MODULE VI

Image processing – Introduction - Fundamental steps in image processing – digital image representations – relationship between pixels – gray level histogram –spatial convolution and correlation – edge detection – Robert, Prewitt, Sobel.

Introduction

What Is Digital Image Processing?

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*. The field of *digital image processing* refers to processing digital images by means of a digital computer. Note that a digital image is composed of a finite number of elements, each of which has a particular location and value. These elements are referred to as *picture elements*, *image elements*, *pels*, and *pixels*. ***Pixel*** is the term most widely used to denote the elements of a digital image.

Fundamental Steps in Digital Image Processing

There are some fundamental steps but as they are fundamental, all these steps may have sub-steps. The fundamental steps are described below with a neat diagram.

1. Image Acquisition:

This is the first step or process of the fundamental steps of digital image processing. Image acquisition could be as simple as being given an image that is already in digital form. Generally, the image acquisition stage involves pre-processing, such as scaling etc.

2. 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. Such as, changing brightness & contrast etc.

3. 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.

4. Color Image Processing:

Color image processing is an area that has been gaining its importance because of the significant increase in the use of digital images over the Internet. This may include color modeling and processing in a digital domain etc.

5. Wavelets and Multi-Resolution Processing:

Wavelets are the foundation for representing images in various degrees of resolution.Images subdivision successively into smaller regions for data compression and for pyramidal representation.

6. Compression:

Compression deals with techniques for reducing the storage required to save an image or the bandwidth to transmit it. Particularly in the uses of internet it is very much necessary to compress data.

7. Morphological Processing:

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

8. 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.

9. Representation and Description:

Representation and description almost always follow the output of a segmentation stage, which usually is raw pixel data, constituting either the boundary of a region or all the points in the region itself. Choosing a representation is only part of the solution for transforming raw data into a form suitable for subsequent computer processing. Description deals with extracting attributes that result in some quantitative information of interest or are basic for differentiating one class of objects from another.

10. Object recognition:

Recognition is the process that assigns a label, such as "vehicle" to an object based on its descriptors.

11. Knowledge Base:

Knowledge may be as simple as detailing regions of an image where the information of interest is known to be located, thus limiting the search that has to be conducted in seeking that information. The knowledge base also can be quite complex, such as an interrelated list of all major possible defects in a materials inspection problem or an image database containing high-resolution satellite images of a region in connection with change-detection applications.

