Digital Image Fundamentals

2.1 Light and the Electromagnetic Spectrum

The nature of light reflected by the object determines the colors that we perceive. A body may reflect balanced or unbalanced type of light. The object that returns balanced type of light appears brighter than the object that reflects unbalanced light.

In $monochromatic\ or\ achromatic$ light colors are absent whereas $chromatic$ light contains various colors. There are some attributes that are associated with achromatic light as listed below.

- $Intensity$ of achromatic light determines its amount known as level.
- There are 256 $levels$ in a monochromatic image and these levels are called the gray scale
- An image formed using gray scale is known as $grayscale\ image$

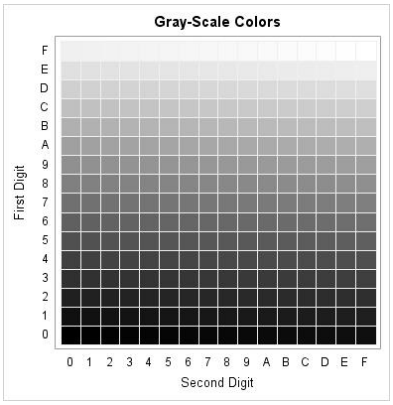


Figure 2-1: Grayscale matrix

Similarly, the chromatic light has some associated characteristics that include.

- $Frequency$: Chromatic (color) light spans the electromagnetic energy spectrum from approximately 0.43 to 0.79 mm
- $Radiance$ is the total amount of energy that flows from the light source, and it is usually measured in watts (W)
- $Luminance$ gives a measure of the amount of energy an observer perceives from a light source and is measured in lumens (lm).
- $Brightness$ is a subjective descriptor of light perception that is practically impossible to measure

A prominent method of image generation is based on measuring the light intensity incident on some light sensitive device. However, it is not only method and there are other sources that can be used to generate images that may include the sound waves, electron beams for electron microscopy and software based virtual image generation.

2.2 Image Sensing and Acquisition

Images can be acquired using single sensing elements (photodiode) or an array of sensing elements. When a light wave strikes a photo diode, it produces voltage that is proportional to intensity of light. The selectivity of sensor can be improved by using a filter in front of diode.

A linear sensor can capture the light coming from a linear source. We can think of linear sensor as “line” imaging device.

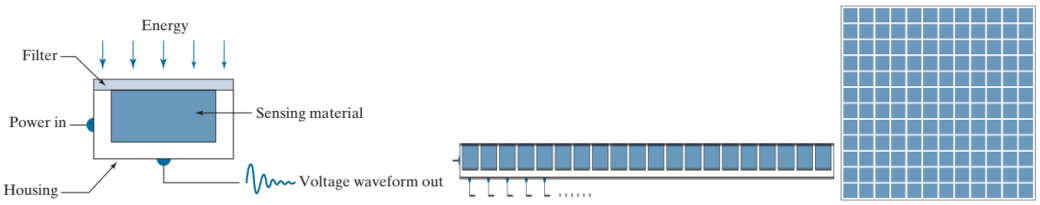


Figure 2-2: A) Single photo diode B) 1D Array of Photo Diodes C) A 2D Matrix of Light Sensing Elements (Photo Diodes)

And finally, if the diodes are arranged in the form of a matrix, a 2D image can be constructed.

2.2.1 Image formation Model

An image can be considered as 2D function $f(x,y).x,y$ denote the position of pixel which is discrete. For example, in above picture, position of each diode determines the position of pixel. The position of a diode in row one and column one can be denoted as $f(0,0)$ [or $f(1,1)$ if indexing starts from 1] and the value of light intensity on that diode is the value of corresponding pixel located at position $(0,0)$ in image. Image pixel values are normally non-negative however it depends upon the intensity interpretation scheme that we use. If we use negative values to represent some levels and positive values for remaining levels, our image may contain negative intensity values.

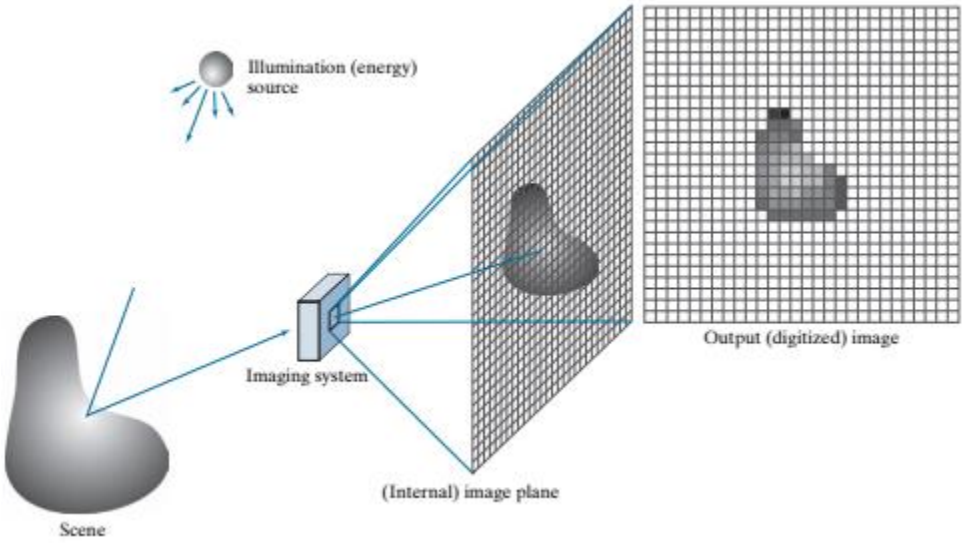


Figure 2-3: Image Formation

There are two factors that affect the value of $f(x,y)$.

- $\boldsymbol{Illumination\ i(x,y)}$: The amount of source illumination incident on the scene being viewed, and
- $\boldsymbol{Reflectance\ r(x,y)}$: The amount of illumination reflected by the objects in the scene

An image $f(x,y)$ is formed as product of these two components

$$f(x,y) = i(x,y)r(x,y) \qquad (2-1)$$

$$\text{Where} \quad 0 < i(x,y) < \infty \text{ and } 0 < r(x,y) < 1$$

Although theoretically intensity $i(x,y)$ could be zero to infinite however practically, we define its finite discrete levels i.e.

$$0 < i(x, y) < Lmax$$

Or equivalently

$$Lmin < i(x, y) < Lmax$$

Generally, the interval $[Lmin, Lmax]$ define the gray scale. Normally this interval is taken as $[0,1]$ or $[0, C]$ where C is some upper limit on level count i.e. 255.

2.2.2 Sampling and Quantization

A computer is a machine that work with discrete quantitates however most of the quantities occurring in our real life are continuous in nature. For example, intensity of light, temperature and amount of water. It is required to convert continuous quantities into discrete for computer processing.

- $Sampling$: it is the process of measuring a continuous quantity or signal after equal intervals of time. For example, we can measure intensity of light after every one second.
- $Quantization$: Continuous quantity measured after equal intervals of time may be fractional. For example, in one second you measure intensity of light that is 5 and after next second, it is measured as 5.8. In order to make computer processing better and efficient, it is required to round such fractional quantities to complete quantities.

Similarly, when light is made incident on a 2D matrix of photo diodes as shown in Figure 2-4, it is required to measure the light intensity in equal intervals of time and then quantize them into discrete levels. Figure 2-6 shows this description pictorially.

