period of the transform is necessary to specify F(u,v) in the frequency domain.

Conjugate Symmetry of the Fourier transform

If f(x,y) is real (true for all of our cases), the Fourier transform exhibits conjugate symmetry: F(u,v)=F*(-u,-v); F*(u,v) is the complex conjugate of F(u,v)

Or the more interesting: |F(u,v)|=|F(-u,-v)|

Implications of Periodicity and Symmetry

Consider a 1-D case:

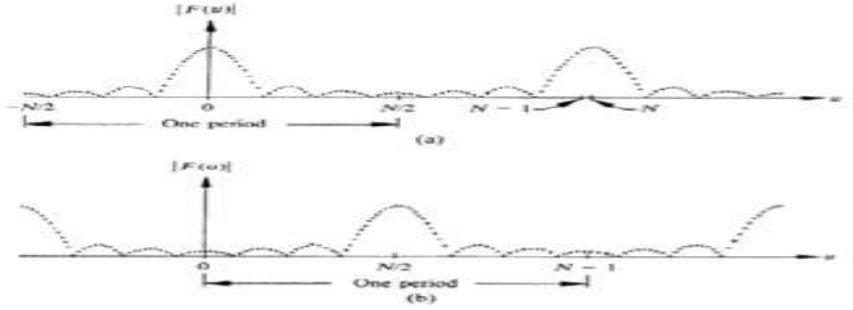
F(u)=F(u+N) indicates F(u) has a period of length N

|F(u)|=|F(-u)| shows the magnitude is centered about the origin.

Because the Fourier transform is formulated for values in the range from [0,N-1], the result is two back-to-back half periods in this range. To display one full period in the range, move (shift) the origin of the transform to the point u=N/2

Fourier spectrum with back-to-back half periods in the range [0,n-1]

Shifted spectrum with a full period in the same range



Distributivity & Scaling

The Fourier transform (and its inverse) are distributive over addition but not over multiplication. So,

$$af(x,y) \Leftrightarrow aF(u,v)$$

$$f(ax,by) \Leftrightarrow \frac{1}{|ab|}F(u/a,v/b)$$

for two scalars a and b

$$\Im\{f_1(x,y)+f_2(x,y)\} = \Im\{f_1(x,y)\}+\Im\{f_2(x,y)\}$$

$$\Im\{f_1(x,y)\times f_2(x,y)\} \neq \Im\{f_1(x,y)\}\times\Im\{f_2(x,y)\}$$

Average Value

- A widely used expression for the average value of a 2-D discrete function is:

$$\bar{f}(x,y) = \frac{1}{N^2}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} f(x,y)$$

- From the definition of F(u,v), for u=v=0,

$$F(0,0) = \frac{1}{N}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} f(x,y)$$

- Therefore, $\bar{f}(x,y) = \frac{1}{N}F(0,0)$

The Laplacian

The Laplacian of a two variable function f(x,y) is given as: $\nabla^2 f(x,y) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$

From the definition of the 2-D Fourier transform,

$$\Im\{\nabla^2 f(x,y)\} \Leftrightarrow -(2\pi)^2(u^2+v^2)F(u,v)$$

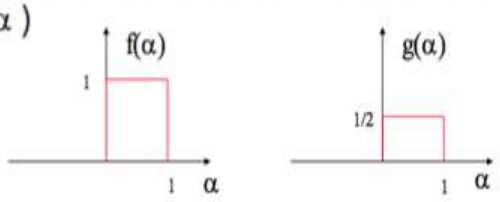
The Laplacian operator is useful for outlining edges in a image.

Convolution and Correlation

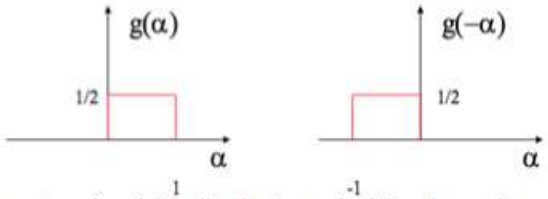
- The convolution of two functions f(x) and g(x) is denoted f(x)*g(x) and is given by:

$$f(x)*g(x) = \int_{-\infty}^{+\infty} f(\alpha)g(x-\alpha)d\alpha$$

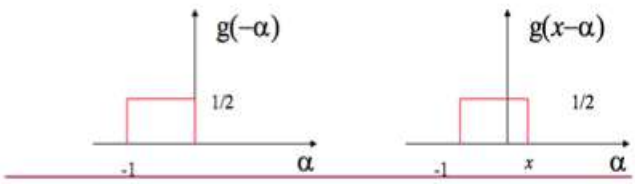
- Where α is a dummy variable of integration.
- Example: Consider the following functions f(α) and g(α)



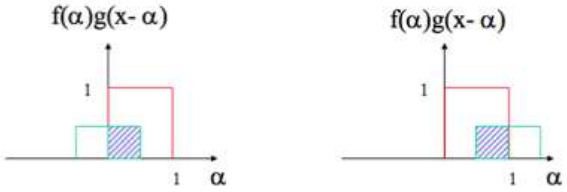
- Compute g(-α) by folding g(α) about the origin



- Compute g(x-α) by displacing g(-α) by the value :



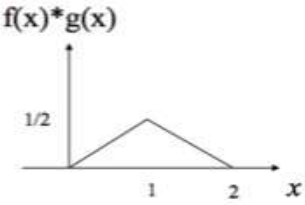
- Then, for any value x, we multiply $g(x-\alpha)$ and $f(\alpha)$ and integrate from $-\infty$ to $+\infty$
- For $0 \le x \le 1$ we have For $1 \le x \le 2$ we have



- Thus we have

$$f(x)*g(x)=\begin{cases} x/2 & 0\le x\le 1 \\ 1-x/2 & 1\le x\le 2 \\ 0 & elsewhere. \end{cases}$$

- Graphically,



Convolution and Impulse Function

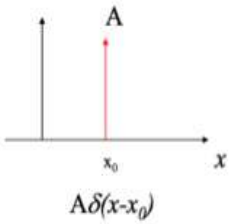
- Of particular interest will be the convolution of a function $f(x)$ with an impulse function $\delta(x-x_0)$

$$\int_{-\infty}^{+\infty} f(x)\delta(x-x_0)dx = f(x_0)$$

- The function $\delta(x-x_0)$ may be viewed as having ar area of unity in an infinitesimal neighborhood around x_0 and 0 elsewhere. That is

$$\int_{-\infty}^{+\infty}\delta(x-x_0)dx = \int_{x_0^-}^{x_0^+}\delta(x-x_0)dx = 1$$

- We usually say that $\delta(x-x_0)$ is located at $x=x_0$ and the strength of the impulse is given by the value of f(x) at $x=x_0$
- If $f(x)=A$ then, $A\delta(x-x_0)$ is impulse of strength A at $x=x_0$.
- Graphically this is:



$A\delta(x-x_0)$

Convolution and the Fourier Transform

- f(x)*g(x) and F(u)G(u) form a Fourier transform pair
- If f(x) has transform F(u) and g(x) has transform G(u) then f(x)*g(x) has transform F(u)G(u)

$$f(x)*g(x) \Leftrightarrow F(u)G(u)$$
$$f(x)g(x) \Leftrightarrow F(u)*G(u)$$

- These two results are commonly referred to as the *convolution theorem*

Frequency Domain Filtering

- Enhancement in the frequency domain is straightforward
  - Compute the Fourier transform
  - Multiply the result by a filter transform function
  - Take the inverse transform to produce the enhanced image
- In practice, small spatial masks are used considerably more than the Fourier transform because of their simplicity of implementation and speed of operation
- However, some problems are not easily addressable by spatial techniques
  - Such as homomorphic filtering and some image restoration techniques

Lowpass Frequency Domain Filtering

- Given the following relationship

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

- where F(u,v) is the Fourier transform of an image to be smoothed
- The problem is to select an H(u,v) that yields an appropriate G(u,v)
- We will consider *zero-phase-shift* filters that do not alter the phase of the transform (i.e. they affect the real and imaginary parts of F(u,v) in exactly the same manner)

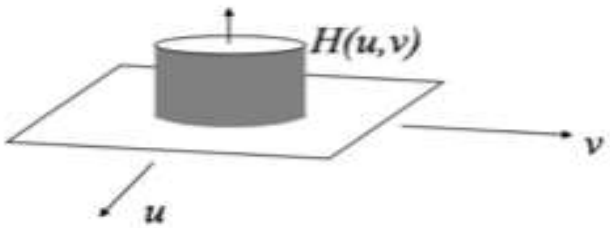
Ideal LowPass Filter (ILPF)

- A transfer function for a 2-D ideal lowpass filter (ILPF) is given as

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

- where D_0 is a stated nonnegative quantity (the cutoff frequency) and D(u,v) is the distance from the point (u,v) to the center of the frequency plane

$$D(u,v) = \sqrt{u^2 + v^2}$$



- The point D_0 traces a circle from the frequency origin giving a locus of cutoff frequencies (all are at distance D_0 from the origin)
- One way to establish a set of "standard" loci is to compute circles that encompass various amounts of the total signal power P_T
- P_T is given by

$$P_T = \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} P(u,v)$$

- where P(u,v) is given as

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

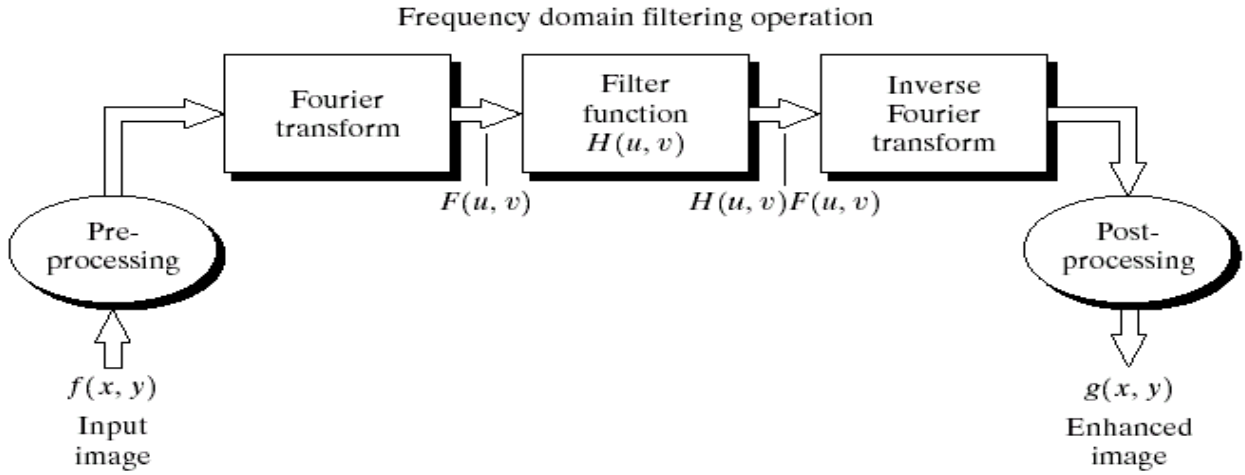
- For the centered transform, a circle of radius *r* encompasses β percent of the power, where

$$\beta = 100\left[\sum_u \sum_v P(u,v) / P_T\right]$$ (the summation is over all points (u, v) encompassed by the circle)

The DFT and Image Processing

To filter an image in the frequency domain:

1. Multiply the input image by $(-1)^{x+y}$ to center the transform.
2. Compute *F(u,v)* the DFT of the image
3. Multiply *F(u,v)* by a filter function *H(u,v)*
4. Compute the inverse DFT of the result in (3)
5. Obtain the real part of the result in (4)
6. Multiply the result in (5) by $(-1)^{x+y}$



Smoothing Frequency Domain Filters

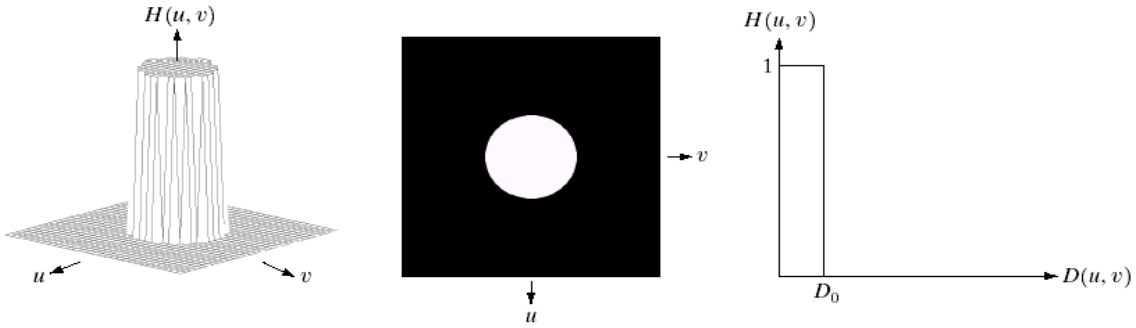
Smoothing is achieved in the frequency domain by dropping out the high frequency components. The basic model for filtering is:

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

where *F(u,v)* is the Fourier transform of the image being filtered and *H(u,v)* is the filter transform function. *Low pass filters* – only pass the low frequencies, drop the high ones.

4.4. Ideal Low Pass Filter

Simply cut off all high frequency components that are a specified distance D_0 from the origin of the transform



Changing the distance changes the behavior of the filter.

The transfer function for the ideal low pass filter can be given as:

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

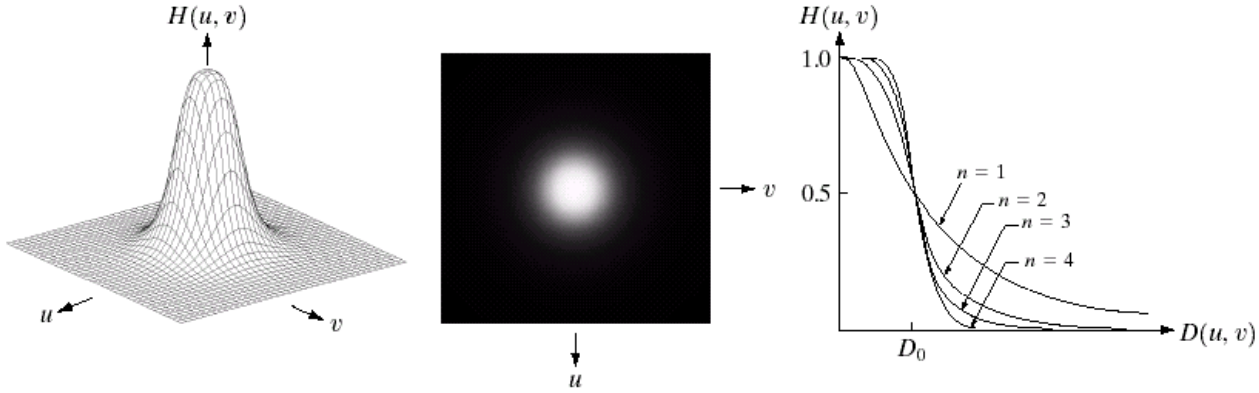
where *D(u,v)* is given as:

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

4.4.1. Butterworth LowPass Filters

The transfer function of a Butterworth lowpass filter of order *n* with cutoff frequency at distance D_0 from the origin is defined as:

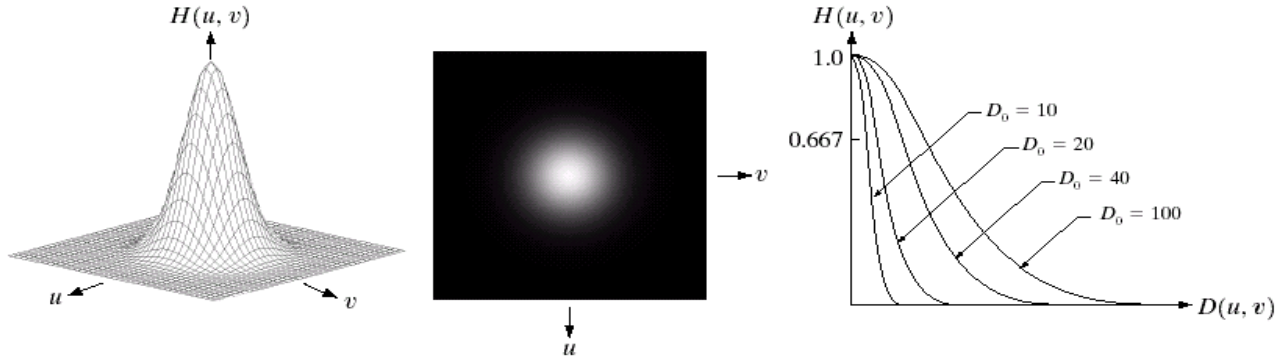
$$H(u,v) = \frac{1}{1 + [\frac{D(u,v)}{D_0}]^{2n}}$$



4.4.2. Gaussian Lowpass Filters

The transfer function of a Gaussian lowpass filter is defined as:

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



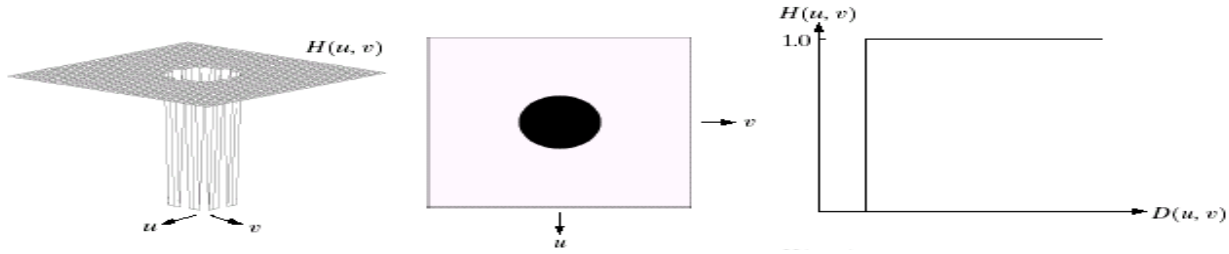
- A low pass Gaussian filter is used to connect broken text.
- Different lowpass Gaussian filters used to remove blemishes in a photograph

4.5. Sharpening in the Frequency Domain (High Pass Filters)

Edges and fine detail in images are associated with high frequency components. *High pass filters* – only pass the high frequencies, drop the low ones. High pass frequencies are precisely the reverse of low pass filters, so: $H_{hp}(u, v) = 1 - H_{lp}(u, v)$

The ideal high pass filter is given as: where D_0 is the cut off distance as before.