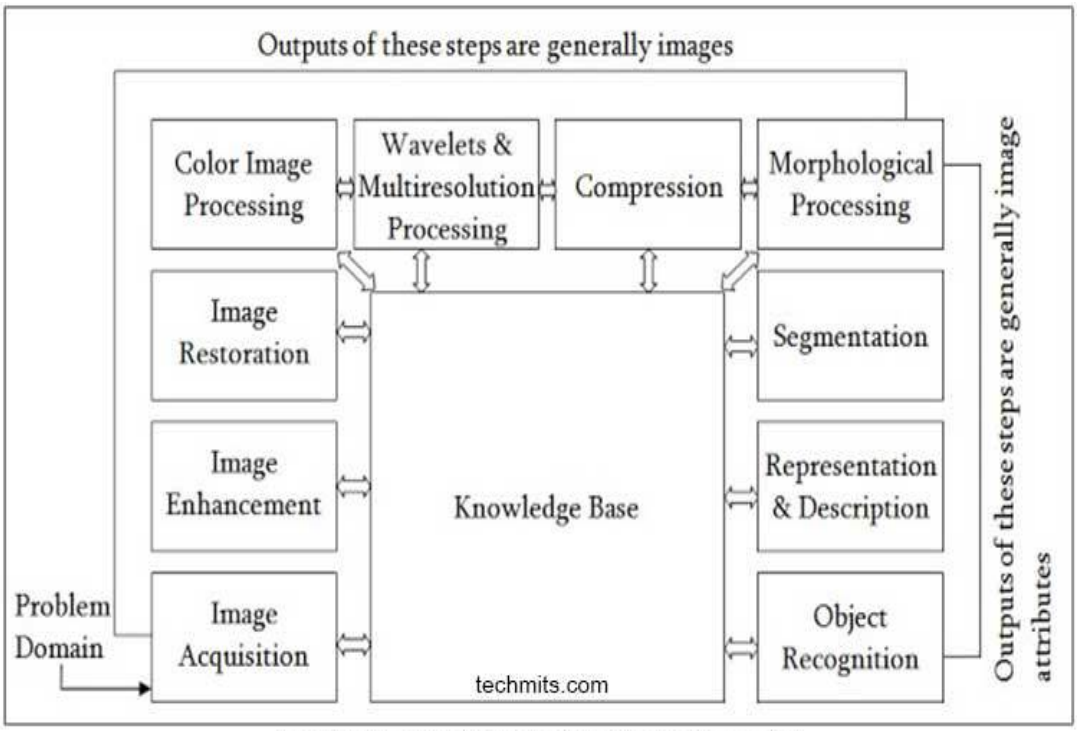


Fig: Fundamental Steps in Digital Image Processing

Types of Digital Images

1. Binary Image
2. Gray-scale Image
3. Color image

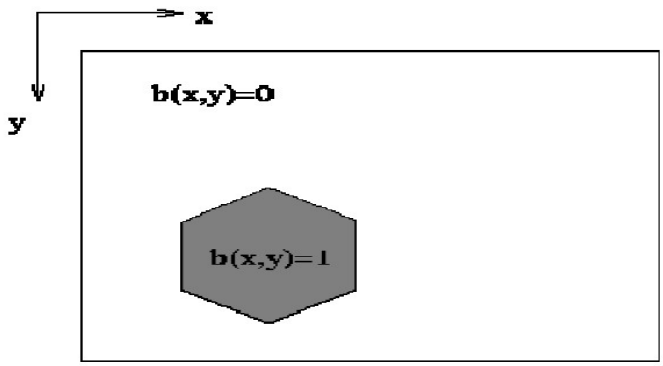
Binary Image

Binary images are the simplest type of images and can take on two values, typically black and white, or '0' and '1'. A binary image is referred to as a 1 bit/pixel image because it takes only 1 binary digit to represent each pixel. These types of images are most frequently in computer vision application where the only information required for the task is general shapes, or outlines information. For example, to position a robotics gripper to grasp an object or in optical character recognition (OCR).

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

We define the characteristic function of an object in an image to be

$$b(x,y)\begin{cases}=1 & \text{for points on the object}\\ =0 & \text{for background points.}\end{cases}$$



(a)

(b)

(a) binary image representation (b) binary Lenna image

Each pixel is stored as a single bit (0 or 1).

A 640 x 480 monochrome image requires 37.5 KB of storage.

Gray Scale Images

Gray scale images are referred to as monochrome, or one-color image. They contain brightness information only, no color information. **The number of different brightness level available. The typical image contains 8 bit/ pixel (data, which allows us to have (0-255) different brightness (gray) levels)**. The 8 bit representation is typically due to the fact that the byte, which corresponds to 8-bit of data, is the standard small unit in the world of digital computer.

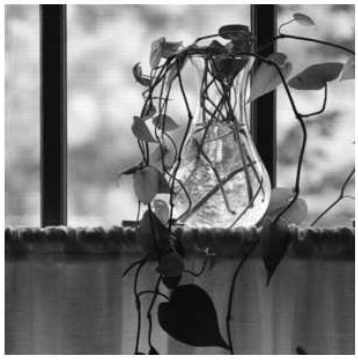




Figure Examples of gray-scale images

Each pixel is usually stored as a byte (value between 0 to 255). A 640 x 480 grayscale image requires over 300 KB of storage.

COLOR IMAGES

Representation of color images is more complex and varied. The two most common ways of storing color image contents are

1)RGB representation—in which each pixel is usually represented by a 24-bit number containing the amount of its red (R), green (G), and blue (B) components.

2) **indexed representation**—where a 2D array contains indices to a color palette (or lookup table - (LUT)).

24-Bit (RGB) Color Images Color images can be represented using three 2D arrays of same size, one for each color channel: red (R), green (G), and blue (B) as shown in the following figure. Each array element contains an 8-bit value,indicating the amount of red, green, or blue at that point in a [0, 255] scale. The combination of the three 8-bit values into a 24-bit number allows 224 (16,777,216, usually referred to as 16 million or 16 M) color combinations. An alternative representation uses 32 bits per pixel and includes a fourth channel, called the alpha channel, that provides a measure of transparency for each pixel and is widely used in image editing effects.

The following figure we see a representation of a typical RGB color image.

Color image (a) and its R (b), G (c), and B (d) components.

Example of 24-Bit Colors Image

Each pixel is represented by three bytes (e.g., RGB).It Supports 256 x 256 x 256 possible combined colors (16,777,216).A 640 x 480 24-bit color image would require 921.6 KB of storage.