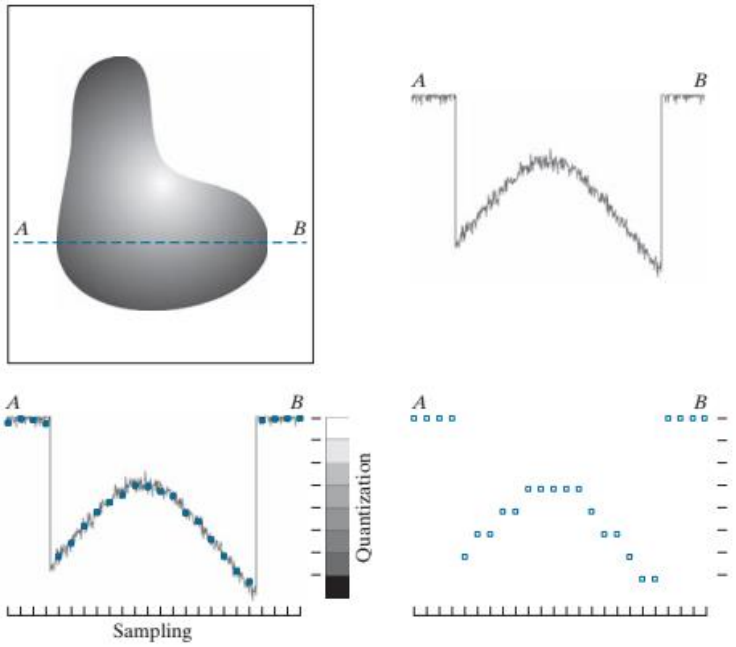


Figure 2-4: Image Sampling and Quantization

Following figure 2-7 shows a continuous image and its equivalent digitization.

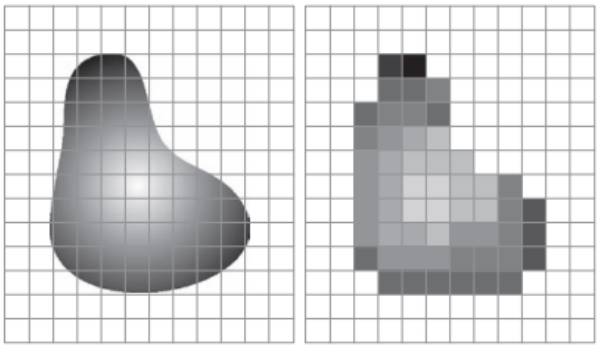


Figure 2-5: Continuous Image and its Digitization

2.2.3 Representing Digital Images

Let $f(s,t)$ is the function that represents the continuous image and $f(x,y)$ is its equivalent representation in discrete form obtained after sampling and digitization. Let there are M rows and N columns in resulting digital image. A matrix of discrete values can represent the digital image. This matrix will contain the sampled and quantized values of intensity of light incident on each sensor. If we index the sensors starting from zero, then $f(0,0)$ will represent the value of light intensity incident on first sensor.

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

There are different notations that can be used to represent the digital image in the form of unsigned integer values. Here is another notation of image.

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

There are few characteristics associated with this image representation in matrix form.

- Each single diode represents value for a pixel
- Each pixel value is unsigned integer normally ranging from 0-255 or in normalized form 0-1
- Indexing of image pixel starts form top-left corner
- Such image representation is also known as representation in $spatial\ domain$
- Number of intensity levels are denoted as $L = 2^k$ where k is the number of bits used to store the value of a pixel. Total number of bits required to store an image is $MxNxk$ where M is the number of rows, N is the number of columns and k is the number of bits used to store value of one pixel.
- $Dynamic\ range$: It represents the highest and lowest intensity levels that an image can have.
- $Intensity\ Contrast$: It is the difference between lowest and highest intensity level in an image
- $Contrast\ Ratio$: it is the ratio of highest and lowest intensity levels present in the image.

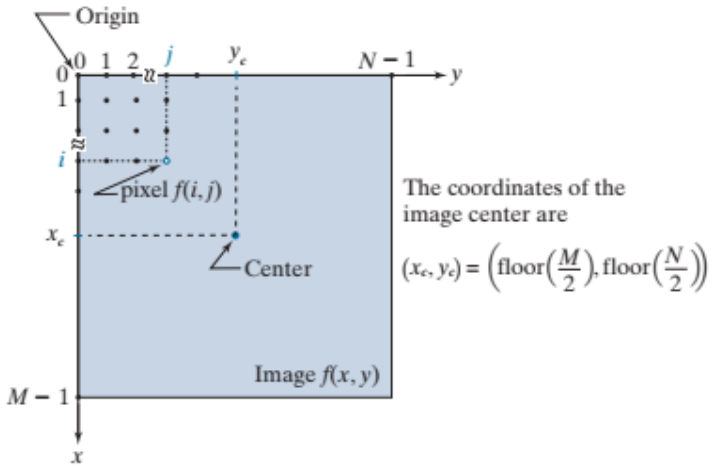


Figure 2-6: Image Representation and its Characteristics

Program P2-1 provides the code to load and display image using OpenCV
Program P2-2 provides the code display different features associated with image
Program P2-3 shows that how can you create a matrix of values and display it as an image

---

Spatial and Intensity Resolution

There are two type of resolutions (details) that we can consider in the context of an image.

- $Spatial\ Resolution$: This defines the number of pixels in unit area. For example, the number of pixels that can be drawn in a millimeter area. More the pixels, higher the resolution. It can be measured in dots per inch.
- $Intensity\ Resolution$: It defines the number of intensity levels that are present in the image. For example, an image has intensity levels in the interval [0-255], its resolution is 255 whereas another image having intensity levels from [0-1000] and such an image has intensity levels 1000. Usually the intensity levels are power of 2.

Program P2-4 provides the code to find spatial and intensity resolution of and image using OpenCV

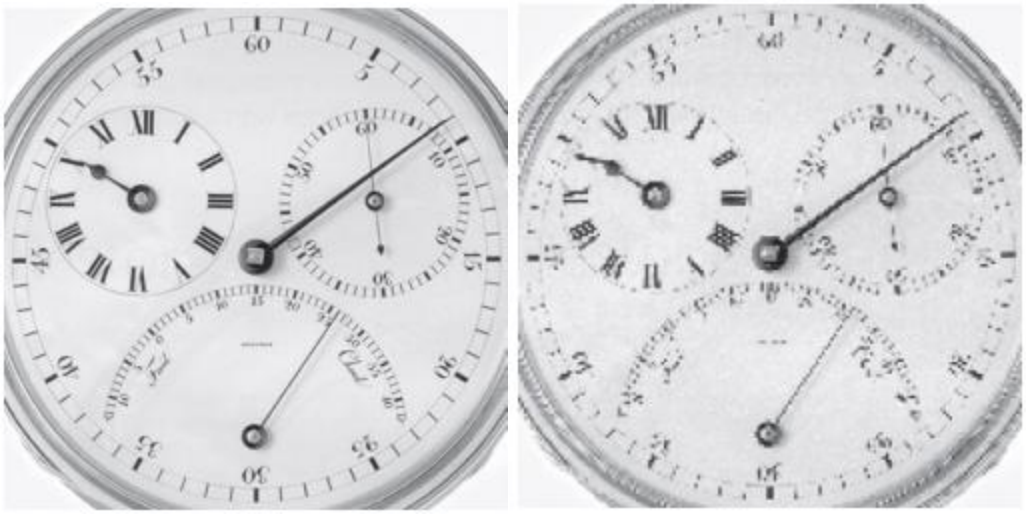


Figure 2-7: A) High Spatial Resolution Image – 930 dpi B) Low Resolution Image – 72 dpi