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

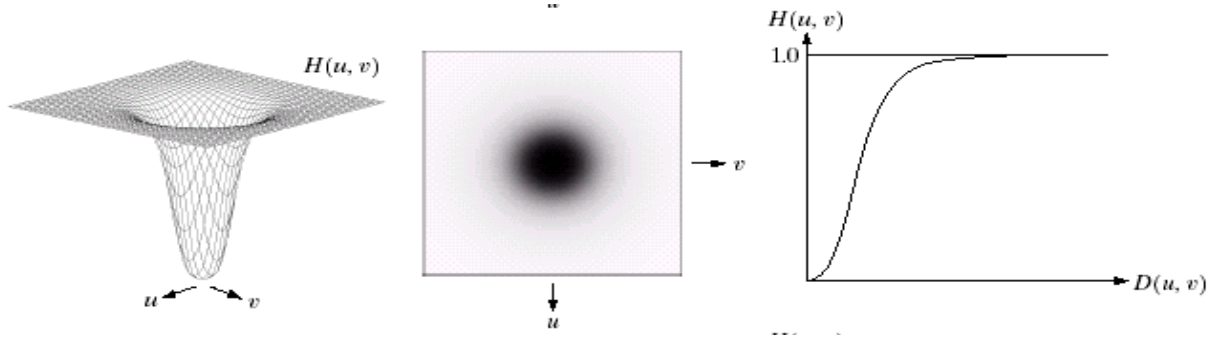


4.5.1. Butterworth High Pass Filters

The Butterworth high pass filter is given as:

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

where *n* is the order and D_0 is the cut off distance as before

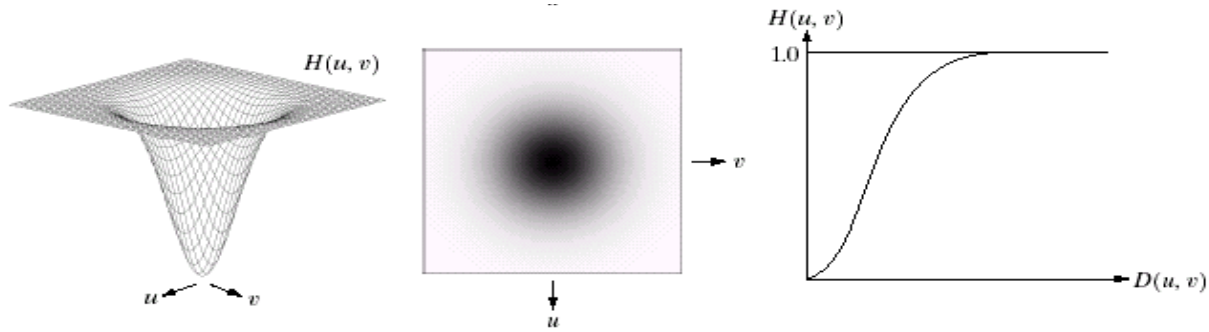


4.5.2. Gaussian High Pass Filters

The Gaussian high pass filter is given as:

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

where D_0 is the cut off distance as before



4.5.3. Highpass Filter Comparison

$$Unsharp\ Masking\ f_{hp}(x,y) = f(x,y) - f_{lp}(x,y) \qquad (1)$$

$$High-boost\ filtering\ f_{hb}(x,y) = Af(x,y) - f_{lp}(x,y)\ where\ (A \geq 1) \qquad (2)$$

$$f_{hb}(x,y) = (A-1)f(x,y) + f(x,y) - f_{lp}(x,y) \qquad (3)$$

$$f_{hb}(x,y) = (A-1)f(x,y) + f_{hp}(x,y) \qquad (4)$$

$$From\ (1)\ for\ Frequency\ Domain\ F_{hp}(u,v) = F(u,v) - F_{lp}(u,v) \quad (5)$$

$$F_{lp}(u,v) = H_{lp}(u,v)\,F(u,v) \quad (6)$$

$$Unsharp\ Masking\ H_{hp}(u,v) = 1 - H_{lp}(u,v) \quad (7)$$

$$High-boost\ filtering\ H_{hb}(u,v) = (A-1) + H_{hp}(u,v) \quad (8)$$

$$High-frequency\ emphasis\ H_{hfe}(u,v) = a + bH_{hp}(u,v)\ \ where\ \ (a \geq 0,\ b > a) \quad (9)$$

The process consists of multiplying the composite filter by the (centered) transform of the input image and then taking the inverse transform of the product. Multiplication of the real part of this result by $(-1)^{x+y}$ gives us the high-boost filtered image $f_{hb}(x,y)$ in the spatial domain.

4.6. Laplacian in the Frequency Domain

$$\Im\left[\frac{d^n f(x)}{dx^n}\right] = (ju)^n F(u) \quad \rightarrow$$

$$\Im\left[\frac{d^2 f(x,y)}{dx^2} + \frac{d^2 f(x,y)}{dy^2}\right] = (ju)^2 F(u,v) + (jv)^2 F(u,v) = -(u^2+v^2)F(u,v)$$

Hence, a Laplacian filter in the Fourier domain is described as:

$$H(u,v) = -(u^2+v^2)$$

Or, when shifted:

$$H(u,v) = -((u-M/2)^2 + (v-N/2)^2)$$

$$\nabla^2 f(x,y) = F^{-1}\left\{-\left[\left(u-\frac{M}{2}\right)^2 + \left(v-\frac{N}{2}\right)^2\right] F(u,v)\right\} \quad (1)$$

$$g(x,y) = f(x,y) - \nabla^2 f(x,y) \quad (2)$$

$$g(x,y) = F^{-1}\left\{\left[1 - \left(\left(u-\frac{M}{2}\right)^2 + \left(v-\frac{N}{2}\right)^2\right)\right] F(u,v)\right\} \quad (3)$$

Equation (1) gives the laplacian result whereas equation (2) and (3) gives the enhanced image after taking the laplacian.

4.7. Fast Fourier Transform

The reason that Fourier based techniques have become so popular is the development of

the *Fast Fourier Transform (FFT)* algorithm. It allows the Fourier transform to be carried out in a reasonable amount of time and reduces the amount of time required to perform a Fourier transform by a factor of 100 – 600 times! Note: DFT takes $O(M^2)$ to compute.

If we let

$$W_M = e^{-i2\pi/M}$$

the Discrete Fourier Transform can be written as

$$F(u) = \frac{1}{M} \sum_{x=0}^{M-1} f(x)\ {W_M}^{ux}$$

If M is a multiple of 2, $M = 2K$ for some positive number K.

Substituting $2K$ for M gives

$$F(u) = \frac{1}{2K} \sum_{x=0}^{2K-1} f(x)\ {W_{2K}}^{ux}$$

Separating out the K even terms and the K odd terms,

$$F(u) = \frac{1}{2}\left\{\frac{1}{K}\sum_{x=0}^{K-1} f(2x){W_{2K}}^{u(2x)} + \frac{1}{K}\sum_{x=0}^{K-1} f(2x+1){W_{2K}}^{u(2x+1)}\right\}$$

Notice that

$$W_{2K}^{2u} = W_K^u$$

$$W_{2K}^{2(u+1)} = W_{2K}^{2u}\ W_{2K}^u = W_K^u W_{2K}^u$$

So,

$$F(u) = \frac{1}{2}\left\{\frac{1}{K}\sum_{x=0}^{K-1} f(2x)W_K^{ux} + \frac{1}{K}\sum_{x=0}^{K-1} f(2x+1)W_K^{ux}W_{2K}^u\right\}$$

$$F(u) = \frac{1}{2}\left\{\frac{1}{K}\sum_{x=0}^{K-1} f(2x)W_K^{ux} + \frac{1}{K}\sum_{x=0}^{K-1} f(2x+1)W_K^{ux}W_{2K}^u\right\}$$

$$F(u) = \frac{1}{2}\left\{F_{\text{even}}(u) + F_{\text{odd}}(u)W_{2K}^u\right\}$$

Use this to get the first K terms ($u = 0..K-1$), then re-use these parts to get the last K terms ($u = K..2K-1$):

$$F(u+K) = \frac{1}{2}\left\{F_{\text{even}}(u) - F_{\text{odd}}(u)W_{2K}^u\right\}$$

Only have to do $u = 0..M/2-1$, $x = 0..M/2-1$ and a little extra math.

To derive $F(u+K)$ *use* $W_K^{u+K} = W_K^u$ *and* $W_{2K}^{u+K} = -W_{2K}^u$

Computational Complexity:

Discrete Fourier Transform	$O(N^2)$
Fast Fourier Transform	$O(N \log N)$

Remember:
The Fast Fourier Transform is just a faster *algorithm* for computing the Discrete Fourier Transform—it is *not* any different in result.

What About 2-D?

Remember, we can use separability of the Fourier Transform to break a 2-D transform into $2N$ 1-D transforms:

DFT	$O(N^4)$
DFT using separability	$O(N^3)$
FFT using separability	$O(N^2 \log N)$

4.8. Frequency and Spatial Domain Filtering

Similar jobs can be done in the spatial and frequency domains. Filtering in the spatial domain can be easier to understand. Filtering in the frequency domain can be much faster – especially for large images

4.9. Convolution and Padding

Convolution theorem†

$$f(x, y) * h(x, y) \Leftrightarrow F(u, v)H(u, v);$$
$$f(x, y)h(x, y) \Leftrightarrow F(u, v) * H(u, v)$$

Hence, we can perform the convolution as a product in the Fourier domain, and then take the inverse transform.

Note:
for finite-length sequences the product of the **DFT**s is used, which is equivalent to **circular** convolution. Zero padding is required to make it equivalent to **linear** convolution.

In other words:
the Fourier transform assumes that the *f* and *h* functions are **periodic**, with period equal to their length. This should be taken into account, otherwise a wrong result is obtained. The error source is already visible in the data domain:

$$f_e(x) = \begin{cases} f(x) & 0 \le x \le A - 1 \\ 0 & A \le x \le P \end{cases}$$

and

$$g_e(x) = \begin{cases} g(x) & 0 \le x \le B - 1 \\ 0 & B \le x \le P. \end{cases}$$

$$P \geq A + B - 1$$

$$f_e(x,y) = \begin{cases} f(x,y) & 0 \leq x \leq A-1 \text{ and } 0 \leq y \leq B-1 \\ 0 & A \leq x \leq P \text{ or } B \leq y \leq Q \end{cases}$$

$$P \geq A + C - 1$$

and

and

$$h_e(x,y) = \begin{cases} h(x,y) & 0 \leq x \leq C-1 \text{ and } 0 \leq y \leq D-1 \\ 0 & C \leq x \leq P \text{ or } D \leq y \leq Q \end{cases}$$

$$Q \geq B + D - 1.$$

2-D Convolution

$$f(x,y) * h(x,y) = \frac{1}{MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(m,n) h(x-m, y-n).$$

$$f(x,y) * h(x,y) \Leftrightarrow F(u,v)H(u,v)$$

and

$$f(x,y)h(x,y) \Leftrightarrow F(u,v) * H(u,v).$$

2-D Correlation

The correlation of two functions $f(x,y)$ and $h(x,y)$ is defined as

$$f(x,y) \circ h(x,y) = \frac{1}{MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f^*(m,n) h(x+m, y+n)$$

$$f(x,y) \circ h(x,y) \Leftrightarrow F^*(u,v)H(u,v)$$

$$f^*(x,y)h(x,y) \Leftrightarrow F(u,v) \circ H(u,v)$$

5. Image Compression

Objective: To reduce the amount of data required to represent an image. Important in data storage and transmission

- Progressive transmission of images (internet, www)
- Video coding (DHTV, Teleconferencing)
- Digital libraries and image databases (Medical Imaging, Satellite Images)

There are two compression techniques

- Loss-less (Information Preserving)

Images can be compressed and restored without any loss of information. Application: Medical Images, GIS

- Lossy

Perfect recovery is not possible but provides a large data compression. Example: TV Signals, Teleconferencing

5.1. Data Compression

It refers to the process of reducing the amount of data required to represent a given quantity of information. Data are the means where information is conveyed. Relative data redundancy R_D: $R_D = 1 - {}^{1}/_{C_R}$ *where* C_R is the compression ratio; $C_R = {}^{n_1}/_{n_2}$

$n_1 and\ n_2$denote the number of information carrying units.

Three basic data redundancy can be identified and exploited

- Coding redundancy: to reduce the amount of data representing the same information
- Interpixel redundancy: includes spatial redundancy, geometric redundancy and interframe redundancy
- Psychovisual redundancy: certain information has less relative importance in normal visual processing and can be eliminated.

5.1.1. Coding Redundancy

Assume a discrete random variable r_k in the interval [0.1] represents the gray levels of an image and that each r_k occurs with probability $p_r(r_k) = \frac{n_k}{n}; k = 0,1, \dots, L-1$

And $L_{avg} = \sum_{k=0}^{L-1} l(r_k)P_r(r_k)$is the average number of bits to represent each pixel, where $l(r_k)$is the number of bits required for each value r_k.

r_k	$p_r(r_k)$	Code 1	$l_1(r_k)$	Code 2	$l_2(r_k)$
$r_0 = 0$	0.19	000	3	11	2
$r_1 = 1/7$	0.25	001	3	01	2
$r_2 = 2/7$	0.21	010	3	10	2
$r_3 = 3/7$	0.16	011	3	001	3
$r_4 = 4/7$	0.08	100	3	0001	4
$r_5 = 5/7$	0.06	101	3	00001	5
$r_6 = 6/7$	0.03	110	3	000001	6
$r_7 = 1$	0.02	111	3	000000	6

TABLE 8.1 Example of variable-length coding.

$$L_{avg} = \sum_{k=0}^{7} l_2(r_k)p_r(r_k)$$
$$= 2(0.19) + 2(0.25) + 2(0.21) + 3(0.16) + 4(0.08)$$
$$+ 5(0.06) + 6(0.03) + 6(0.02)$$
$$= 2.7 \text{ bits},$$
$$R_D = 1 - {}^{1}/_{1.11} = 0.099$$

r_k	$p_r(r_k)$	Code 1	$l_1(r_k)$	Code 2	$l_2(r_k)$
$r_0 = 0$	0.19	000	3	11	2
$r_1 = 1/7$	0.25	001	3	01	2
$r_2 = 2/7$	0.21	010	3	10	2
$r_3 = 3/7$	0.16	011	3	001	3
$r_4 = 4/7$	0.08	100	3	0001	4
$r_5 = 5/7$	0.06	101	3	00001	5
$r_6 = 6/7$	0.03	110	3	000001	6
$r_7 = 1$	0.02	111	3	000000	6

L_{avg} 3 bits/symbol

L_{avg} 2.7 bits/symbol

Concept: Assign the longest code word to the symbol with the least probability of occurrence.