Indexed Color Images: A problem with 24-bit color representations is backward compatibility with older hardware that may not be able to display the 16 million colors simultaneously. A solution—devised before 24-bit color displays and video cards were widely available—consisted of an indexed representation, in which a 2D array of the same size as the image contains indices (pointers) to a *color palette* (or *color map*) of fixed maximum size (usually 256 colors). The color map is simply a list of colors used in that image. Following figure shows an indexed color image

and a 4 × 4 detailed region, where each pixel shows the index and the values of R, G, and B at the color palette entry that the index points to.

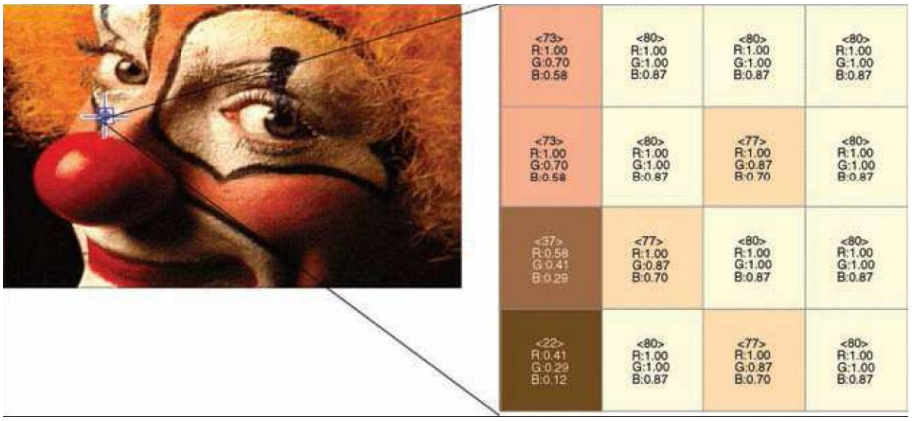
Example of 8-Bit Color Image

□□One byte for each pixel

□□Supports 256 out of the millions s possible, acceptable color quality

□□Requires Color Look-Up Tables (LUTs)

□□A 640 x 480 8-bit color image requires 307.2 KB of storage (the same as 8-bit grayscale)



An indexed color image and the indices in a 4 × 4 neighborhood. Original image

Digital image representation

There are three types of digital image creation:

Vector Images

Bitmaps

Procedural Modelling

Vector images

One way to describe an image using numbers is to declare its contents using position and size of geometric forms and shapes like lines, curves, rectangles and circles; such images are called vector images.

Vector graphics is an image file format suitable for pictures with areas of solid, clearly separated colors— cartoon images, logos, and the like. Instead of being painted pixel by pixel, a vector graphic image is drawn object by object in terms of each object's geometric shape. Many file formats for vector graphics—.fh,.ai,.wmf,.eps, etc.—they contain the parameters to mathematical formulas defining how shapes are drawn.

A line can be specified by its endpoints, a square by the length of a side, a rectangle by the length of two sides, a circle by radius.

A vector image is resolution independent, this means that you can enlarge or shrink the image without affecting the output quality. Vector images are the preferred way to represent Fonts, Logos and many illustrations.

Bitmap images

Bitmap, or raster, images are "digital photographs", they are the most common form to represent natural images and other forms of graphics that are rich in detail. Bitmap images are how graphics is stored in the video memory of a computer. The term bitmap refers to how a given pattern of bits in a pixel maps to a specific color.

Raster image

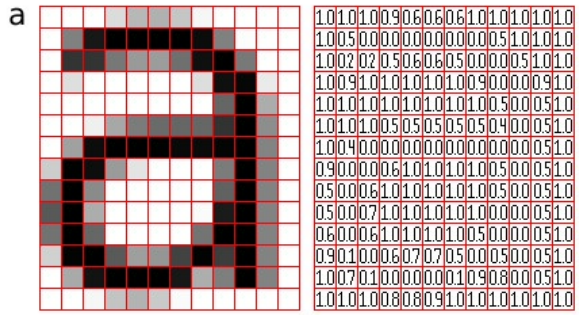
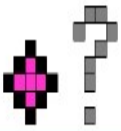