And following figure shows two images having different intensity resolutions

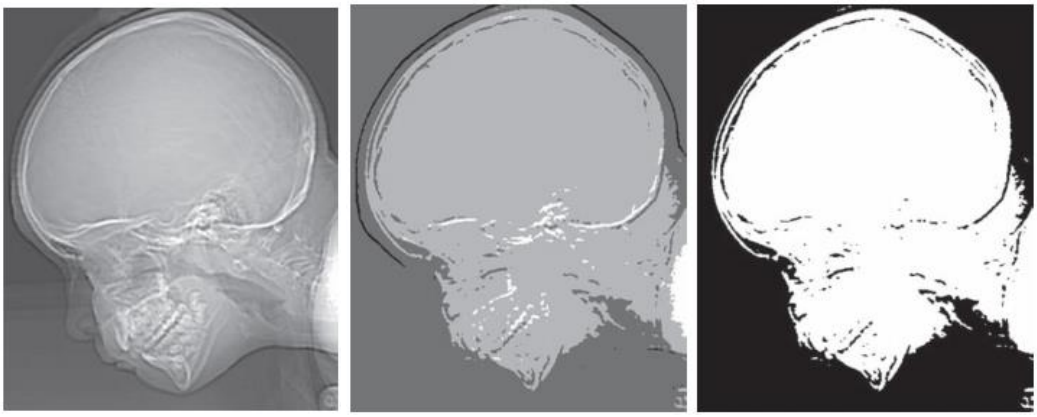


Figure 2-8: Intensity Resolution. A) 256 Level Image B) 4 Intensity Levels C) 2 Intensity Levels

There are some concepts associated with intensity resolutions:

- $Contours$: It is an outline representing or bounding the shape or form of something
- $False\ Contours$: With the decrease of intensity resolution, very fine ridge-like structures in areas of constant intensity are introduced (false lines) which are named as false contours
- $Image\ information$: Due to nature of image the level of information present in an image may vary. There is no well-defined limit however some images contain more information whereas other contain less information

Figure 2-9: A) Image with low level of details B) Medium Level of detail C) High level of detail

2.2.4 Image Interpolation

There are multiple operations where image interpolation is required. These image operations may include the zooming, shrinking and rotating. Estimating the unknown pixel values using known pixel values is called interpolation. There are three type of interpolations which are usually used:

- Nearest neighborhood interpolation
- Bilinear interpolation
- Bicubic interpolation

Now we explain the algorithms of these interpolation methods.

2.2.4.1 Nearest neighborhood interpolation

This method simply determines the "nearest" neighboring pixel, and assumes the intensity value of it. First, let's consider this in a 1-dimensional case. Observe the plot below:

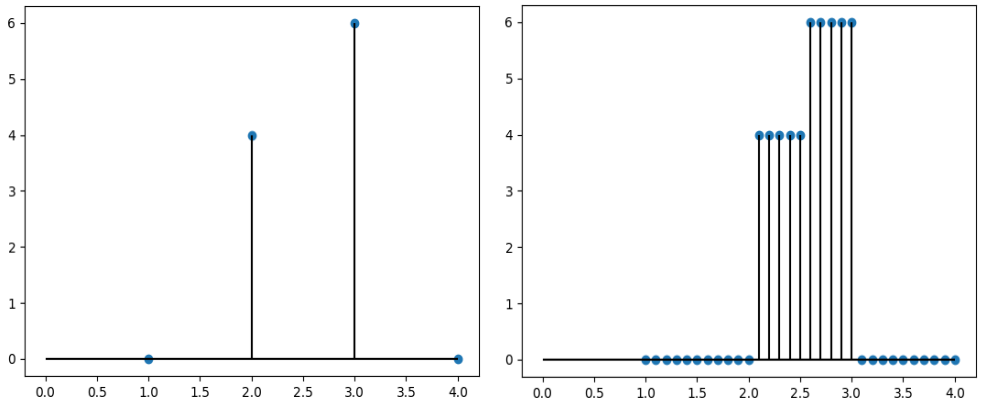


Figure 2-10: Nearest neighborhood interpolation A) Original Data B) Interpolated Data

Let's say that we want to insert more data points between the points $x1 = 2\ and\ x2 = 3$ where $f(x1) = 4$ and $f(x2) = 6$ as shown in the figure by solid black lines and their blue tips. Using nearest neighborhood our result will look like the figure 2-12 B. In Figure 2-12 B, you can see that between two points x1=2 and x2=3, we have inserted 7 points and assigned each point same intensity value as that of its nearest neighbor.

Now we extend this concept to 2D plane. Let we have a 2x2 image that we want to zoom to 9x9. This image $upsampling$ is done as:

10	20
30	40

10	10	20	20
10	10	20	20
30	30	40	40
30	30	40	40

<we will explain the interpolations later. We will introduce the code to find these type of interpolations>

2.3 Pixels and their relationships

There are different types of relationships that can exist among the pixels. This relationship is normally based on some type of criteria that we define.

2.3.1 Adjacency

Adjacency is the relationship of two pixels that defines the nearness. Depending upon the criteria we define, adjacency of two pixels is found. For example, intensity-based adjacency may define the closeness of two pixels whether they have similar intensity values or not. For Spatial adjacency, pixels are related to each other based on their location in the image.

2.3.2 4-neighbour relationship $N_4(p)$

A set of pixels that is found in four different directions from reference pixel is called 4-neighbourhood. Following table shows the four neighborhoods of a pixel.

$$N_4(p) = \{f(x, y-1), f(x-1, y), f(x+1, y), f(x, y+1) \qquad (2-2)$$

Table 1: 4-Neighbourhood of a pixel

	$f(x, y-1)$	
$f(x-1, y)$	$\boldsymbol{f(x, y)}$	$f(x+1, y)$
	$f(x, y+1)$	

2.3.3 8-neighbour relationship $N_8(p) = N_4(p) \cup N_D(p)$

A set of pixels that is found in eight different directions from reference pixel is called 4-neighbourhood. Following table shows the four neighborhoods of a pixel.

$$N_8(p) = \{f(x-1, y-1), f(x, y-1), f(x+1, y-1),$$
$$f(x-1, y), f(x+1, y),$$
$$f(x-1, y+1) f(x, y+1) f(x+1, y+1)\} \qquad (2-3)$$

Table 2: 4-Neighbourhood of a pixel

$f(x-1, y-1)$	$f(x, y-1)$	$f(x+1, y-1)$
$f(x-1, y)$	$\boldsymbol{f(x, y)}$	$f(x+1, y)$
$f(x-1, y+1)$	$f(x, y+1)$	$f(x+1, y+1)$

If a neighborhood of a pixel p contains p itself, it is called $closed$ neighborhood and if p is not contained in the neighborhood, it is called $open$ neighborhood.

2.3.4 m-neighbor relationship

It is also called m-adjacency (mixed adjacency). It is modification of 8-adjacency which is introduced to eliminate the ambiguities that often arise when 8-adjacency is used. Let V define the set of intensities that determine the adjacency (if two pixels have intensity values from that set, they are said to be adjacent). Two pixels p and q are said to be m-adjacent if they meet following conditions:

1. q is in $N_4(p)$ OR
2. q is in $N_D(p)$ and the set $N_4(p) \cap N_4(q) = \emptyset$

Two conditions need to be fulfilled. If a pixel q is in 4-neighbours of p, it is in m-adjacency. Second condition has two parts

- p, q are in diagonals of each other and
- There should be no pixel which is in 4-adjacency of both and have pixel value from set V.