5.1.2. Huffman Coding

Huffman code: Consider a 6 symbol source

	a_1	a_2	a_3	a_4	a_5	a_6
$p(a_i)$	0.1	0.4	0.06	0.1	0.04	0.3

- Huffman coding: give the smallest possible number of code symbols per source symbols.

Step 1: Source reduction

Original source		Source reduction			
Symbol	Probability	1	2	3	4
a_2	0.4	0.4	0.4	0.4	0.6
a_6	0.3	0.3	0.3	0.3	0.4
a_1	0.1	0.1	0.2	0.3	
a_4	0.1	0.1	0.1		
a_3	0.06	0.1			
a_5	0.04				

Step 2: Code assignment procedure

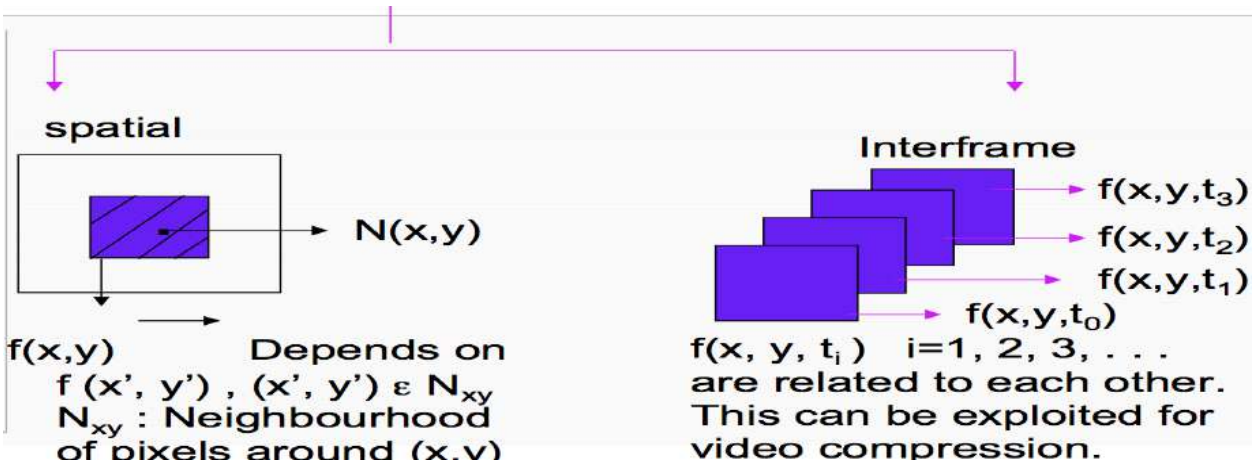
Original source			Source reduction							
Sym.	Prob.	Code	1		2		3		4	
a_2	0.4	1	0.4	1	0.4	1	0.4	1	0.6	0
a_6	0.3	00	0.3	00	0.3	00	0.3	00	0.4	1
a_1	0.1	011	0.1	011	0.2	010	0.3	01		
a_4	0.1	0100	0.1	0100	0.1	011				
a_3	0.06	01010	0.1	0101						
a_5	0.04	01011								

The code is instantaneous uniquely decodable without referencing succeeding symbols.

Average length:

(0.4) (1) + 0.3 (2) + 0.1 X 3 + 0.1 X 4 + (0.06 + 0.04) 5 = 2.2 bits/symbol

5.1.3. Interpixel/Interframe Redundancy



Interpixel Redundancy: Parts of an image are highly correlated. In other words, we can predict a given pixel from its neighbor.

5.1.4. Psychovisual Redundancy

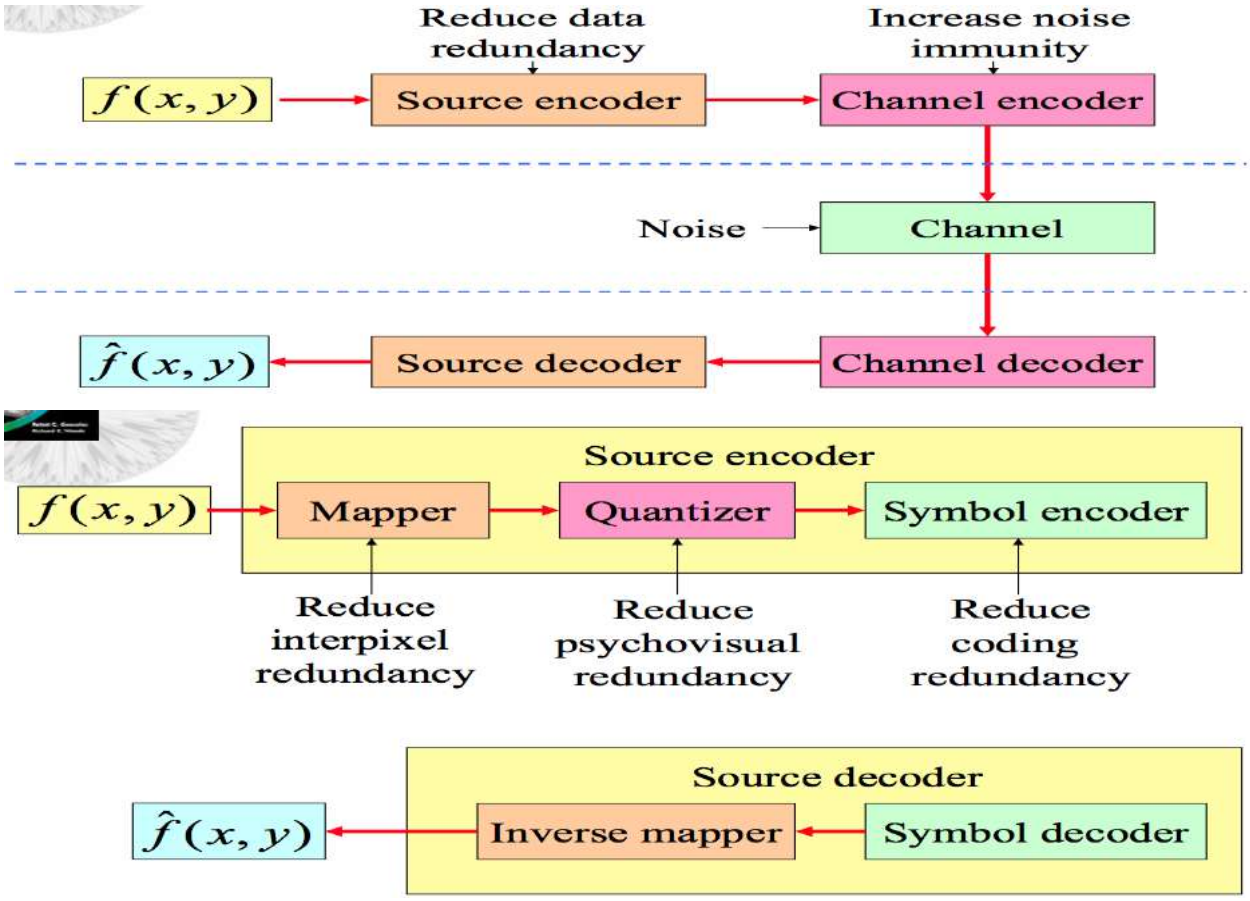
Certain information simply has less relative importance than other information in normal visual processing. This information is said to be psychovisually redundant. It can be eliminated without significantly impairing the quality of image perception.

Pixel	Gray Level	Sum	IGS Code
$i-1$	N/A	0000 0000	N/A
i	0110 1100	0110 1100	0110
$i+1$	1000 1011	1001 0111	1001
$i+2$	1000 0111	1000 1110	1000
$i+3$	1111 0100	1111 0100	1111

TABLE 8.2 IGS quantization procedure.

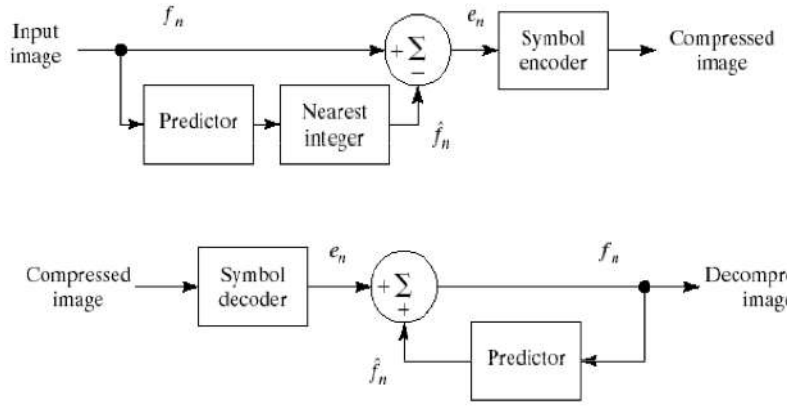
A sum-initially to zero-is first formed from the current 8-bit grey-level value and the four least significant bits of a previously generated sum. If the 4 most significant bits of the current value are 1111_2, however, 0000_2 is added instead. The 4 most significant bits of the resulting sum are used as the coded pixel value.

5.2. Image Compressing Models



5.2.1. Error-Free Compression (Lossless)

Error-free compression means lossless compression. In numerous applications error-free compression is the only acceptable means of data reduction. They normally provide compression ratios of 2 to 10.



Prediction error:

$e_n = f_n - \hat{f}_n$

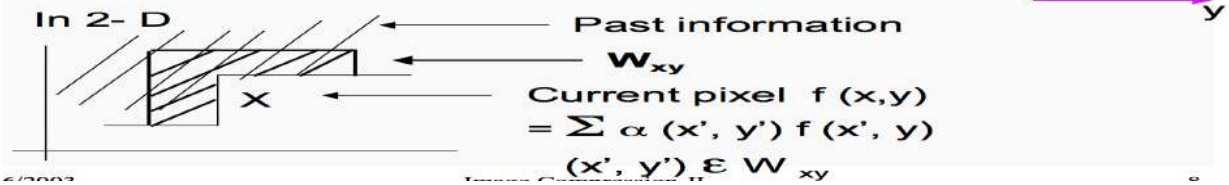
e_n is coded using a variable length code

$f_n = e_n + \hat{f}_n$

Example 1: $\hat{f}_n = \text{Int}\left(\sum_{i=1}^{m} \alpha_i f_{n-i}\right)$

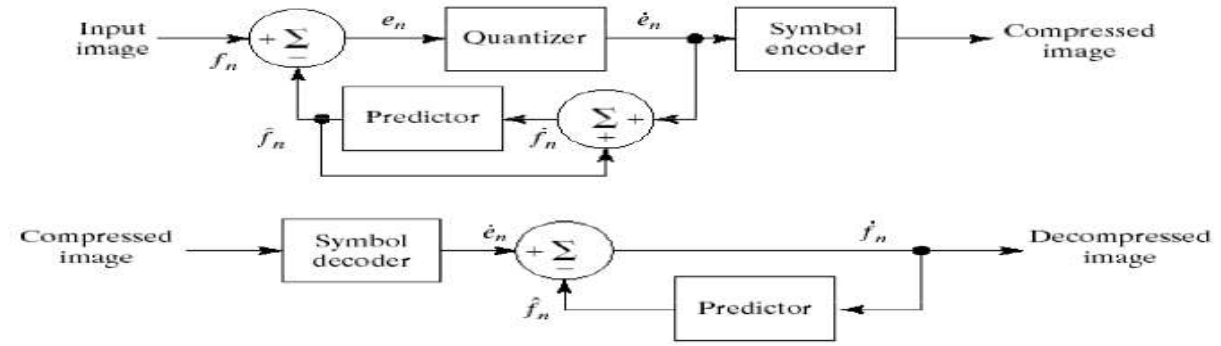
→ Linear predictor ; m = order of predictor

Example 2: $\hat{f}_n(x,y) = \text{Int}\left(\sum_{i=1}^{m} \alpha_i f(x, y-i)\right)$

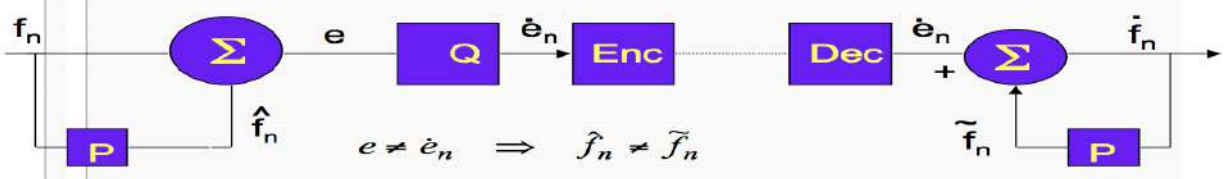


5.2.2. Lossy Compression

Unlike the error-free approaches, lossy encoding is based on the concept of compromising the accuracy of the reconstructed image in exchange for increased compression.



Lossy compression: uses a quantizer to compress further the number of bits required to encode the 'error'.
First consider this:



Notice that, unlike in the case of loss-less prediction, in lossy prediction the predictors P "see" different inputs at the encoder and decoder

This results in a gradual buildup of error which is due to the quantization error at the encoder site.
In order to minimize this buildup of error due to quantization we should ensure that 'Ps' have the same input in both the cases.

f_n	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$\hat{f}_n = f_{n-1}$															
$e_n =$	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
$\dot{e}_n =$	0	2	2	2	2	.	.	.	.	.					
$\dot{f}_n =$	0	2	4	6	8	10	.	.	.	.	.	.			