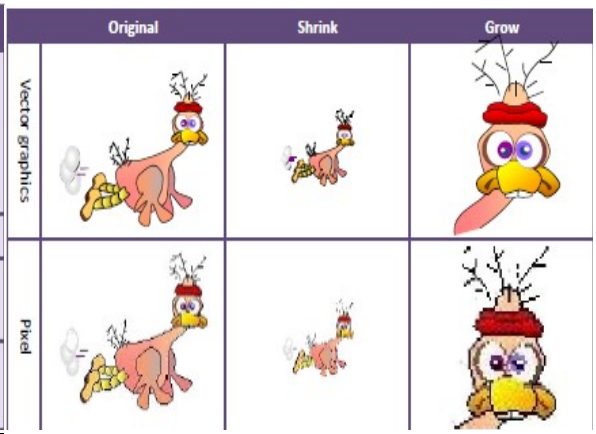
Figure:A rasterized form of the letter 'a' magnified 16 times using pixel doubling

A bitmap images take the form of an array, where the value of each element, called a ***pixel*** picture element, correspond to the color of that portion of the image. Each horizontal line in the image is called a ***scan line***.

The letter 'a' might be represented in a 12x14 matrix as depicted in the above figure the values in the matrix depict the brightness of the ***pixels*** (picture elements). Larger values correspond to brighter areas while lower values are darker.

Bitmaps vs. Vector graphics

	Pixel	Vector
Example		
Grow and shrink	Bad	Good
Speed	Fast to create. Very hard to edit!	Slow to create. Much faster to edit.
Applications	Paint Photoshop	Power Point Illustrator



Relationship between pixels

In this section, we consider several important relationships between pixels in a digital image. An image is denoted by f(x, y).When referring in this section to a particular pixel, we use lowercase letters, such as p and q.

Neighbors of a Pixel

A pixel p at coordinates (x, y) has four *horizontal* and *vertical* neighbors whose coordinates are given by

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

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

The four *diagonal* neighbors of p have coordinates

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

and are denoted by ND(p). These points, together with the 4-neighbors, are called the 8-*neighbors* of p, denoted by $N8$(p). As before, some of the points in ND(p) and $N8$(p) fall outside the image if (x, y) is on the border of the image.

Adjacency, 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, if their gray levels are equal).For instance, in a binary image with values 0 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 gray-level values used to define adjacency. In a binary image, V={1} if we are referring to adjacency of pixels with value 1. In a grayscale image, the idea is the same, but set *V* typically contains more elements. For example, in the adjacency of pixels with a range of possible gray-level values 0 to 255, set *V* could be any subset of these 256 values. We consider three types of adjacency:

(a) 4-*adjacency*. Two pixels p and q with values from *V* are 4-adjacent if q is in the set *N*4(p).

(b) 8-*adjacency*. Two pixels p and q with values from *V* are 8-adjacent if q is in the set *N*8(p).

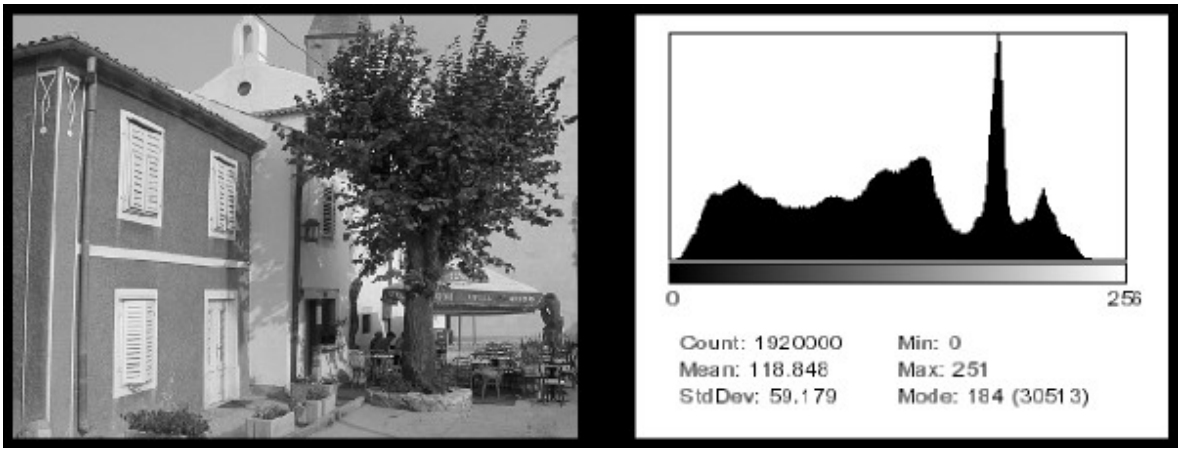
(c) *m-adjacency* (mixed adjacency).Two pixels p and q with values from *V* are madjacent if

(i) q is in *N*4(p), *or*
(ii) q is in *N*D(p) *and* the set has no pixels whose values are from *V*.