Example

Let if set of intensities V= {1} then

$$\begin{bmatrix} 0 & 1 & 1_q \\ 0 & 1_p & 0 \\ 0 & 0 & 1 \end{bmatrix} \rightarrow \begin{bmatrix} 0 & \mathbf{1} & 1_q \\ 0 & 1_p & \mathbf{0} \\ 0 & 0 & 1 \end{bmatrix} \rightarrow \begin{bmatrix} 0 & 1 & 1_q \\ 0 & 1_p & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- Condition-1: q must be in 4-adjacency in order to be in m-adjacency. (**false**)
- Condition-2: q is in diagonals of p and there are two pixels which are common i.e. $N_4(p) \cap N_4(q) = \{1,0\}$ that are shown as red. Now part-2 of condition-2 says that if there are any common points in four-neighborhood of both p and q, their intensities must not be from V. In our example, there are two common pixels with intensity 0 and 1. Intensity "0" is in our set V, hence it satisfies the condition for q to be in m-adjacency. But intensity "1" is from set V and it violates the condition therefore, p and q are not in m-adjacency hence their direct path is removed. Note that p, q are in 8-adjacency but not in m-adjacency.

Question

Discuss the m-adjacency if $v = \{1,2\}$ and pixels are given as follows:

$$\begin{bmatrix} 0 & 1 & 1_q \\ 0 & 2_p & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Program P2-5 4-neighbourhood of an image
Program P2-6 8-neighbourhood of an image
Program P2-7 m-adjacency of an image

2.3.5 Digital Path

If there exists a set of mutually adjacent pixels P between two pixels p and q such that we can reach from p to q by following the adjacency, the set of pixels is called the path P.

$$P = \{(x_0, y_0), (x_1, y_1), \ldots., (x_n, y_n)\} \qquad (2-4)$$

The number of pixels in P is called the $length\ of\ path$. The path could be $open$ if $(x_0, y_0) \neq (x_n, y_n)$ otherwise the path is $closed$.

<Write a program to find the path between two pixels>

2.3.6 Connected component

For a particular image, if there exists a path between two pixels, the pixels are said to be $connected$. The set of pixels that are connected with each other is called the $connected\ component$. There may be one or more such connected components in the image. If there is only one such set, it is referred as $connected\ set$. A connected component is also referred as $Region$.

2.3.7 Adjacent and Disjoint Regions

Two regions Ri and Rj are called adjacent if $RiURj$ form a connected set. Regions that are not adjacent are said to be $disjoint$.

<Write a program to find all connected components in given image>

2.3.8 Image foreground and background

If an image has K regions $R_1, R_2, \ldots, R_k$ then $R_1 \cup R_2 \cup \ldots \cup R_k$ forms the foreground of image whereas its complement is called the background.

$$Foreground = \{R_1 \cup R_2 \cup \ldots \cup R_k\} \qquad (2-5)$$

$$Background = \{R_1 \cup R_2 \cup \ldots \cup R_k\}' \ = \ image \ - \ \{R_1 \cup R_2 \cup \ldots \cup R_k\} \qquad (2-6)$$

<Write a program to find foreground and background of given image>

2.3.9 The Contour

The $boundary$ of an image is called the contour. A $boundary$ of a region R is a collection of pixels in R that are adjacent with its background (complement of region). We have to define the type of connectivity when calculating the contour i.e. $N_4(p)\ or\ N_8(p)$.

$Inner\ border$ refers to set of pixels in foreground that are adjacent to background similarly $outer\ border$ is the set of pixels that are adjacent to foreground. An $edge$ is defined as set of pixels with derivative values that precede some specific threshold.

<write a program to find the boundary in an image. Also find inner border and outer border of an object in an image>

2.3.10 Distance Measure

For many applications, we need to measure the distance between source and destination pixels. There are number of distance measure used in this context however only three are discussed here. To measure the distance between two pixels following conditions must be satisfied:

- $D(p, q) \geq 0$
- $D(p, q) = D(q, p)$
- $D(p, s) \leq D(p, q) + D(q, s)$

2.3.10.1 Euclidean Distance

This distance between two pixels $p(x, y)$ and $q(u, v)$ is measured using following formula:

$$D_e(p, q) = \sqrt{(x-u)^2 - (y-v)^2} \quad (2-7)$$

2.3.10.2 City Block Distance

It is also called the D_4 distance and is measured using following formula:

$$D_4(p, q) = |x-u| + |y-v| \quad (2-8)$$

This distance measure forms the diamond like pattern of distances from reference point as shown below:

$$D_4(p, q_i) = \begin{Vmatrix} & & 2 & & \\ & 2 & 1 & 2 & \\ 2 & 1 & 0 & 1 & 2 \\ & 2 & 1 & 2 & \\ & & 2 & & \end{Vmatrix}$$

2.3.10.3 Chessboard Distance

This measure is also called the D_8 distance and is measured using formula:

$$D_8(p, q) = \max(|x-u|, |y-v|) \quad (2-9)$$

And it forms a rectangular like pattern for distances from reference point. As an example, for $D \leq 2$ the pattern looks like this

$$D_8(p, q_i) = \begin{Vmatrix} 2 & 2 & 2 & 2 & 2 \\ 2 & 1 & 1 & 1 & 2 \\ 2 & 1 & 0 & 1 & 2 \\ 2 & 1 & 1 & 1 & 2 \\ 2 & 2 & 2 & 2 & 2 \end{Vmatrix}$$

Program P2-9 Distance Calculation implements these formulas

2.4 Basic Mathematics for Digital Image Processing

It has been observed that computer science students usually avoid mathematics. However, computing stands for mathematics so a computer scientist must convince himself to understand and program mathematics. Digital image processing requires some initial mathematics and we will review it here in this context. The reader of this text must have observed that an image is matrix of unsigned discrete values. So we will perform some matrix operations.

2.4.1 Elementwise matrix operations

Element wise operations means that each element of matrix must be applied some operations individually. Simplest operation is adding a scalar element in each pixel value in an image as shown here:

$$A = \begin{bmatrix} 1 & 1 & 1 \\ 2 & 2 & 2 \\ 3 & 3 & 3 \end{bmatrix} \rightarrow A + 5 = \begin{bmatrix} 6 & 6 & 6 \\ 7 & 7 & 7 \\ 8 & 8 & 8 \end{bmatrix}$$

2.4.2 Matrix Operations

We can also perform different matrix operations easily using python vector computing capabilities. For example, following two matrices can easily be added together using single python statement.

Input matrices	Result	Python Statement
$\boldsymbol{A}=\begin{bmatrix}\mathbf{1} & \mathbf{1} & \mathbf{1}\\ \mathbf{2} & \mathbf{2} & \mathbf{2}\\ \mathbf{3} & \mathbf{3} & \mathbf{3}\end{bmatrix}, \boldsymbol{B}=\begin{bmatrix}\mathbf{6} & \mathbf{6} & \mathbf{6}\\ \mathbf{7} & \mathbf{7} & \mathbf{7}\\ \mathbf{8} & \mathbf{8} & \mathbf{8}\end{bmatrix}$	$A+B=\begin{bmatrix}7 & 7 & 7\\ 9 & 9 & 9\\ 11 & 11 & 11\end{bmatrix}$	$img1 + img2$

2.4.3 Linear vs Non-Linear Operations

An operation β is called a linear operation if it fulfills following properties:

$$\beta[af_1(x,y)+bf_2(x,y)] = a\beta f_1(x,y)+b\beta f_2(x,y) = ag_1(x,y)+bg_2(x,y) \quad (\mathbf{2-10})$$

There are two properties that must be satisfied for linear operation.