Example:

$$\hat{f}_n = \alpha \dot{f}_{n-1}$$

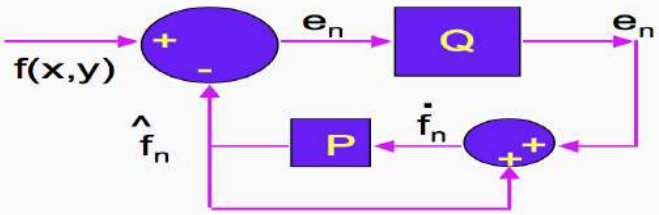
$$\text{and } \dot{e}_n = \begin{cases} +\xi & e_n > 0 \\ -\xi & e_n < 0 \end{cases}$$

$0 < \alpha < 1$
prediction coefficient

$$\dot{f}_n = \dot{e}_n + \hat{f}_n$$
$$= \dot{e}_n + \alpha \dot{f}_{n-1}$$

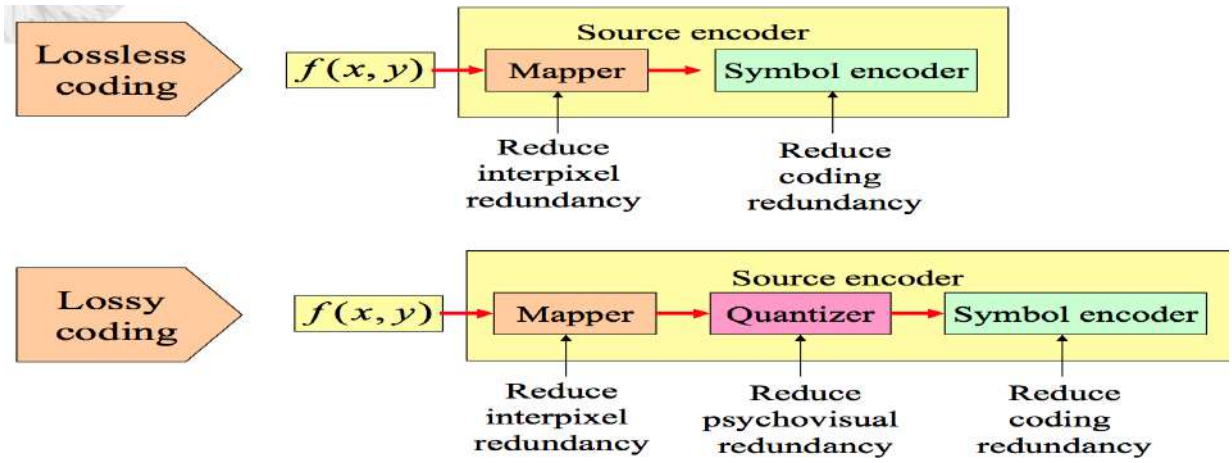
with feedback

$f_n =$	0	1	2	3	4
$e_n =$	_	1	2	1	2
$\dot{e}_n =$	_	0	2	0	2
$\dot{f}_n =$	0	0	2	2	4
$\hat{f}_n =$	_	0	0	2	2

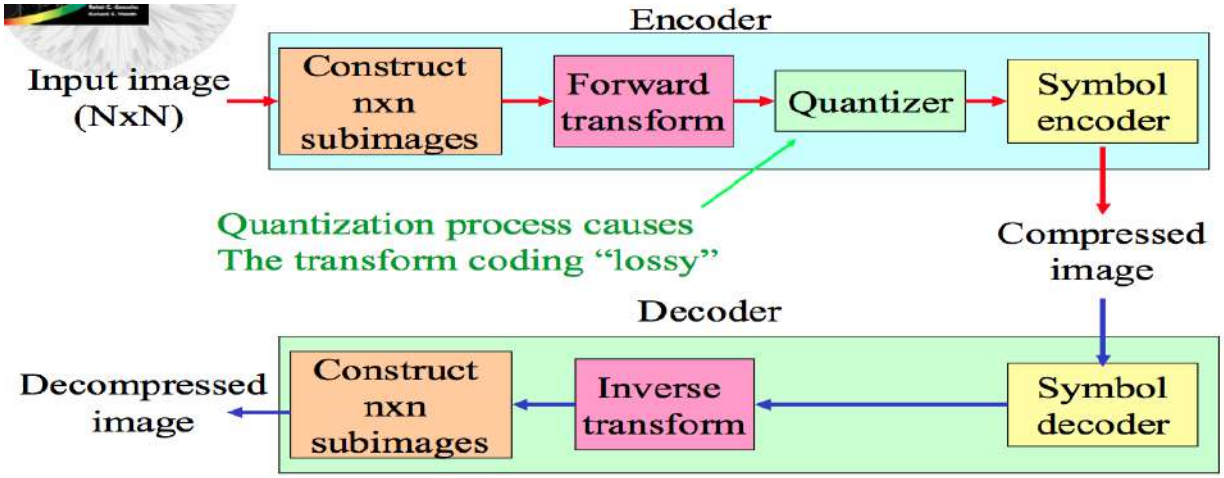


Note: The quantizer used here is-- floor $(e_n/2)*2$. This is different from the one used in the earlier example. Note that this would result in a worse response if used without Feedback (output will be flat at "0").

5.2.3. Lossless Vs. Lossy Coding



5.2.4. Transform Coding



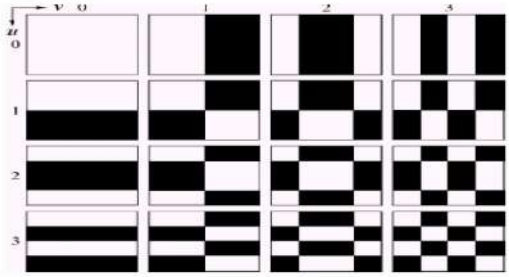
Examples of transformations used for image compression: DFT and DCT

Parameters that effect transform coding performance:

- Types of transformation
- Size of subimage
- Quantization algorithm

5.2.5. Hadamard Transform

$$g(x,y,u,v) = h(u,v,x,y) = \frac{1}{N}(-1)^{\sum_{i=0}^{m-1}\left[b_i(x)p_i(u)+b_i(y)p_i(v)\right]}$$



N = 4

$N = 2^m$

$b_k(z)$ = the k^{th} bit of z

$p_0(u) = b_{m-1}(u)$

$p_1(u) = b_{m-1}(u) + b_{m-2}(u)$

$p_2(u) = b_{m-2}(u) + b_{m-3}(u)$

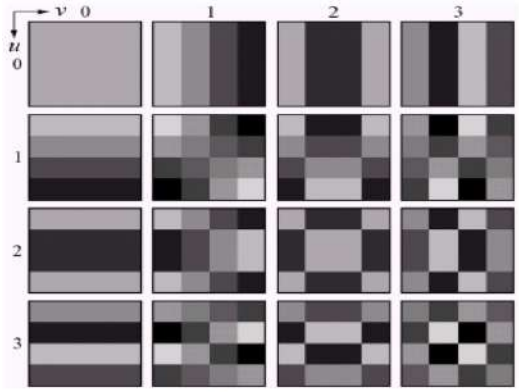
...

$p_{m-1}(u) = b_1(u) + b_0(u)$

Advantage: simple, easy to implement
Disadvantage: not good packing ability

5.2.6. Discrete Cosine Transform

$$g(x,y,u,v) = h(u,v,x,y) = \alpha(u)\alpha(v)\cos\left[\frac{(2x+1)u\pi}{2N}\right]\cos\left[\frac{(2y+1)v\pi}{2N}\right]$$



N = 4

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

DCT is one of the most frequently used transform for image compression. For example, DCT is used in JPG files.

Advantage: good packing ability, modulate computational complexity

6. Image Segmentation: Thresholding

Segmentation subdivides an image into constituent regions or objects. The level to which the subdivision is carried depends on the problem being solved. That is, segmentation should stop when the objects of interest in an application have been isolated. For example, in the automated inspection of electronic assemblies, interest lies in analyzing images of the products with the objective of determining the presence or absence of specific anomalies, such as missing components or broken connection paths. There is no point in carrying segmentation past the level of detail required to identify those elements.

Segmentation of non-trival images is one of the most difficult tasks in image processing. Segmentation accuracy determines the eventual success or failure of computerized analysis procedures. For this reason, considerable care should be taken to improve the probability of rugged segmentation.

Image segmentation algorithms generally are based on one of two basic properties of intensity values: *discontinuity and similarity.* In the first category, the approach is to partition an image based on abrupt changes in intensity, such as edges in an image. The principal approaches in the second category are based on partitioning an image into regions that are similar according to a set of pre-defined criteria. Thresholding, region growing, and region splitting and merging are examples of methods in this category.

Segmentation attempts to partition the pixels of an image into groups that strongly correlate with the objects in an image. The principal areas of interest are:

- Detection of isolated points
- Detection of lines and edges in an image

For similarity, the principal approaches are based on Thresholding, region growing, region splitting and merging.

Using discontinuity and similarity of gray-level pixel values is applicable to both static and dynamic images. For dynamic images, the concept of motion can be exploited in the segmentation process.

6.1. Detection of Discontinuities

There are three basic types of grey level discontinuities that we tend to look for in digital images:

- Points
- Lines
- Edges

Detecting discontinuities (points, lines and edges) is generally accomplished by mask processing. The response equation:

$R = w_1 z_1 + w_2 z_2 + \cdots + w_9 z_9 = \sum_{i=1}^{9} w_i z_i$

where z_i is the gray level of the pixel associated with mask coefficient w_i. As usual the response of the mask is defined with respect to its center location.

6.1.1. Point Detection

The detection of isolated points in an image is straight forward in principle. Using the mask shown below we say that a point has been detected at the location on which the mask is centered if $|R| \geq T$; where T is a non-

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

negative threshold.

An isolated point is detected if the response of the mask is greater than a predetermined threshold: |R|>T. This measures the weighted difference between a center point and its neighbors. The mask is the same as the high frequency filtering mask. The emphasis here is on the detection of points: only differences that are large enough to be considered isolated points in an image are of interest.

6.1.2. Line Detection

The next level of complexity is to try to detect lines. The masks below will extract lines that are one pixel thick and running in a particular direction.

−1	−1	−1	−1	−1	2	−1	2	−1	2	−1	−1
2	2	2	−1	2	−1	−1	2	−1	−1	2	−1
−1	−1	−1	2	−1	−1	−1	2	−1	−1	−1	2
Horizontal			+45°			Vertical			−45°		

With a constant background, the maximum response occurs when the line is "lined up" with the center of the mask. Note that the preferred direction of each mask is weighted with a larger coefficient than other possible directions. Let $R_1, R_2, R_3 and\ R_4$denote the response of the masks. If, at a certain point in the image, $|R_i| > |R_j| for\ all\ j \neq i$; that point is said to be more likely associated with a line in the direction of mask i.

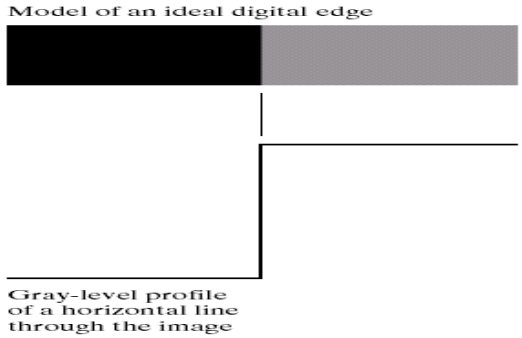
6.1.3. Edge Detection

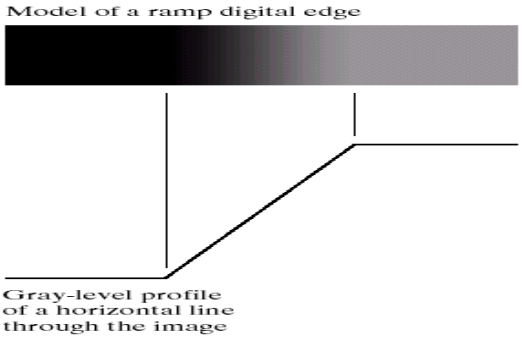
Edge detection is by far the most common approach for detecting discontinuities in gray levels. Isolated points and 1-pixel thin lines are not common in most practical applications.

Basic formulation and initial assumptions

- An edge is a boundary between two regions with relatively distinct gray-level properties
- Regions are sufficiently homogeneous so that the transition between the regions can be determined on the basis of gray-level discontinuities alone
- If this is not valid, some other techniques will be used

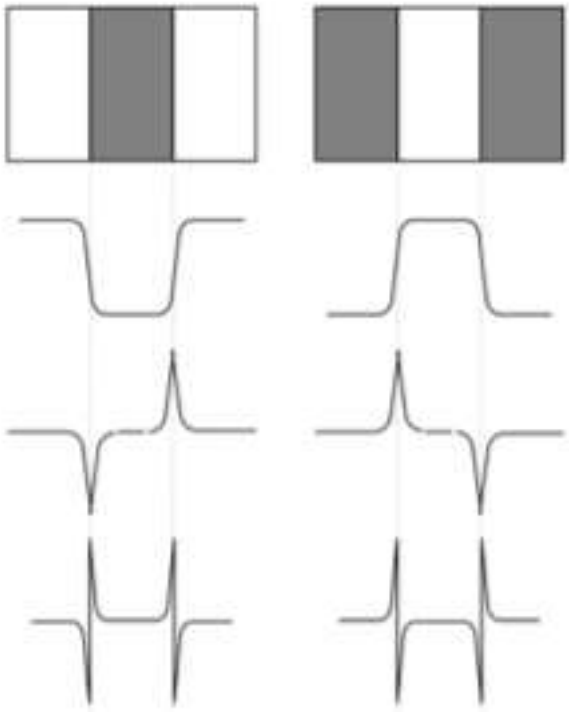
The basic idea behind most edge detection techniques is the computation of a local derivative operator. An edge is a set of connected pixels that lie on the boundary between two regions.





6.1.4. Derivative Operators

- An image of a dark stripe on a light background (and vice-vesa)
- A profile of the lines in the image (modeled as a gradual rather than sharp transition). Edges in images tend to be slightly blurred as a result of sampling
- The first derivative: the magnitude detects the presence of an edge
- The second derivative: the sign tells the type of transition (light-to-dark or dark-to-light). Note also the presence of a Zero-crossing at each edge



The first derivative at any point in an image is computed using the magnitude of the gradient

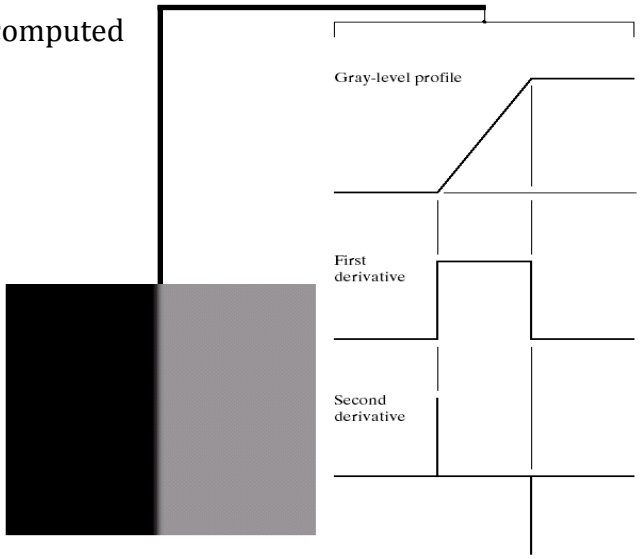
$$\nabla f = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

The magnitude of this vector is given by:

$$\nabla f = mag(\nabla f) = \sqrt{G_x^2 + G_x^2}$$

$$\nabla f = |G_x| + |G_y|$$

The direction of the gradient vector is the angle $\alpha(x, y) = \tan^{-1}\left(\frac{G_y}{G_x}\right)$



1st derivative tells us where an edge is and 2nd derivative can be used to show edge direction. However, derivative based edge detectors are extremely sensitive to noise.

Common Edge Detectors

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

−1	0	0	−1
0	1	1	0

Roberts

−1	−1	−1	−1	0	1
0	0	0	−1	0	1
1	1	1	−1	0	1

Prewitt

−1	−2	−1	−1	0	1
0	0	0	−2	0	2
1	2	1	−1	0	1

Sobel

Often, problems arise in edge detection in that there are is too much detai. One way to overcome this is to smooth images prior to edge detection.

6.1.5. Laplacian Edge Detection

The Laplacian is typically not used by itself as it is too sensitive to noise.

0	−1	0
−1	4	−1
0	−1	0

−1	−1	−1
−1	8	−1
−1	−1	−1

Usually when used for edge detection the Laplacian is combined with a smoothing Gaussian filter.

Although the Laplacian responds to changes in intensity, it is seldom used in edge detection for several reasons.

- As a second derivative operator it is typically unacceptably sensitive to noise
- The Laplacian produces double edges
- Unable to detect direction

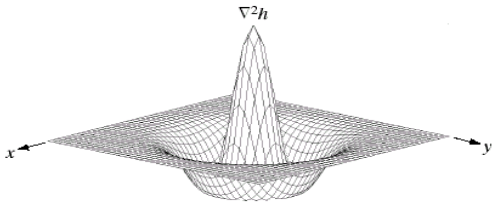
As such, the Laplacian is used in the secondary role of detector for establishing whether a pixel is on the light or dark side of an edge. A more general use of the Laplacian is to find the location of edges using the zero-crossing property. Basic idea is to convolve an image with the Laplacian of a 2-D Gaussian function of the form

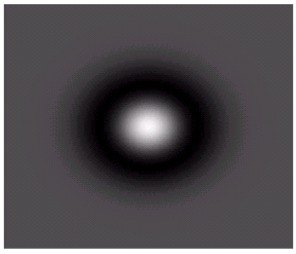
$$h(x,y) = \exp\left(-\frac{x^2+y^2}{2\sigma^2}\right); where\ \sigma = standard\ deviation$$

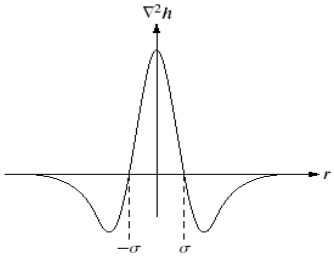
if $r^2 = x^2 + y^2$, then the Laplacian is then

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

The Laplacian of Gaussian (or Mexican hat) filter uses the Gaussian for noise removal and the Laplacian for edge detection.