- $Additivity$: This equation tells that an operation applied on sum of two terms must be equal to applying that operation on individual term and them summing them.
- $Homegeneity$: the output of a linear operation on a constant multiplied by an input is the same as the output of the operation due to the original input multiplied by that constant

If an operation fails to satisfy these two conditions, is called the $non-linear$ operation

2.4.3.1 Arithmetic Operations

Followings are few arithmetic operations that can be performed on two images $f(x,y)$ and $g(x,y)$ (matrix operations)

Operation	Mathematical Equation
Sum	$S(x,y)=f(x,y)+g(x,y)$
Difference	$D(x,y)=f(x,y)-g(x,y)$
Product	$P(x,y)=f(x,y)*g(x,y)$
Division	$V(x,y)=f(x,y)\div g(x,y)$

An arithmetic operation can be used to extract different features from the images. For example, removing noise from noisy images and finding the change in an image. Image subtraction is normally used in mask mode radiography in which first a patient is imaged and this image is named as mask. Then a contrast agent is injected into the body of patient and patient is imaged again. The new image is then subtracted from original image (image taken without injection) that gives the change caused due to contrast agent.

<Write a program to compare two images using image subtraction. Read an image from user and create its two copies, create some noise using numpy and add this noise to a copy of original image. Now subtraction noisy image from actual image. It will give you the change in image>

<write a program to find the negative of an image>

<get two images one without contrast enhancement and other after injecting contrast agent. Subtract them to find the change in image>

Image multiplication and division can be used for shading correction present in an image.

2.4.3.2 Set Operations

A set is collection of distinct objects. If we denote a set with S and its object with ai where i is the index of elements then a set can be written as $S = \{a_0, a_1, a_2, \dots, a_n\}$. Following list shows different concepts associated with sets.

- $Set\ membership$: $a_i \in S$ denotes that element a_i belongs to set S and $a_i \notin S$ denotes that a_i does not belong to set S
- $Cartesian\ Product$: It is defined over two sets A,B and is a set of all ordered pairs in which first element of each pair is from set A and second element is from set B.
$$AXB = \{(a,b)|a \in A, b \in B\} \quad (2-11)$$
- $Subset\ of\ a\ set$: If every element of set T is member of Set S then T is said to be the subset of S and denoted as $T \subseteq S$.
- $Union$: Union of two sets A and B is a set that contains all elements of both sets and denoted as $A \cup B$. For images union operations can be defined as
$$A \cup B = \{\max_z (A,B)|a \in A, b \in B\} \quad (2-12)$$
- $Intersection$: Intersection of two sets A and B is a set that contains elements which are common in both sets and denoted as $A \cap B$.
$$A \cap B = \{\min_z (A,B)|a \in A, b \in B\} \quad (2-13)$$
- $Disjoint\ or\ Mutually\ Exclusive$: If two sets A and B have no common element, they are called disjoint sets and their intersection is empty set $A \cap B = \emptyset$.
- $Sample\ Space\ \Omega$: It is the set of all possible subsets for a given set S. For an image, the sample space is set of all pixels of that image.
- $Set\ Complement$: complement of set A is all the elements that are not in A.
$$A^c = \{w|w \notin A\} \quad (2-14)$$
- Set Difference: The difference of two sets A, B consists of all elements that are in A and are not in B.
$$A - B = \{w|w \in A \ \wedge w \notin B\} \quad (2-15)$$
- There are some other set operations that are listed in following table.

Law Name	Mathematical Notation
Commutative Law	$A \cap B = B \cap A \quad (2-16)$ $A \cup B = B \cup A \quad (2-17)$
Associative Law	$(A \cup B) \cup C = A \cup (B \cup C) \quad (2-18)$ $(A \cap B) \cap C = A \cap (B \cap C) \quad (2-19)$
Distributive Law	$A \cup (B \cap C) = (A \cup B) \cap (A \cup C) \quad (2-20)$ $A \cap (B \cup C) = (A \cap B) \cup (A \cap C) \quad (2-21)$
DeMorgan's Law	$(A \cup B)^C = A^C \cap B^C \quad (2-22)$ $(A \cap B)^C = A^C \cup B^C \quad (2-23)$

<write a program in python that reads three images and verify the above given law visually. Ie. It performs these operations on images and shows that LHS=RHS>

2.4.3.3 Logical Operations

The operations that results in TRUE or FALSE are called the logical operations. An image generated through logical operations is a binary image. The pixel with values TRUE are shown as white and those with value FALSE are shown as black. A set containing pixels with TRUE value forms the $foreground$ of image and the set with FALSE value pixels forms the $background$ on the image. To perform logical operations two images are required (Except Not operation). An image with varying values of pixels can be considered as binary image if we consider zero values as FALSE and non-zero values as TRUE. Few of the logical operations are listed below:

Logical operation	Description
AND	$A\ AND\ B$: The result is the set of pixels that is defined as $$A \cap B = \{\min_{z}\ (A,B) \mid a \in A, b \in B\}$$
OR	$A\ OR\ B$: This operation is similar to union operation of images $$A \cup B = \{\max_{z}\ (A,B) \mid a \in A, b \in B\}$$
NOT	$NOT\ A$: Zero values in the image A are converted to non-zero values and vice versa

<write a program to perform logical operations on input images>

2.4.3.4 Spatial Operations

These are the operations that are performed directly on the values of image pixels. The operations that we have seen till now are all spatial operations. These operations can be categorized as:

- Single pixel operations
- Neighborhood operations
- Geometric spatial transformations

Single Pixel Operations (SPO)

Single pixel operations are performed on each pixel separately without considering value of other pixels in the image. The operation is denoted as $s = T(z)$. Where T some operation (transformation). For example, if A represents an image, then $2 * A$ operation multiplies each pixel with value 2 that is a transformation.

Neighborhood Operations (NHO)

These are the operations that consider the pixel neighborhood to transform it to some new value. This may be N_4, N_8 or any other user defined. As an example, let we define a window of 8 pixels that is used to find the new value of its center pixel. The NHO transformation could be as follows:

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 3 \\ 4 & 4 & 4 & 4 \end{bmatrix} \rightarrow \begin{bmatrix} \mathbf{0} & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 2 & 2 & 2 & 2 & 0 \\ 0 & 3 & 3 & 3 & 3 & 0 \\ 0 & 4 & 4 & 4 & 4 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Table 3: A) Matrix B) Matrix with zero padding

1	1	1	1
2	2	2	2
3	3	3	3
4	4	4	4

0	0	0	0	0	0
0	1	1	1	1	0
0	2	2	2	2	0
0	3	3	3	3	0
0	4	4	4	4	0
0	0	0	0	0	0

Averaging Window	Resulting Matrix	Averaging window	Resulting Matrix
0 0 0 0 1 1 0 2 2	6/9= 0.67	0 0 0 1 1 1 2 2 2	0.67 9/9=1
0 0 0 1 1 1 2 2 2	0.67 1 9/9=1	0 0 0 1 1 0 2 2 0	0.67 1 1 0.67
0 1 1 0 2 2 0 3 3	0.67 1 1 0.67 1.33	1 1 1 2 2 2 3 3 3	0.67 1 1 0.67 1.33 2

First, we padded the original matrix with zeros by introducing zero rows and zero-columns. Each 3x3 windows is averaged using formula (indexing is done when window is extracted from image and re-indexed):

$$g(x,y) = \left(\frac{1}{3X3}\right)\sum_1^3\sum_1^3 f(x,y) = \frac{1}{9}\sum_1^3\sum_1^3 f(x,y) \qquad (2-24)$$

By following this pattern, we can generate the final matrix that is

0	0	0	0	0	0

0	1	1	1	1	0
0	2	2	2	2	0
0	3	3	3	3	0
0	4	4	4	4	0
0	0	0	0	0	0

0.67	1	1	0.67
1.33	2	2	1.33
2	3	3	2
1.56	2.33	2.33	1.56

<write a program that generates the output image using 3x3 averaging, mean and mode window on input image>

Geometric Operations (GO)

These are also known as rubber sheet transformation. The name is given as application of such transformation do not bring any change in structure of image however the shape of image is changed. Geometric transformations of digital images consist of two basic operations:

1. Spatial transformation of coordinates.
2. Intensity interpolation that assigns intensity values to the spatially transformed pixels.

Such a transformation can be expressed as:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = T \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} t_{11} & t_{12} \\ t_{21} & t_{22} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \qquad (2-25)$$

Where (x,y) are pixel coordinates in original image and (x',y') are transformed coordinates.

We will be mostly interested in affine transformations that include scaling, translation, rotation and sheering. It is the property of such transformation that they maintain the positions of points, corners, straight lines and planes. It is possible to use homogeneous coordinates to express all four affine transformations. It is done by using a single 3x3 matrix as given below:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = A \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} t_{11} & t_{12} & t_{13} \\ t_{21} & t_{22} & t_{23} \\ t_{31} & t_{32} & t_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \qquad (2-26)$$

This transformation can handle all four operations i.e. scaling, rotation, translation and shearing depending upon the values chosen for matrix A. In following table, we provide for this affine matrix A to implement different geometric operations.

Transformation Name	Affine Matrix (A)	Coordinate Equations
Identity	$\begin{matrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{matrix}$	$x' = x$ $y' = y$
Scaling / Reflection	$\begin{matrix} c_x & 0 & 0 \\ 0 & c_y & 0 \\ 0 & 0 & 1 \end{matrix}$	$x' = c_x x$ $y' = c_y y$

Rotation	$\begin{matrix} cos\theta & -sin\theta & 0 \\ sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{matrix}$	$x' = xcos\theta - sin\theta$ $y' = xsin\theta + ycos\theta$
Translation	$\begin{matrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{matrix}$	$x' = x + t_x$ $y' = y + t_y$
Shear (Vertical)	$\begin{matrix} 1 & s_v & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{matrix}$	$x' = x + s_v y$ $y' = y$
Shear (horizontal)	$\begin{matrix} 1 & 0 & 0 \\ s_h & 1 & 0 \\ 0 & 0 & 1 \end{matrix}$	$x' = x$ $y' = s_h x + y$

<write a python program to perform all four transformations on input image>

2.4.3.5 Image Registration

Image registration is used to align two or more images of same scene. In this process, we have a $reference$ image that is used to align the $input$ images. In affine transformations, the mapping function is $known$ however in image registration such a function is $not - known$ and need to be estimated. The registration is required in two cases usually:

- Image registration for images taken at same time by using different instruments. Examples of image registration include aligning two or more images taken at approximately the same time, but using different imaging systems, such as an MRI (magnetic resonance imaging) scanner and a PET (positron emission tomography) scanner.
- Or, perhaps the images were taken at different times using the same instruments, such as satellite images of a given location taken several days, months, or even years apart.

In either case, combining the images or performing quantitative analysis and comparisons between them requires compensating for geometric distortions caused by differences in viewing angle, distance, orientation, sensor resolution, shifts in object location, and other factors.

A good approach is to use the control points ($tie - points$) in both images. $Tie - points$ are the points in both images whose locations are precisely known. Researchers have developed algorithms to find such $tie - points$ automatically. Even some imaging systems have physical artifact embedded in the imaging system such as pins, screws. Such embedded points are known as $reseau\ marks$ or $fiducial\ marks$. The known points can be used as guide to mark the $tie - points$ in the image.

<Write a program to register two user given images>

Let we explain the process with the help of an example. Let we have four tie-points in both images as listed below:

Input image	Reference image
$(\boldsymbol{v_1}, \boldsymbol{w_1}), (\boldsymbol{v_2}, \boldsymbol{w_2})(\boldsymbol{v_3}, \boldsymbol{w_3}), (\boldsymbol{v_4}, \boldsymbol{w_4})$	$(x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4)$

And let the mapping between these corresponding points is given by following model:

X points in reference image	Y points in reference image	
$\boldsymbol{x_1 = c_1 v_1 + c_2 w_1+ c_3 v_1 w_1 + c_4}$	$y_1 = c_5 v_1 + c_6 w_1 + c_7 v_1 w_1 + c_8$	$(\mathbf{2} - 27)$
$\boldsymbol{x_2 = c_1 v_2 + c_2 w_2 + c_3 v_2 w_2 + c_4}$	$y_2 = c_5 v_2 + c_6 w_2 + c_7 v_2 w_2 + c_8$	
$\boldsymbol{x_3 = c_1 v_3 + c_2 w_3 + c_3 v_3 w_3 + c_4}$	$y_3 = c_5 v_3 + c_6 w_3 + c_7 v_3 w_3 + c_8$	
$\boldsymbol{x_4 = c_1 v_4 + c_2 w_4 + c_3 v_4 w_4 + c_4}$	$y_4 = c_5 v_4 + c_6 w_4 + c_7 v_4 w_4 + c_8$	

By solving these equations, we can easily find the values of coefficients cs. Once these coefficients are found, all of input image pixels are transformed to reference image and the result is the desired registered image. The correctness of result is based on selection of correct tie-points. If the image is complex, it can be broken sub parts and each sub part is worked independently. Following is good example of registration taken from Gonzalez's Book (Digital Image Processing, FOURTH EDITION, Rafael C. Gonzalez • Richard E. Woods).

Table 4: A) Reference Image B) Distorted Input Image C) Registered Input Image D) Difference of A and C

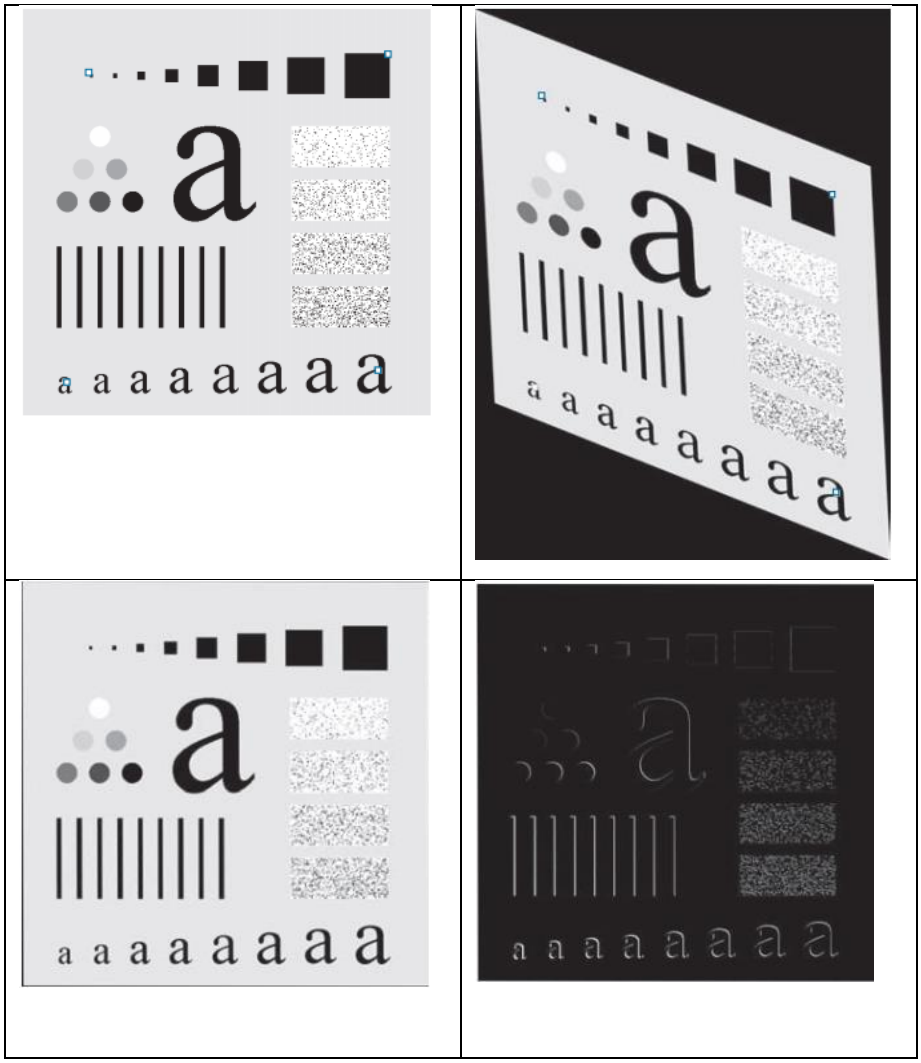


Figure 2-11: Image Registration

<Write a program that take a reference image and geometrically distorted image. Use the registration formula as given above to register the input image with reference image.>