0	0	−1	0	0
0	−1	−2	−1	0
−1	−2	16	−2	−1
0	−1	−2	−1	0
0	0	−1	0	0

6.2. Edge Linking and Boundary Detection

Ideally, edge detection techniques yield pixels lying only on the boundaries between regions. In practice, this pixel set seldom characterizes a boundary completely because of

- Noise
- Breaks in the boundary due to non-uniform illumination

- Other effects that introduces spurious discontinuities

Thus, edge detection algorithms are usually followed by linking and other boundary detection procedures designed to assemble edge pixels into meaningful boundaries.

6.2.1. Edge Linking: Local Precessing

Basic Idea:

Analyze the characteristics of pixels in a small neighborhood (3*3, 5*5, etc) for every point (x,y) that has undergone edge detection. All points that are "similar" are linked, forming a boundary of pixels that share some common property.

Two principal properties for establishing similarity:

The strength of the response of the gradient operator used to produce the edge pixels. The direction of the gradient

$$\alpha(x,y) = \tan^{-1}\left(\frac{G_y}{G_x}\right)$$

- An edge pixel at (x',y') in the neighborhood centered at (x,y) is similar in magnitude to the pixel at (x,y) if

$$\left|\nabla f(x,y) - \nabla f(x',y')\right| \le T$$

- where T is a predetermined threshold
- An edge pixel at (x',y') in the neighborhood centered at (x,y) is similar in angle to the pixel at (x,y) if

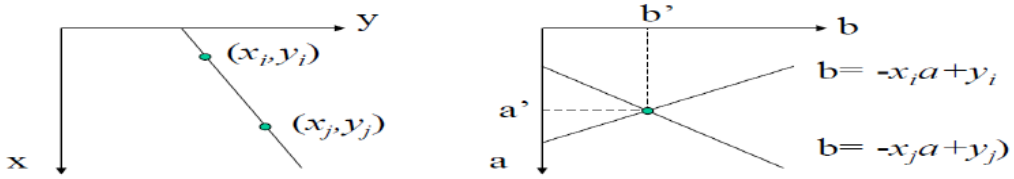
$$\left|\alpha(x,y) - \alpha(x',y')\right| \le A$$

- where A is a predetermined angle threshold
- A point in the neighborhood of (x,y) is linked to (x,y) if both magnitude and angle criteria are satisfied

6.2.2. Global Processing: Hough Transform

Now lets consider global relationships between pixels. Suppose that, for n points in an image, we want to find all subsets of these points that lie on straight lines. Consider a point (x_i, y_i) and the equation for a straight-line $y_i = ax_i + b$. Infinitely many lines pass through the point (x_i, y_i), all satisfying the equation for varying values of a and b. However, writing the equation as $b = y_i - ax_i$ and considering the ab plane (also called the parameter space) yields the equation of a single line for a fixed points (x_i, y_i).

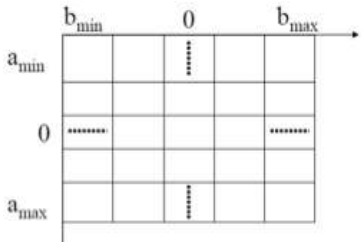
- A second point (x_j, y_j) also has a line in the parameter space associated with it
- This line intersects the line associated with (x_i, y_i) at (a', b')
  - a' is the slope and b' is the intercept of the line containing both (x_i, y_i) and (x_j, y_j) in the xy plane
- In fact, all points that lie on this line have corresponding lines in the parameter space that intersect at (a', b')



Hough Transform: Accumulator Cells

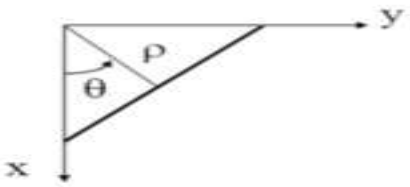
The computational attractiveness of the Hough transform arises from the subdivision of the parameter space into accumulator cells. (a_{min}, a_{max}) *and* (b_{min}, b_{max}) are expected ranges of slope and intercept values

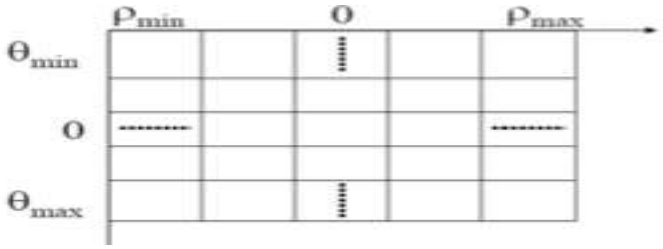
- The cell at coordinates (i,j), with cell value A(i,j), corr square associated with parameter space coordinate
- The collection of accumulator cells is commonly call (or Hough array) and are computed as
  1. Set all cells to zero.
  2. For every point (x_k,y_k) in the image plane, let *a* equal e subdivision values on the *a* axis and solve for *b* using
  3. The resulting b's are rounded off to the nearest allowe
  4. If a choice of a_p results in solution b_q, we let A(p,q)=A(
  5. At the end of the procedure, a value of M in A(i,j) corre the xy plane lying on the line $y=a_ix+b_j$
- The accuracy of the collinearity of these points is de number of subdivisions in the *ab* plane



A problem with this representation is that the slope and intercept approach infinity as the line approaches the vertical. Solution: use the normal representation of a line given by

$$x\cos\theta + y\sin\theta = \rho$$





Hough Transform

- Instead of straight lines in the *ab* plane, we now have sinusoidal curves in the $\rho\theta$ plane
- M collinear points lying on the line

$$x\cos\theta_j + y\sin\theta_j = \rho_i$$

- yields M sinusoidal curves that intersect at (ρ_i,θ_j) in the parameter space
- The range of θ is ±90°, measured with respect to the x axis
  - A horizontal line has θ=0°, with ρ equal to the positive x intercept
  - A vertical line has θ=+90°, with ρ equal to the positive y intercept or θ=-90°, with ρ equal to the negative y intercept
- The range of ρ is $\pm(2)^{½}D$. Where D is the distance between corners in the image

Procedure:

1. Compute the gradient of an image and threshold it to obtain a binary image.
2. Specify subdivisions in the $\rho\theta$-plane
3. Examine the counts of the accumulator cells for high pixel concentrations
4. Examine the relationship (principally for continuity) between pixels in a chosen cell.

6.3. Thresholding

Thresholding is usually the first step in any segmentation approach. Single value thresholding can be given mathematically as:

$$g(x,y) = \begin{cases} 1 \, if \; f(x,y) > T \\ 0 \, if \; f(x,y) \le T \end{cases}$$

Based on the histogram of an image partition the image histogram using a single global threshold. The success of this technique very strongly depends on how well the histogram can be partitioned.

6.3.1. Global Thresholding Algorithm

The basic global threshold, T, is calculated

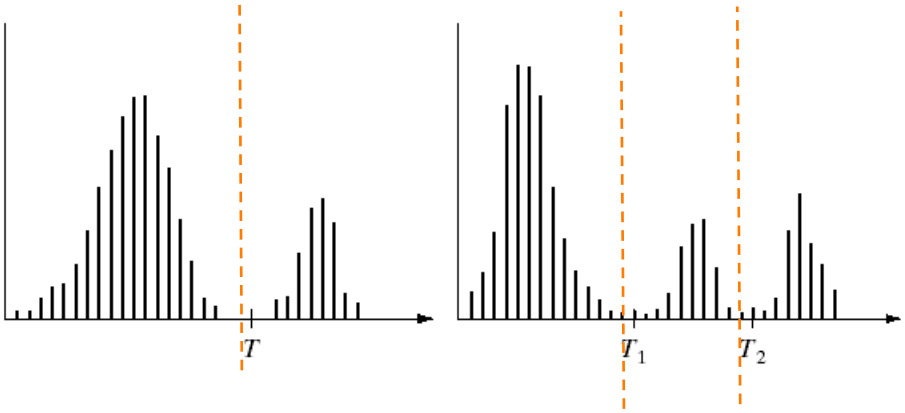
as follows:

1. Select an initial estimate for T (typically the average grey level in the image)
2. Segment the image using T to produce two groups of pixels: G_1 consisting of pixels with grey levels >T and G_2 consisting pixels with grey levels ≤ T
3. Compute the average grey levels of pixels in G_1 to give μ_1 and G_2 to give μ_2
4. Compute a new threshold value:
5. Repeat steps 2 – 4 until the difference in T in successive iterations is less than a predefined limit T_∞

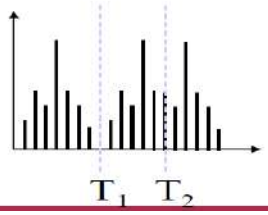
This algorithm works very well for finding thresholds when the histogram is suitable

Problem with Single Value Thresholding

Single value thresholding only works for bimodal histograms. Also, Uneven illumination can really upset a single valued thresholding scheme. Images with other kinds of histograms need more than a single threshold.



- Suppose several objects with differing gray levels (with a dark background) comprise the image
- An object may be classified as belonging to one object class if $T_1 < f(x,y) \leq T_2$, to a second class if $f(x,y) > T_2$ or to the background if $f(x,y) \leq T_1$
- This, however, is generally less reliable than single level thresholding



- Thresholding may be viewed as an operation that tests against a given function of the form

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

- where $f(x,y)$ is the gray level of point (x,y) and $p(x,y)$ is some local property of the point -- the average gray level of a neighborhood around (x,y)
- The thresholded image is given by

$$g(x,y) = \begin{cases} 1 \text{ if } f(x,y) > T \\ 0 \text{ if } f(x,y) \leq T \end{cases}$$

- Pixels labeled 1 (or any other convenient gray level value) correspond to objects
- When T depends only on $f(x,y)$ the threshold is called *global*
- If T depends on $f(x,y)$ and $p(x,y)$ the threshold is *local*
- If, in addition, T depends on the spatial coordinates (x,y), the threshold is called *dynamic*
- For example, a local threshold may be used if certain information about the nature of the objects in the image is known a priori
- A dynamic threshold may be used in the case where object illumination is non-uniform

6.3.2. Basic Adaptive Thresholding

An approach to handling situations in which single value thresholding will not work is to divide an image into sub images and threshold these individually. Since the threshold for each pixel depends on its location within an image this technique is said to *adaptive*.

6.3.3. Thresholding Based on Boundaries

Important aspect of threshold selection: the ability to reliably identify mode peaks in a given histogram. The chances of selecting a good "good" threshold are enhanced if mode peaks are: tall, narrow, symmetric, and separated by deep valleys.

One approach for improving the histogram shape is to consider only those pixels that lie on or near a boundary between objects and the background. An obvious advantage is that

the histogram becomes less dependent on the size of objects in the image. By choosing pixels on or near object boundaries (assuming an equal probability of choosing a pixel on the object or boundary) the histogram peaks tend to be made more symmetric. Using pixels that satisfy some simple measures based on the gradient and Laplacian operators tends to deepen the valleys between histogram peaks.

Determining if a pixel lies on boundary: compute the gradient.

Determining what side, background (dark) or object (light), a pixel lies on: compute the Laplacian.

Using the gradient and Laplacian, a three-level image may be formed according to

$$s(x,y) = \begin{cases} 0 \; if \; \nabla f < T \\ +if \; \nabla f \geq T \; and \; \nabla^2 f \geq 0 \\ -if \; \nabla f \geq T \; and \; \nabla^2 f < 0 \end{cases}$$

- For a dark object on a light background, s(x,y) is produced where
  - all pixels not on an edge are labeled 0
  - all pixels on the dark side of an edge are labeled +
  - all pixels on the light side of an edge are labeled -
- For a light object on a dark background, s(x,y) is produced where
  - all pixels not on an edge are labeled 0
  - all pixels on the dark side of an edge are labeled -
  - all pixels on the light side of an edge are labeled +

The information obtained with this procedure can be used to generate a segmented, binary image in which 1's correspond to objects of interest and 0's correspond to the background. The transition (along a horizontal or vertical scan line) from a light background to a dark object must be characterized by the occurrence of a − followed by a + in $s(x, y)$. The interior of the object is composed of pixels that are labeled either 0 or +. Finally, the transition from the object back to the background is characterized by the occurrence of a + followed by a −. Thus a horizontal or vertical scan line containing a section of an object has the following structure:

$$(\cdots)(-, +)(0 \text{ or } +)(+, -)(\cdots)$$

6.4. Region Based Segmentation

Edges and thresholds sometimes do not give good results for segmentation. Region-based segmentation is based on the connectivity of similar pixels in a region.

- Each region must be uniform.
- Connectivity of the pixels within the region is very important.

There are two main approaches to region-based segmentation: *region growing* and *region splitting.*

Let R represent the entire image region.

Segmentation is a process that partitions R into subregions, $R_1, R_2, \ldots, R_n$, such that

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

(b) R_i is a connected region, $i = 1,2,\ldots,n$

(c) $R_i \cap R_j = \phi$ for all i and $j, i \neq j$

(d) $P(R_i)$ = TRUE for $i = 1,2,\ldots,n$

(e) $P(R_i \cup R_j)$ = FALSE for any adjacent regions R_i and R_j

where $P(R_k)$: a logical predicate defined over the points in set R_k

For example: $P(R_k)$=TRUE if all pixels in R_k have the same gray level.

- Condition (a) indicates that the segmentation must be complete, i.e., every pixel must be in a region.
- Condition (b) requires that points in a region must be connected.
- Condition (c) indicates that the regions must be disjoint.
- Condition (d) deals with the properties that must be satisfied by the pixels in a segmented region – for example $P(R_i)$ = TRUE if all pixels in R_i have the same intensity.
- Condition (e) indicates that regions R_i and R_j are different in the sense of predicate P.

6.4.1. Region Growing

Groups' pixels or subregions into larger regions based on predefined criteria (gray tone or texture). Basic method:

Start with a set of "seed" points and from these grow regions by appending to each seed those neighboring pixels that have properties similar to the seed, such as specific ranges of gray level.

Problems in region growing:

- Selection of the seeds
- Criteria of similarity
- Gray level's similarity/ connectivity/ texture/ moments
- Formulation of a stopping rule
- Growing a region should stop when no more pixels satisfy the criteria for inclusion in that region.

Algorithm

1. Assume we find a good threshold, and use it to partition the regions into pure black n while
2. Use different labels to identify different objects
    a. Use region growing to connect parts that should have belong to the same region
    b. This is called "Connected component analysis"
    c. The region with the same label generate one segment

❖ **Seed Pixels**

Pixels of defective welds tend to have the maximum allowable digital value (255). All pixels having values of 255 are all selected as seed pixels.

❖ **Criteria for region growing**

1) The absolute gray-level difference between any pixel and the seed had to be less than 65. This is based on histogram (Difference between 255 and the location of the first major valley to the left.
2) To be included in one of the regions, the pixel had to be 8-connected to at least one pixel in that region. If a pixel was found to be connected to more than one region, the regions were merged.

6.4.2. Region Splitting and Merging

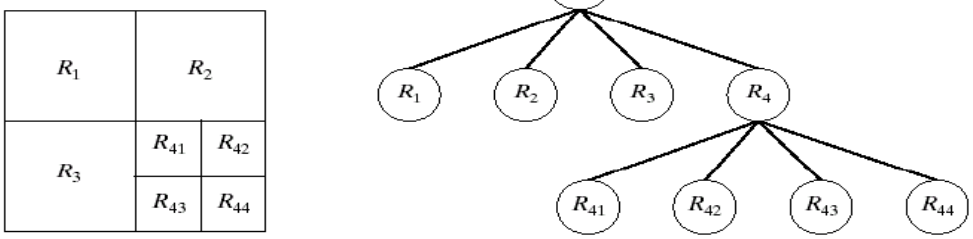
Region splitting is the opposite of region growing.