2.4.3.6 Vector and Matrix Operations

A color image consists of three channels and could be considered as 3D image. There are image formats that can contain more than three channels like MRI slices that may be more than 100 depending upon the capability of imaging device. A multi-spectral (MS) image is one that is captured using multiple frequencies of EM spectrum. In such MS images, vector and matrix operations are used routinely. In order to understand the vector operations, we consider color images that contain 3 spectral parts i.e. red, green and blue. Each pixel in color image, therefore, contains 3 values: $f(x,y) = \left(I_{red}, I_{green}, I_{blue}\right)^T$ or equivalently it can be represented as $\boldsymbol{z} = [z1\ z2\ z3]^T$. We write it as transpose to clear that it is not spatial representation but transposed version of that representation. Pixel value in an image with n slices can be represented as $\boldsymbol{z} = [z_1\ z_2\ \ z_3 \dots\ z_n]^T$

1. *Inner product* is defined as $\boldsymbol{a} \cdot \boldsymbol{b} = \boldsymbol{a^T b} = a_1 b_1 + a_2 b_2 + \cdots . + a_n b_n = \sum_i a_i b_i \quad (\mathbf{2-28})$

To make the concept clear, we provide here an example.

Example

Let $A = [1,2,3]$ and $B = [4,5,6]$ then its inner product is given as:

$$A \cdot B = 1 * 4 + 2 * 5 + 3 * 6 = 4 + 10 + 18 = 32$$

Now we solve same using vector notation:

$$A = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \; and \; B = \begin{bmatrix} 4 \\ 5 \\ 6 \end{bmatrix}$$

$$A^T \cdot B = [1 \quad 2 \quad 3] \cdot \begin{bmatrix} 4 \\ 5 \\ 6 \end{bmatrix}$$

$$= 1 * 4 + 2 * 5 + 3 * 6 = 4 + 10 + 18$$

$$= 32$$

Vector Norms

A norm is a function that assign positive value to a vector this value is length or size of vector. There are multiple types of norms which are used in vector processing.

2. $Absolute\ Value\ Norm\ (L1\ Norm)$: It is the norm for one dimensional vector. If $\boldsymbol{x}$ is some vector, Absolute value norm is denoted as $\|\boldsymbol{x}\|$ or $|\boldsymbol{x}|$.
3. $Euclidean\ Norm$: for some vector $\boldsymbol{x}$ it is denoted as

$$\|\boldsymbol{x}\|_2 = |\boldsymbol{x}|_2 = \sqrt{x_1^2 + x_2^2 + x_3^2 + \cdots + +x_n^2} \qquad (2-29)$$

Three are different names used for Euclidean Norm that may include Euclidean Length, L^2 Distance and L^2 norm. A Euclidean norm is square root of inner product (length of vector $\boldsymbol{z}$) that is

$$\|\boldsymbol{z}\| = (\boldsymbol{z}^T\boldsymbol{z})^{\frac{1}{2}} \qquad \boldsymbol{(2-29)}$$

4. $Euclidean\ Distance$ between two vectors $(\boldsymbol{z}, \boldsymbol{a})$ can be written as

$$D(\boldsymbol{z}, \boldsymbol{a}) = \|\boldsymbol{z} - \boldsymbol{a}\| = [(\boldsymbol{z} - \boldsymbol{a})^T(\boldsymbol{z} - \boldsymbol{a})]^{\frac{1}{2}} \qquad \boldsymbol{(2-30)}$$

$$= [(z_1 - a_1)^2 + (z_2 - a_2)^2 + \cdots . + (z_n - a_n)^2]^{\frac{1}{2}}$$

2.4.3.7 Image Transforms

The approaches discussed till now are based on spatial domain processing. There are situations when image processing can become efficient by transforming the image into some other domain (transform domain). You can consider a transform domain as way of image representation. For example, you can represent the location of a pixel in an image either using rectangular coordinates (x, y) or polar coordinates (r, θ) where

$$\boldsymbol{r^2 = x^2 + y^2} \quad and \quad \boldsymbol{\theta = \tan^{-1}\frac{y}{x}} \qquad \boldsymbol{(2-31)}$$

And

$$\boldsymbol{x = rcos\theta} \quad and \quad \boldsymbol{y = rsin\theta} \qquad \boldsymbol{(2-32)}$$

By having this knowledge, we look at general form of $transform\ domain$. Let $f(x, y)$ is an image in spatial domain and (u, v) are coordinates or transform variables of transform domain $T(u, v)$.

$$T(u,v) = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)r(x,y,u,v) \qquad (2-33)$$

Here you can see that for every pixel of image, pixel value $f(x,y)$ is transformed using $r(x,y,u,v)$ and then multiplied with original image pixel value. Such transformation-product values are summed for all rows and columns of image. You can observe transformed value of a pixel is different from its corresponding value in original spatial domain. This is called the ***forward transformation*** and $r(x,y,u,v)$ is called the ***forward transformation kernel***.

By rearranging the equation, we can obtain pixel original value $f(x,y)$ from $T(u,v)$ as follows:

$$f(x,y) = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} T(u,v)s(x,y,u,v) \qquad (2-34)$$

where $s(x,y,u,v)$ is called an ***inverse transformation kernel.*** Both equations combinedly are named as ***transform pair***. There are number of transform domains that may include $Walsh,\ Hadamard,\ slant,$ $discrete\ cosine\ and\ \ Haar$ transforms. You will encounter may such transformation when working with image processing. Although there are built in software routines available however being a student of digital image processing course, you should try to code them yourself.

2.4.3.8 Separable Kernel

A kernel (vector or matrix) can be written as product of two vectors/matrices, it is called separable kernel.

$$r(x,y,u,v) = r_1(x,u)r_2(y,v) \qquad (2-35)$$

A kernel is said to be $symmetric$ if $r_1(x,u)$ is functionally equivalent to $r2(y,v)$ i.e.

$$r(x,y,u,v) = r_1(x,u)r_1(y,v) \qquad (2-36)$$

Example:

$$\begin{bmatrix}1 & 0 & -1\\ 2 & 0 & -2\\ 1 & 0 & -1\end{bmatrix} = \begin{bmatrix}1\\ 2\\ 1\end{bmatrix} * \begin{bmatrix}1 & 0 & -1\end{bmatrix}$$

2.4.3.9 Image Intensities as Random Variables

Probability theory is very useful tool when working with random experiments. If we consider value of each pixel in an image independent of other pixels, a random variable can be used to represent image pixels. Here we list few basic concepts of probability in context of image processing. Let there be L levels of intensity in an image and Z_i represents the intensity value of a pixel for L levels where $i = 0,1,2,3,\dots,L-1$. If there are M rows and N columns in an image then.