- First there is a large region (possible the entire image).
- Then a predicate (measurement) is used to determine if the region is uniform.
- If not, then the method requires that the region be split into two regions.
- Then each of these two regions is independently tested by the predicate (measurement).
- This procedure continues until all resulting regions are uniform

The main problem with region splitting is determining where to split a region. One method to divide a region is to use a quadtree structure. Quadtree: a tree in which nodes have exactly four descendants.

a b

FIGURE 10.42
(a) Partitioned image.
(b) Corresponding quadtree.



If only splitting, it is likely that adjacent regions have identical properties. Adjacent regions whose combined region satisfies P should be merged.

Algorithm

1. Splitting into 4 disjoint quadrants if P(R)=False
2. Merging any adjacent quadrants for which P is TRUE
3. Stop when no further merging and splitting is possible.

P(Ri)=TRUE if at least 80% of the pixels in Ri have the property

$$|Z_j - m_i| \leq 2\sigma_i$$

$Z_j \rightarrow is\ the\ pixel\ intensity\ value$;
$m_i \rightarrow is\ the\ Mean\ of\ the\ region;\ 2\sigma_i \rightarrow Standard\ deviation$

Can specify P(R) based on texture content for texture segmentation

The split and merge procedure:

- Split into four disjoint quadrants any region R_i for which $P(R_i)$ = FALSE.
- Merge any adjacent regions R_j and R_k for which $P(R_j U R_k)$ = TRUE. (the

quadtree structure may not be preserved)

– Stop when no further merging or splitting is possible.

7. Representation and Description

Images and segmented regions must be represented and described in a form suitable for further processing. The representations and the corresponding descriptions are selected according to the computational and semantic requirements of the image analysis task.

Representing a region involves two choices:

- External characteristics (boundary)
- Internal characteristics (pixels comprising the region)

An *external representation* is chosen when the primary focus is on shape characteristics. And an *internal representation* is selected when the primary focus is on regional properties, such as color and texture.

Description: E.g. a region may be represented by its boundary, and its boundary described by some feature such as length, regularity… Features should be insensitive to translation, rotation, and scaling. Both boundary and regional descriptors are often used together.

7.1. Chain Codes

Regions can be represented by their boundaries in a data structure instead of an image. The simplest form is just a linear list of the boundary points of each region. This method generally is unacceptable because:

- The resulting list tends to be quite long
- Any small disturbances along the boundary cause changes in the list that may not be related to the shape of the boundary

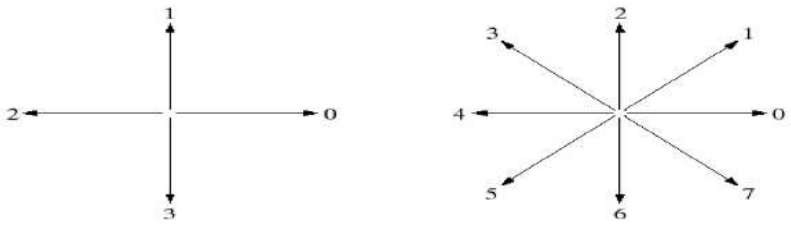
A variation of the list of points is the *chain code*, which encodes the information from the list of points at any desired quantization. Conceptually, a boundary to be encoded is overlaid on a square grid whose side length determines the resolution of the encoding. Starting at the beginning of the curve, the grid intersection points that come closest to it are used to define small line segments that join each grid point to one of its neighbor. The directions of these line segments are then encoded as small integers from zero to the number of neighbor used in encoding.

The chain code of a boundary depends on the starting point. However, the code can be normalized by treating it as a circular sequence. Size (scale) normalization can be achieved by altering the size of the sampling grid. Rotation normalization can be achieved by using the first difference of the chain code instead of the code itself.

- Count the number of directions changes (in counter clockwise direction) that separate two adjacent elements of the code (e.g. 10103322->3133030).

a b

FIGURE 11.1 Direction numbers for (a) 4-directional chain code, and (b) 8-directional chain code.



In order to represent a boundary, it is useful to compact the raw data (list of boundary pixels). Chain Codes: list of segments with defined length and direction

- 4-directional chain codes
- 8-directional chain codes

Chain codes are used to represent a boundary by a connected sequence of straight-line segments of specified length and direction. The direction of each segment is coded by using a numbering scheme such as the ones shown in fig 11.1.

The method generally is unacceptable for two principal reasons:

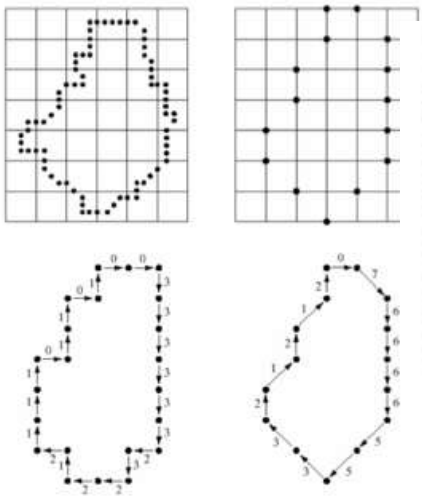
1. The resulting chain of codes tends to be quite long and,
2. Any small disturbances along the boundary due to noise or imperfect segmentation cause changes in the code that may not be related to the shape of the boundary.

An approach frequently used to circumvent the problem just discussed is to resample the boundary by selecting a larger grid spacing, as illustrated below:

The chain code of a boundary depends on the starting point. We can normalize also for rotation by using the *first difference* of the chain code instead of the code itself. The first-difference of the 4-direction chain code 10103322 is 3133030. If we elect to treat the code as a circular the result is 33133030.

To remove the dependence from the starting point: the code is circular sequence, the new starting point is the one which gives a sequence of numbers giving the smallest integer. To normalize w.r.t rotation: first difference can be used.

E.g. 10103322->3133030 (counting ccw) and adding the last transition (circular sequence: 2->1)-> 3133030.

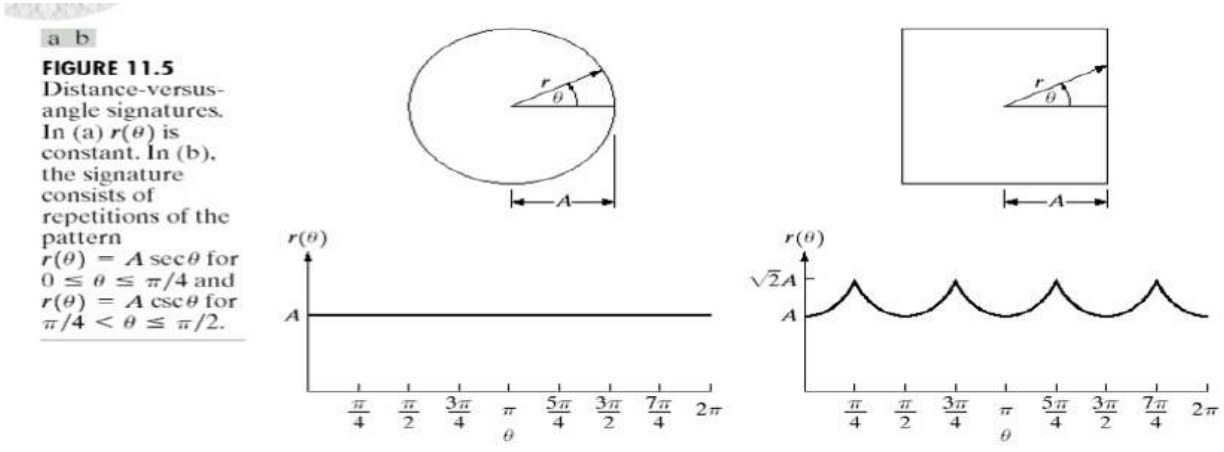


a b
c d

FIGURE 11.2
(a) Digital boundary with resampling grid superimposed. (b) Result of resampling. (c) 4-directional chain code. (d) 8-directional chain code.

7.2. Signatures

A signature is a 1-D functional representation of a boundary and may be generated in various ways. One of the simplest is to plot the distance from the centroid to the boundary as a function of angle.



a b

FIGURE 11.5
Distance-versus-angle signatures. In (a) $r(\theta)$ is constant. In (b), the signature consists of repetitions of the pattern $r(\theta) = A\sec\theta$ for $0 \le \theta \le \pi/4$ and $r(\theta) = A\csc\theta$ for $\pi/4 < \theta \le \pi/2$.

Signatures thus generated are invariant to translation, but they do depend on rotation and scaling. Normalization with respect to rotation can be achieved by finding a way to select

the same starting centroid.

Another way is to select the point in the eigen axis. One way to normalize for this result is to scale all functions so that they always span the same range of values, say, [0,1]. Whatever the method used, the basic idea is to remove dependency on size while preserving the fundamental shape of the waveforms.

7.3. Descriptors

The *length* of a boundary is one of its simplest descriptors.

The *diameter* of a boundary B is defined as Diam(B)=max[D(P_i, P_j] this line is so-called the major axis of the boundary

The minor axis of a boundary is defined as the line perpendicular to the major axis.

7.3.1. Shape Numbers

The *shape number* of a boundary based on the 4-directional code (fig. 11.1 above) is defined as the first difference of the smallest magnitude. The order n of a shape number is defined as the number of digits in its representation. Moreover, n is even for a closed boundary.

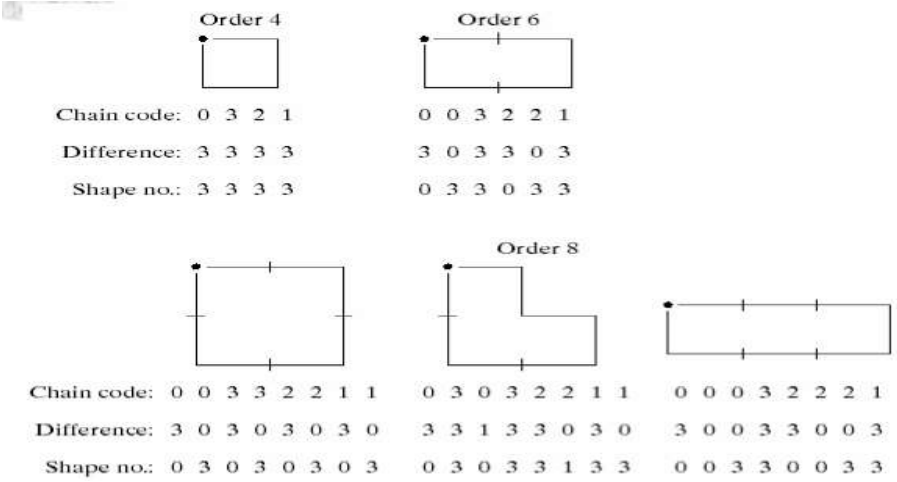


FIGURE 11.11 All shapes of order 4, 6, and 8. The directions are from Fig. 11.1(a), and the dot indicates the starting point.

7.3.2. Fourier Descriptors

- Given the list of K boundary points, start at an arbitrary point and traverse the boundary in a particular direction.
- The coordinates can be expressed in the form $x(k) = x_k$ and $y(k) = y_k$ where k = 0, 1, ..., K-1 shows the order of traversal.
- Each coordinate pair can be treated as a complex number $s(k) = x(k) + j\,y(k)$ for k = 0, 1, ..., K-1.
- The complex coefficients of the discrete Fourier transform of s(k) are called the Fourier descriptors of the boundary.

The sequence of boundary points can be treated as a sequence of complex points in the complex plane
- It becomes a 1-D descriptor
- Can be DFT-transformed

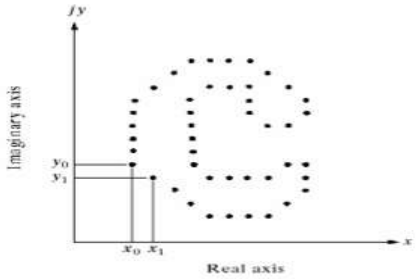
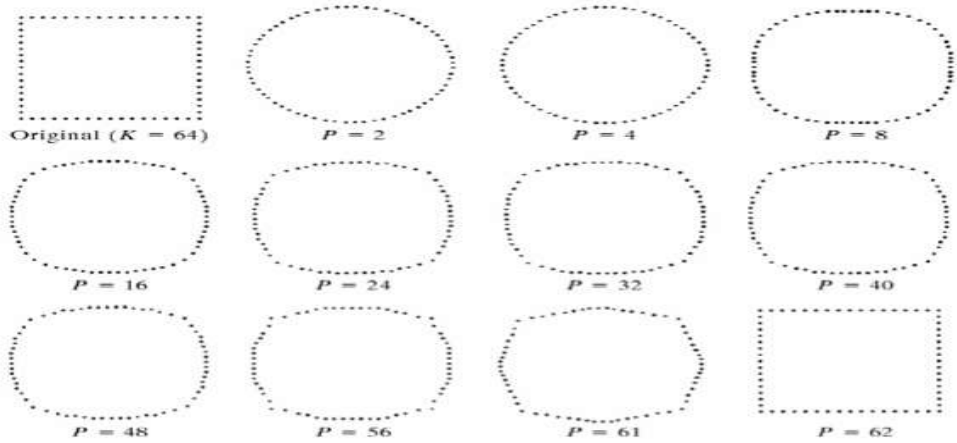


FIGURE 11.13 A digital boundary and its representation as a complex sequence. The points (x_0, y_0) and (x_1, y_1) shown are (arbitrarily) the first two points in the sequence.

- The inverse Fourier transform reconstructs s(k), i.e., the boundary.
- If only the first P coefficients are used in the reconstruction, we obtain an approximation to the boundary.
- Note that only P terms are used to obtain each component of s(k), but k still ranges from 0 to K-1.
- Since the high-frequency components of the Fourier transform account for fine detail, the smaller P becomes, the more detail is lost on the boundary.

FIGURE 11.14 Examples of reconstruction from Fourier descriptors. *P* is the number of Fourier coefficients used in the reconstruction of the boundary.



For N points with (x(n), y(n)) as one of its point

$$s(k) = x(k) + jy(k); \quad k = 0,1,2,\ldots,K-1$$

Based on the discrete Fourier Transform

$$a(u) = \frac{1}{K}\sum_{k=0}^{K-1} s(k)e^{-j2\pi uk/K} = |a(u)|e^{j\theta}$$

$$\text{for} \quad 0 \le u \le K-1$$

$$s(k) = \sum_{u=0}^{K-1} a(u)e^{j2\pi uk/K} \qquad \text{for } k = 0,1,2,\ldots,K-1$$

If onlv first P Fourier Coefficients are used.

$$\hat{s}(k) = \sum_{u=0}^{P-1} a(u) e^{j2\pi uk/K}$$

Fourier descriptors are not insensitive to translation, but effects on the transform coefficients are known.

Transformation	Boundary	Fourier Descriptor
Identity	$s(k)$	$a(u)$
Rotation	$s_r(k) = s(k)e^{j\theta}$	$a_r(u) = a(u)e^{j\theta}$
Translation	$s_t(k) = s(k) + \Delta_{xy}$	$a_t(u) = a(u) + \Delta_{xy}\delta(u)$
Scaling	$s_s(k) = \alpha s(k)$	$a_s(u) = \alpha a(u)$
Starting point	$s_p(k) = s(k - k_0)$	$a_p(u) = a(u)e^{-j2\pi k_0 u/K}$

TABLE 11.1 Some basic properties of Fourier descriptors.

8. Object Recognition

8.1.1. Pattern and Pattern Classes

A pattern is an arrangement of descriptors. The name feature is used often in the pattern recognition literature to denote a descriptor. A pattern class is a family of patterns that share some common properties. Pattern classes are denoted $w_1, w_2, \ldots, w_W$,where W is the number of classes. Pattern recognition by machine involves techniques for assigning pattern to their respective classes- automatically and with as little human intervention as possible.

- We have three classes
  - *Iris virginica, Iris versicolor, Iris setosa* ω_1, ω_2 and ω_3
- Each flower is described using two features
  - Petal length, petal width

$$X = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

- There are some differences of petal length and width between all classes
- There are also some variability within each class

The setosa type is well differentiated from the two others. It is difficult to differentiate the two other types without error. It is mainly a problem with the selection of good features.