5. $Probability\ of\ z_i\ level$

$$p(z_i) = \frac{n_i}{MN} \qquad (2-37)$$

where n_i is number of pixels having intensity level i.

6. $Total\ probability$ of all intensity levels.

$$\sum_i \frac{n_i}{MN} = 1 \qquad (2-38)$$

7. $Image\ mean\ intensity$

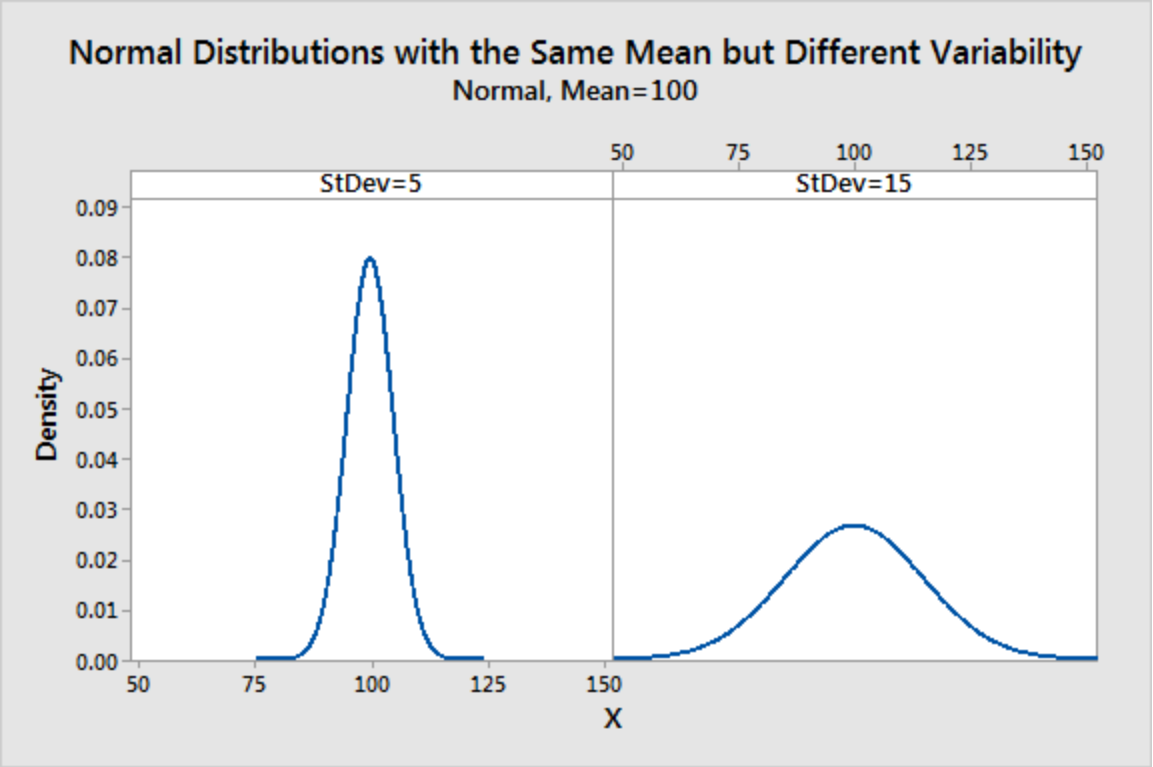
$$m = \sum_i z_i p(z_i) \quad (2-39)$$

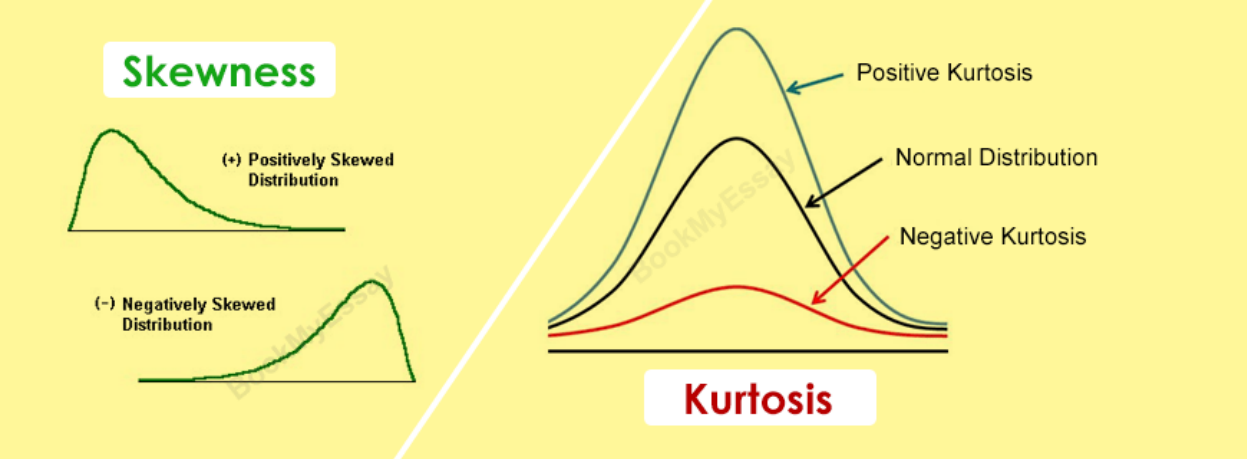
8. *variance of image intensities*

$$\sigma^2 = \sum_i (z_i - m)^2 \; p(z_i) \quad (2-40)$$

Central moments of a random variable are very useful measure that provide good insight of image intensity distributions. Center moment is quantitative measure of shape of a function about its mean value. Image moments are useful to describe objects after segmentation. Simple properties of the image which are found via image moments include area (or total intensity), its centroid, and information about its orientation. Following table lists few moments with their description.

Moment	**Formula**	**Description**
0th	$\mu_0 = \sum_i (z_i - m)^0 \; p(z_i)$ = 1	$\mu_0 = \sum_i 1 * \; p(z_i)$ $= \sum_i p(z_i) = 1$
1st	$\mu_1 = \sum_i (z_i - m)^1 \; p(z_i)$ = 0	For a standard die roll $\mu_1 = \frac{1}{6}[(1-3.5) + (2-3.5) + (3-3.5) + (4-3.5) + (5-3.5) + (6-3.5)$ $== \frac{1}{6}[(-2.5) + (-1.5) + (-0.5) + (0.5) + (1.5) + (2.5)]$ =0
2nd	$\mu_2 = \sum_i (z_i - m)^2 \; p(z_i)$	This moment finds the variance of image intensity distribution
3rd	$\mu_3 = \sum_i (z_i - m)^3 \; p(z_i)$	It finds the skewness of random variable distribution
4th	$\mu_4 = \sum_i (z_i - m)^4 \; p(z_i)$	It finds the kurtosis of random variable distribution
Nth	$\mu_n = \sum_i (z_i - m)^n \; p(z_i)$	





<write a python program to find nth moment of given image. Also draw the distribution of image pixel intensities and discuss your quantitative results with visual results>

3 Exercises

Q.No.1 Consider the following matrices

$$A = \begin{bmatrix} 0 & 3 & 1 & 3 \\ 2 & 1 & 2 & 3 \\ 3 & 0 & 0 & 1 \\ 1 & 0 & 2 & 3 \end{bmatrix} \qquad B = \begin{bmatrix} 0 & 3 & 1 & 3 \\ 2 & 1 & 2 & 3 \\ 3 & 0 & 0 & 1 \\ 1 & 0 & 2 & 3 \end{bmatrix}$$

Let $L = \{0,1,2,3\}$ then if $V_1 = \{0,1\}, V_2 = \{0,1,2\}, V_3 = \{0,1,2,3\}$ write a program to find the m-adjacency of circled pixels