8.2. Recognition Based on Decision (Theoretic Methods)

Let $x = (x_1, x_2, \ldots, x_n)^T$ for W pattern classes $\omega_1, \omega_2, \ldots, \omega_W$

$d_i(x) > d_j(x) \quad j = 1,2,\ldots,W; j \neq i$

- In other words, an unknown pattern **x** is said to belong to the *i*th pattern class if, upon substitution of **x** into all decision functions, $d_i(x)$ yields the largest numerical value.

8.2.1. Minimum Distance Classifier

- Suppose that we define the prototype of each pattern class to be the mean vector of the patterns of that class: $m_j = \frac{1}{N_j}\sum_{x \in \omega_j} x_j \quad j = 1,2,\ldots,W$
- We then assign **x** to class ω_i if $D_i(\mathbf{x})$ is the smallest distance. $D_j(x) = \|x - m_j\|$

selecting the smallest distance is equivalent to evaluating the functions $d_j(x) = x^T m_j - \frac{1}{2} m_j^T m_j \quad j = 1,2,\ldots,W$ assign **x** to class ω_i if $d_i(\mathbf{x})$ is the largest numerical value.

8.2.2. Matching by Correlation

Correlation between a sub-image w(x,y) and an image f(x,y)

- w(x,y) is of size j×K
- f(x,y) is of size M×N
- $J \leq M \text{ and } K \leq N$

The correlation between f(x,y) and w(x,y) is:

$$c(x,y) = \sum_s \sum_t f(s,t)w(x+s, y+t)$$

$$c(x,y) = \sum_s \sum_t f(s,t)w(x+s, y+t)$$

- *correlation coefficient*, which is defined as

$$\gamma(x,y) = \frac{\sum_s \sum_t [f(s,t) - \bar{f}(s,t)][w(x+s, y+t) - \bar{w}]}{\left\{\sum_s \sum_t [f(s,t) - \bar{f}(s,t)]^2 \sum_s \sum_t [w(x+s, y+t) - \bar{w}]^2\right\}^{\frac{1}{2}}}$$

8.3. Human Perception

- Humans have developed highly sophisticated skills for sensing their environment and taking actions according to what they observe, e.g.,
  - recognizing a face,
  - understanding spoken words,
  - reading handwriting,
  - distinguishing fresh food from its smell.
- We would like to give similar capabilities to machines.

8.3.1. Pattern Recognition

- A *pattern* is an entity, vaguely defined, that could be given a name, e.g.,
  - fingerprint image,
  - handwritten word,
  - human face,
  - speech signal,
  - DNA sequence,
  - ...
- *Pattern recognition* is the study of how machines can
  - observe the environment,
  - learn to distinguish patterns of interest,
  - make sound and reasonable decisions about the categories of the patterns.

Table 1: Example pattern recognition applications.

Problem Domain	Application	Input Pattern	Pattern Classes
Document image analysis	Optical character recognition	Document image	Characters, words
Document classification	Internet search	Text document	Semantic categories
Document classification	Junk mail filtering	Email	Junk/non-junk
Multimedia database retrieval	Internet search	Video clip	Video genres
Speech recognition	Telephone directory assistance	Speech waveform	Spoken words
Natural language processing	Information extraction	Sentences	Parts of speech
Biometric recognition	Personal identification	Face, iris, fingerprint	Authorized users for access control
Medical	Computer aided diagnosis	Microscopic image	Cancerous/healthy cell
Military	Automatic target recognition	Optical or infrared image	Target type
Industrial automation	Printed circuit board inspection	Intensity or range image	Defective/non-defective product
Industrial automation	Fruit sorting	Images taken on a conveyor belt	Grade of quality
Remote sensing	Forecasting crop yield	Multispectral image	Land use categories
Bioinformatics	Sequence analysis	DNA sequence	Known types of genes
Data mining	Searching for meaningful patterns	Points in multidimensional space	Compact and well-separated clusters

An Example: Decision Process

- What kind of information can distinguish one species from the other?
  - length, width, weight, number and shape of fins, tail shape, etc.
- What can cause problems during sensing?
  - lighting conditions, position of fish on the conveyor belt, camera noise, etc.
- What are the steps in the process?
  - capture image → isolate fish → take measurements → make decision

An Example: Selecting Features

- Assume a fisherman told us that a sea bass is generally longer than a salmon.
- We can use length as a *feature* and decide between sea bass and salmon according to a threshold on length.
- How can we choose this threshold?

However,

- Even though sea bass is longer than salmon on the average, there are many examples of fish where this observation does not hold.
- Try another feature: average lightness of the fish scales.

Cost of Error

- We should also consider *costs of different errors* we make in our decisions.
- For example, if the fish packing company knows that:
  - Customers who buy salmon will object vigorously if they see sea bass in their cans.
  - Customers who buy sea bass will not be unhappy if they occasionally see some expensive salmon in their cans.
- How does this knowledge affect our decision?
- Assume we also observed that sea bass are typically wider than salmon.
- We can use two features in our decision:
  - lightness: x_1
  - width: x_2
- Each fish image is now represented as a point (*feature vector*)

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

in a two-dimensional *feature space*.

- Does adding more features always improve the results?
  - Avoid unreliable features.
  - Be careful about correlations with existing features.
  - Be careful about measurement costs.
  - Be careful about noise in the measurements.
- Is there some *curse* for working in very high dimensions?

8.3.2. Pattern Recognition Systems

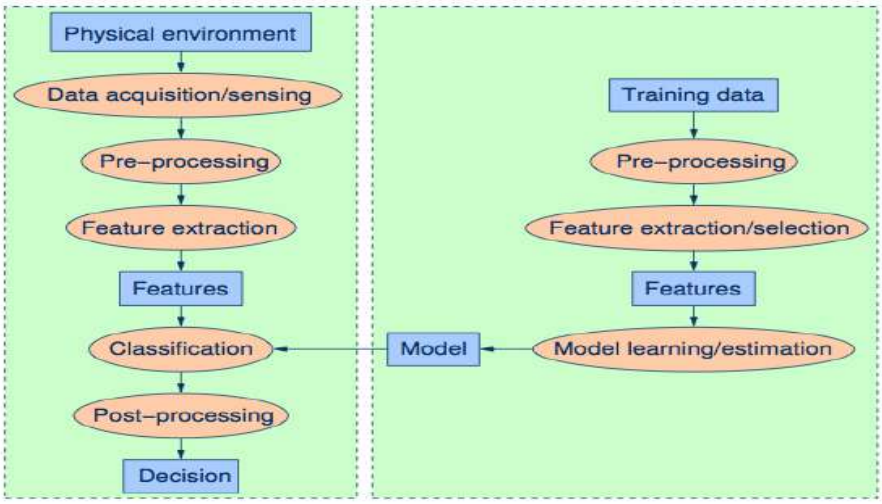


Figure 7: Object/process diagram of a pattern recognition system.

- Data acquisition and sensing:
  - Measurements of physical variables.
  - Important issues: bandwidth, resolution, sensitivity, distortion, SNR, latency, etc.
- Pre-processing:
  - Removal of noise in data.
  - Isolation of patterns of interest from the background.
- Feature extraction:
  - Finding a new representation in terms of features.

- Model learning and estimation:
  - Learning a mapping between features and pattern groups and categories.
- Classification:
  - Using features and learned models to assign a pattern to a category.
- Post-processing:
  - Evaluation of confidence in decisions.
  - Exploitation of context to improve performance.
  - Combination of experts.

Figure 8: The design cycle.

- Data collection:
  - Collecting training and testing data.
  - How can we know when we have adequately large and representative set of samples?
- Feature selection:
  - Domain dependence and prior information.
  - Computational cost and feasibility.
  - Discriminative features.
    - Similar values for similar patterns.
    - Different values for different patterns.
  - Invariant features with respect to translation, rotation and scale.
  - Robust features with respect to occlusion, distortion, deformation, and variations in environment.
- Model selection:
  - Domain dependence and prior information.
  - Definition of design criteria.
  - Parametric vs. non-parametric models.
  - Handling of missing features.
  - Computational complexity.
  - Types of models: templates, decision-theoretic or statistical, syntactic or structural, neural, and hybrid.
  - How can we know how close we are to the true model underlying the patterns?

- Training:
  - How can we learn the rule from data?
  - Supervised learning: a teacher provides a category label or cost for each pattern in the training set.
  - Unsupervised learning: the system forms clusters or natural groupings of the input patterns.
  - Reinforcement learning: no desired category is given but the teacher provides feedback to the system such as the decision is right or wrong.
- Evaluation:
  - How can we estimate the performance with training samples?
  - How can we predict the performance with future data?
  - Problems of overfitting and generalization.
- Pattern recognition techniques find applications in many areas: machine learning, statistics, mathematics, computer science, biology, etc.
- There are many sub-problems in the design process.
- Many of these problems can indeed be solved.
- More complex learning, searching and optimization algorithms are developed with advances in computer technology.
- There remain many fascinating unsolved problems.

9. Image Processing and Restoration: Noise Removal

Image restoration attempts to restore images that have been degraded

- Identify the degradation process and attempt to reverse it
- Similar to image enhancement, but more objective

The sources of noise in digital images arise during image acquisition (digitization) and transmission

- Imaging sensors can be affected by ambient conditions
- Interference can be added to an image during transmission

9.1. Noise Model

We can consider a noisy image to be modelled as follows: $g(x,y) = f(x,y) + \eta(x,y)$

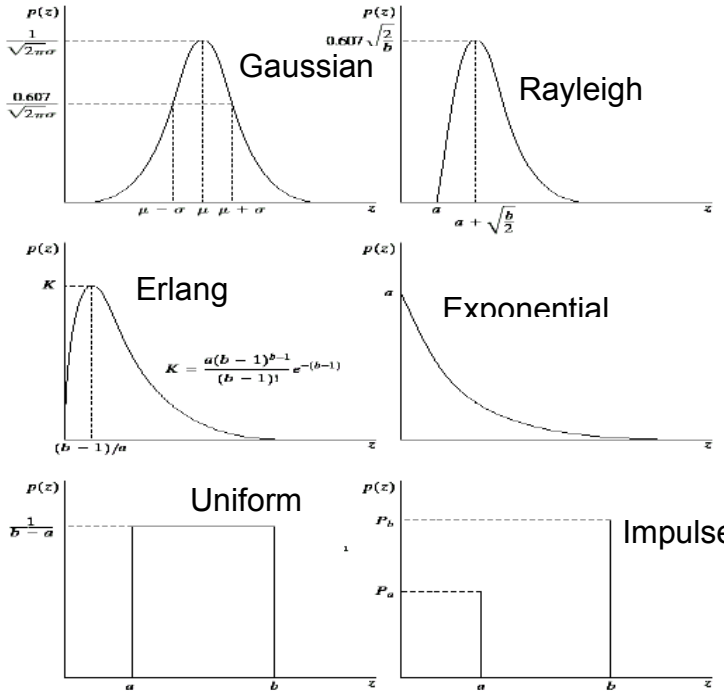
where *f(x, y)* is the original image pixel, *η(x, y)* is the noise term and *g(x, y)* is the resulting noisy pixel. If we can estimate the model the noise in an image is based on this will help us to figure out how to restore the image.

We can consider a noisy image to be modelled as follows: $g(x,y) = f(x,y) + \eta(x,y)$

where *f(x, y)* is the original image pixel, *η(x, y)* is the noise term and *g(x, y)* is the resulting noisy pixel. If we can estimate the model the noise in an image is based on this will help us to figure out how to restore the image.

There are many different models for the image
noise term *η(x, y)*:

- Gaussian
  - Most common model
- Rayleigh
- Erlang
- Exponential
- Uniform
- Impulse
  - *Salt and pepper* noise



9.1.1. Filtering to Remove Noise

We can use spatial filters of different kinds to remove different kinds of noise. The *arithmetic mean* filter is a very simple one and is calculated as follows:

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

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

This is implemented as the simple smoothing filter. It blurs the image to remove noise. There are different kinds of mean filters all of which exhibit slightly different behaviour:

- Geometric Mean
- Harmonic Mean
- Contraharmonic Mean

Geometric Mean

Achieves similar smoothing to the arithmetic mean, but tends to lose less image detail.

$$\hat{f}(x,y) = \left[\prod_{(s,t)\in S_{xy}} g(s,t) \right]^{\frac{1}{mn}}$$

Harmonic Mean:

Works well for salt noise, but fails for pepper noise. Also does well for other kinds of noise such as Gaussian noise

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

Contraharmonic Mean:

Q is the *order* of the filter and adjusting its value changes the filter's behaviour. Positive values of *Q* eliminate pepper noise. Negative values of *Q* eliminate salt noise. Choosing the wrong value for Q when using the contraharmonic filter can have drastic results.

$$\hat{f}(x,y) = \frac{\sum_{(s,t)\in S_{xy}} g(s,t)^{Q+}}{\sum_{(s,t)\in S_{xy}} g(s,t)^{Q}}$$

9.2. Order Statistics Filters

Spatial filters that are based on ordering the pixel values that make up the neighbourhood operated on by the filter

Useful spatial filters include

- Median filter
- Max and min filter
- Midpoint filter
- Alpha trimmed mean filter

9.2.1. Median Filter

Excellent at noise removal, without the smoothing effects that can occur with other smoothing filters.

Particularly good when salt and pepper noise is present

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

9.2.2. Max and Min Filter

Max Filter: $\hat{f}(x,y) = \max_{(s,t)\in S_{xy}} \{g(s,t)\}$

Min Filter: $\hat{f}(x,y) = \min_{(s,t)\in S_{xy}} \{g(s,t)\}$

Max filter is good for pepper noise and min is good for salt noise.

9.2.3. MidPoint Filter

Good for Random Gaussian and uniform noise. $\hat{f}(x,y) = \frac{1}{2}\left[\max_{(s,t)\in S_{xy}} \{g(s,t)\} + \min_{(s,t)\in S_{xy}} \{g(s,t)\}\right]$

9.2.4. Alpha-Trimmed Mean Filter

We can delete the *d/2* lowest and *d/2* highest grey levels. So $g_r(s, t)$ represents the remaining *mn – d* pixels $\hat{f}(x,y) = \frac{1}{mn-d} \sum_{(s,t)\in S_{xy}} g_r(s,t)$

9.3. Periodic Noise

Typically arises due to electrical or electromagnetic interference. Gives rise to regular noise patterns in an image Frequency domain techniques in the Fourier domain are most effective at removing periodic noise.

9.3.1. Band Reject Filters

Removing periodic noise form an image involves removing a particular range of frequencies from that image. *Band reject* filters can be used for this purpose. An ideal band reject filter is given as follows:

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

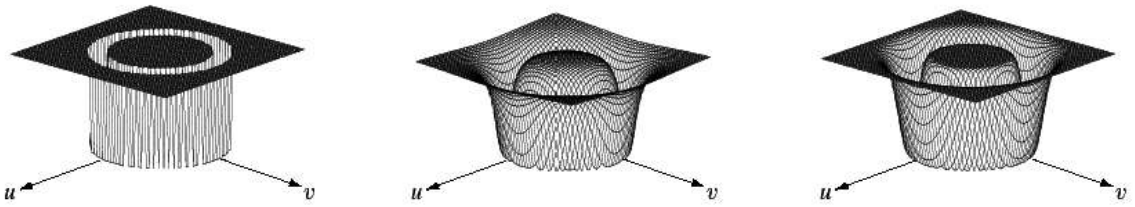
Similarly, a Butterworth bandreject filter of order n is given by the expressio

$$H(u,v) = \frac{1}{1 + \left[\frac{D(u,v)W}{D^2(u,v) - D_0^2}\right]^{2n}} \quad (5.4\text{-}2$$

and a Gaussian bandreject filter is given by

$$H(u,v) = 1 - e^{-\frac{1}{2}\left[\frac{D^2(u,v) - D_0^2}{D(u,v)W}\right]^2}. \quad (5.4\text{-}3$$

The ideal band reject filter is shown below, along with Butterworth and Gaussian versions of the filter



- Restoration is slightly more objective than enhancement
- Spatial domain techniques are particularly useful for removing random noise
- Frequency domain techniques are particularly useful for removing periodic noise

9.3.2. Adaptive Filters

The filters discussed so far are applied to an entire image without any regard for how image characteristics vary from one point to another. The behaviour of **adaptive filters** changes depending on the characteristics of the image inside the filter region. We will take a look at the **adaptive median filter.**

Adaptive Median Filtering

The median filter performs relatively well on impulse noise as long as the spatial density of the impulse noise is not large. The adaptive median filter can handle much more spatially dense impulse noise, and also performs some smoothing for non-impulse noise. The key insight in the adaptive median filter is that the filter size changes depending on the characteristics of the image

Remember that filtering looks at each original pixel image in turn and generates a new filtered pixel

First examine the following notation:

- z_{min} = minimum grey level in S_{xy}; z_{max} = maximum grey level in S_{xy}
- z_{med} = median of grey levels in S_{xy}; z_{xy} = grey level at coordinates (x, y)
- S_{max} =maximum allowed size of S_{xy}

Level A: $A1 = z_{med} - z_{min}$ – $A2 = z_{med} - z_{max}$ – If $A1 > 0$ and $A2 < 0$, Go to level B – Else increase the window size – If window size ≤ repeat S_{max} level A – Else output z_{med}	Level B: $B1 = z_{xy} - z_{min}$ – $B2 = z_{xy} - z_{max}$ – If $B1 > 0$ and $B2 < 0$, output z_{xy} – Else output z_{med}

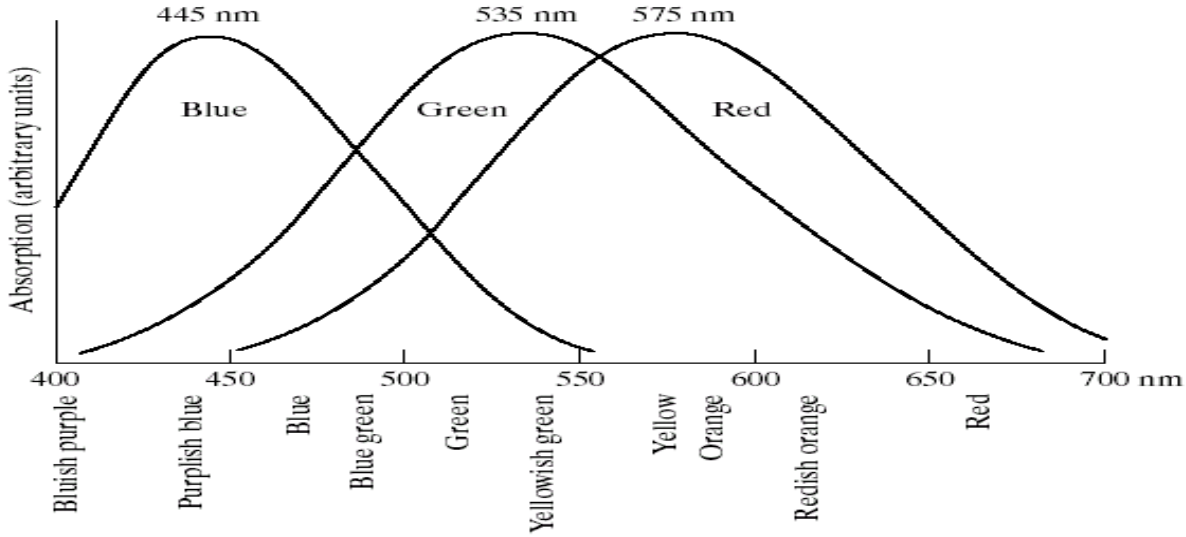
The key to understanding the algorithm is to remember that the adaptive median filter has three purposes:

- Remove impulse noise; Provide smoothing of other noise; Reduce distortion

10. Color Image Processing

The colors that humans and most animals perceive in an object are determined by the nature of the light reflected from the object. For example, green objects reflect light with wavelengths primarily in the range of 500 – 570 nm while absorbing most of the energy at other wavelengths. Chromatic light spans the electromagnetic spectrum from approximately 400 to 700 nm. As we mentioned before human color vision is achieved through 6 to 7 million cones in each eye.

Approximately 66% of these cones are sensitive to red light, 33% to green light and 6% to blue light. Absorption curves for the different cones have been determined experimentally. Strangely these do not match the CIE standards for red (700nm), green (546.1nm) and blue (435.8nm) light as the standards were developed before the experiments!

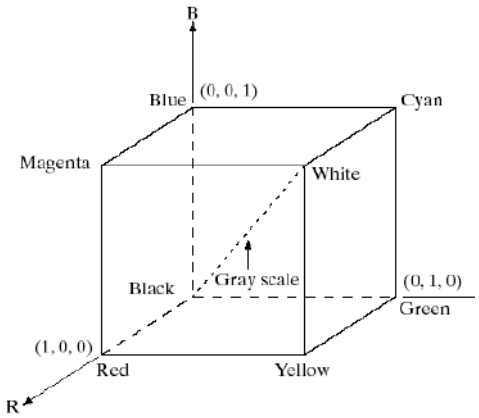


3 basic qualities are used to describe the quality of a chromatic light source:

- **Radiance:** the total amount of energy that flows from the light source (measured in watts)
- **Luminance:** the amount of energy an observer *perceives* from the light source (measured in lumens). Note we can have high radiance, but low luminance
- **Brightness:** a subjective (practically unmeasurable) notion that embodies the intensity of light
- Specifying colours systematically can be achieved using the CIE **chromacity diagram**
- On this diagram the x-axis represents the proportion of red and the y-axis represents the proportion of green used
- The proportion of blue used in a colour is calculated as: $z = 1 - (x + y)$
- Any colour located on the boundary of the chromacity chart is fully saturated
- The point of equal energy has equal amounts of each colour and is the CIE standard for pure white
- Any straight line joining two points in the diagram defines all of the different colours that can be obtained by combining these two colours additively

This can be easily extended to three points

- This means the entire colour range cannot be displayed based on any three colours
- The triangle shows the typical colour gamut produced by RGB monitors
- The strange shape is the gamut achieved by high quality colour printers



10.1. Color Models

- RGB (**R**ed **G**reen **B**lue)
- HSI (**H**ue **S**aturation **I**ntensity)

10.1.1. RGB

In the RGB model each colour appears in its primary spectral components of red, green and blue

The model is based on a Cartesian coordinate system

- RGB values are at 3 corners
- Cyan magenta and yellow are at three other corners
- Black is at the origin
- White is the corner furthest from the origin
- Different colors are points on or inside the cube represented by RGB vectors

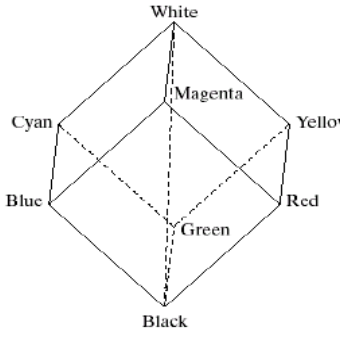
Images represented in the RGB color model consist of three component images – one for each primary color. When fed into a monitor these images are combined to create a composite color image. The number of bits used to represent each pixel is referred to as the color depth. A 24-bit image is often referred to as a full-color image as it allows (2^{8^3})= 16,777,216 colors.

10.1.2. HSI

RGB is useful for hardware implementations and is serendipitously related to the way in which the human visual system works. However, RGB is not a particularly intuitive way in which to describe colors. Rather when people describe colors they tend to use **hue**, **saturation** and **brightness.** RGB is great for color generation, but HSI is great for color description.

The HSI model uses three measures to describe colours:

- **Hue:** A colour attribute that describes a pure colour (pure yellow, orange or red)
- **Saturation:** Gives a measure of how much a pure colour is diluted with white light
- **Intensity:** Brightness is nearly impossible to measure because it is so subjective. Instead we use intensity. Intensity is the same achromatic notion that we have seen in grey level images

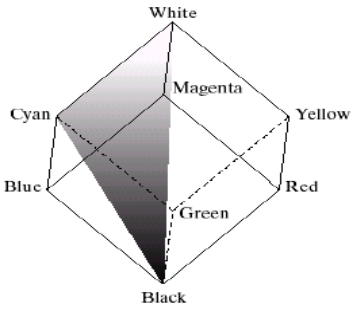


10.1.3. HSI, Intensity and RGB

Intensity can be extracted from RGB images – which is not surprising if we stop to think about it

Remember the diagonal on the RGB colour cube that we saw previously ran from black to white

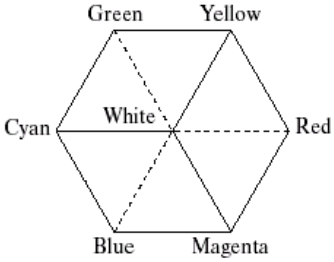
Now consider if we stand this cube on the black vertex and position the white vertex directly above it



Now the intensity component of any color can be determined by passing a plane *perpendicular* to the intensity axis and containing the color point. The intersection of the plane with the intensity axis gives us the intensity component of the color.

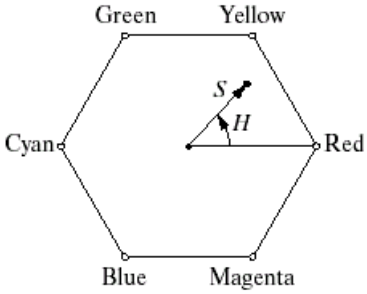
10.1.4. HSI, Hue and RGB

In a similar way we can extract the hue from the RGB color cube. Consider a plane defined by the three points cyan, black and white. All points contained in this plane must have the same hue (cyan) as black and white cannot contribute hue information to a color.



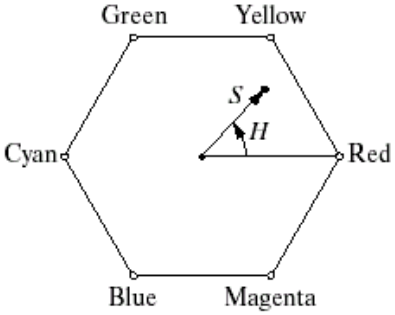
10.1.5. The HSI Color Model

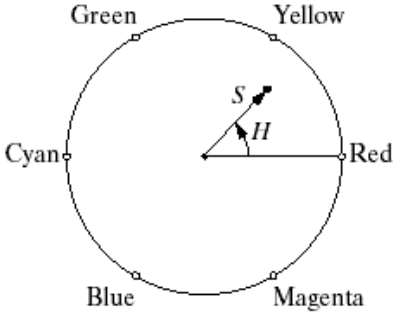
Consider if we look straight down at the RGB cube as it was arranged previously. We would see a hexagonal shape with each primary color separated by 120°
and secondary colors at 60° from the primaries

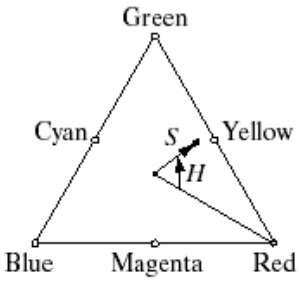


So the HSI model is composed of a vertical intensity axis and the locus of color points that lie on planes perpendicular to that axis. To the right we see a hexagonal shape and an arbitrary color point. The hue is determined by an angle from a reference point, usually red. The saturation is the distance from the origin to the point. The intensity is determined by how far up the vertical intensity axis this hexagonal plane sits (not apparent from this diagram.

Because the only important things are the angle and the length of the saturation vector this plane is also often represented as a circle or a triangle







10.1.6. Converting from RGC to HSI

Given a colour as R, G, and B its H, S, and I values are calculated as follows:

$$H = \begin{cases} \theta & \text{if } B \le G \\ 360 - \theta & \text{if } B > G \end{cases} \qquad \theta = \cos^{-1}\left\{ \frac{\frac{1}{2}\left[(R-G)+(R-B)\right]}{\left[(R-G)^2+(R-B)(G-B)\right]^{\frac{1}{2}}} \right\}$$

$$S = 1 - \frac{3}{(R+G+B)}\left[\min(R,G,B)\right] \qquad I = \tfrac{1}{3}(R+G+B)$$

10.1.7. Converting from HSI to RGB

Given a colour as H, S, and I it's R, G, and B values are calculated as follows:

– RG sector ($0 <= H < 120°$)

$$R = I\left[1 + \frac{S\cos H}{\cos(60 - H)}\right] \qquad G = 3I - (R + B) \qquad B = I(1 - S)$$

– GB sector ($120° <= H < 240°$)

$$R = I(1 - S) \qquad G = I\left[1 + \frac{S\cos(H - 120)}{\cos(H - 60)}\right] \qquad B = 3I - (R + G)$$

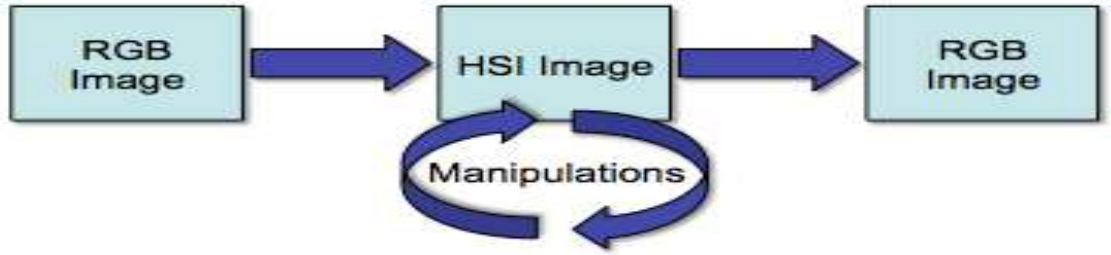
– BR sector ($240° <= H <= 360°$)

$$R = 3I - (G + B) \qquad G = I(1 - S) \qquad B = I\left[1 + \frac{S\cos(H - 240)}{\cos(H - 180)}\right]$$

10.1.8. Manipulating Images in the HSI Model

In order to manipulate an image under the HIS model we:

- First convert it from RGB to HSI
- Perform our manipulations under HSI
- Finally convert the image back from HSI to RGB

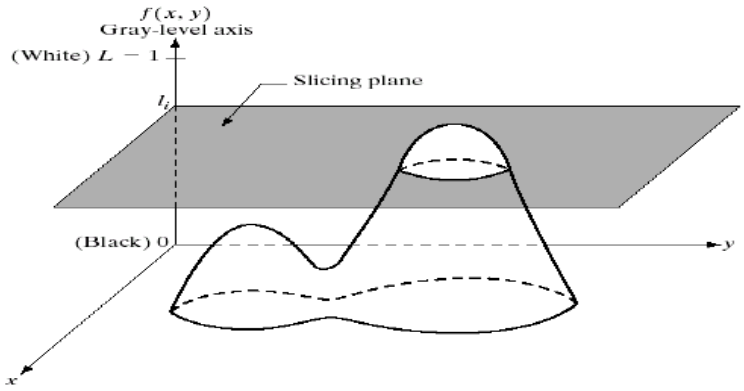


10.2. Pseudocolor Image Processing

Pseudocolour (also called false colour) image processing consists of assigning colours to grey values based on a specific criterion. The principle use of pseudocolour image processing is for human visualization. Humans can discern between thousands of colour shades and intensities, compared to only about two-dozen or so shades of grey.

Intensity Slicing

Intensity slicing and colour coding is one of the simplest kinds of pseudocolour image processing. First we consider an image as a 3D function mapping spatial coordinates to intensities (that we can consider heights). Now consider placing planes at certain levels parallel to the coordinate plane. If a value is one side of such a plane it is rendered in one colour, and a different colour if on the other side



In general intensity slicing can be summarised as:

- Let [*0, L-1*] represent the grey scale
- Let l_0 represent black [$f(x, y) = 0$] and let l_{L-1} represent white [$f(x, y) = L-1$]
- Suppose *P* planes perpendicular to the intensity axis are defined at levels $l_1, l_2, \ldots, l_p$
- Assuming that $0 < P < L-1$ then the *P* planes partition the grey scale into $P + 1$ intervals $V_1, V_2, \ldots, V_{P+1}$
- Grey level colour assignments can then be made according to the relation:
- where c_k is the colour associated with the k^{th} intensity level V_k defined by the partitioning planes at $l = k - 1$ and $l = k$