Gray Level Histogram

In Statistics, **Histogram** is a graphical representation showing a visual impression of the distribution of data.
An **Image Histogram** is a type of histogram that acts as a graphical representation of the lightness/color distribution in a digital image. It plots the number of pixels for each value. Histograms plots how many times (frequency) each intensity value in image occurs.

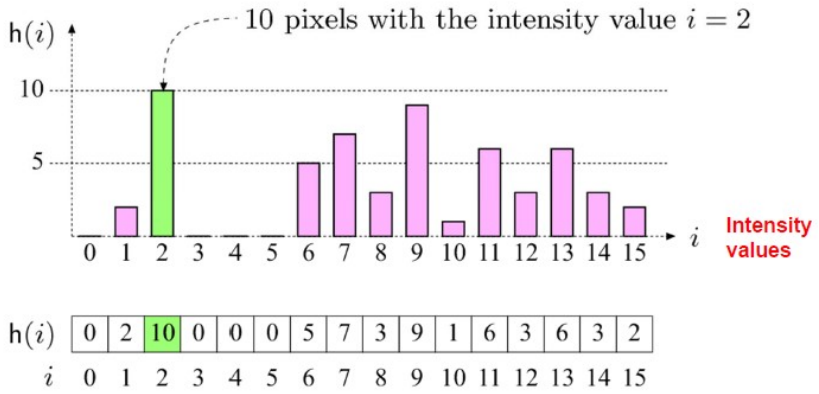
Example:
Image (left) has 256 distinct gray levels (8 bits)
Histogram (right) shows frequency (how many times) each gray level occurs



Why Histograms?

- Histograms are the basis for numerous spatial domain processing techniques
- Histogram manipulation can be used effectively for image enhancement
- Histograms can be used to provide useful image statistics
- Information derived from histograms are quite useful in other image processing applications, such as image compression and 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 kth gray level and n_k is the number of pixels in the image having gray level r_k. The following example shows a gray level histogram with K=16.



From the above histogram we can determine the number of pixels with same intensity value. For example, there are 10 pixels with the intensity value 2.

So, a histogram for a grayscale image with intensity values in range

$$I(u,v) \in [0, K-1]$$

would contain exactly K entries.

E.g. 8-bit grayscale image, $K = 2^8 = 256$

Each histogram entry is defined as:

h(i) = number of pixels with intensity I for all $0 < i < K$.

E.g: h(255) = number of pixels with intensity = 255

Formal definition can be written as:

Formal definition $\mathrm{h}(i) = \mathrm{card}\{(u,v) \mid I(u,v) = i\}$

Number (size of set) of pixels — such that

Histograms are giving only statistical information, but it is not giving any indication of the location of pixels. Different images can have the same histogram. It is not possible to reconstruct the image from the histogram.

Edge Detection

Edge detection is a process that detects the presence and location of edges constituted by sharp changes in intensity of an image. Edges define the boundaries between regions in an image, which helps with segmentation and object recognition. Edge detection of an image significantly reduces the amount of data and filters out useless information, while preserving the important structural properties in an image.

What are edges

We can also say that sudden changes of discontinuities in an image are called as edges. Significant transitions in an image are called as edges.

Types of edges

Generally edges are of three types:

- Horizontal edges
- Vertical Edges
- Diagonal Edges

Why detect edges

Most of the shape information of an image is enclosed in edges. So first we detect these edges in an image and by using these filters and then by enhancing those areas of image which contains edges, sharpness of the image will increase and image will become clearer.

Following are some of the masks for edge detection

- Prewitt operator
- Sobel Operator
- Robinson Compass Masks
- Krisch Compass Masks
- Laplacian Operator.

Above mentioned all the filters are linear filters or smoothing filters.

Prewitt Operator

Prewitt operator is used for edge detection in an image. It detects two types of edges

- Horizontal edges
- Vertical Edges

Edges are calculated by using difference between corresponding pixel intensities of an image. All the masks that are used for edge detection are also known as derivative masks. An image is also a signal so changes in a signal can only be calculated using differentiation. So that's why these operators are also called as derivative operators or derivative masks.

All the derivative masks should have the following properties:

- Opposite sign should be present in the mask.
- Sum of mask should be equal to zero.
- More weight means more edge detection.

Prewitt operator provides us two masks one for detecting edges in horizontal direction and another for detecting edges in vertical direction.

Vertical direction

1	0	1
-1	0	1
-1	0	1

Above mask will find the edges in vertical direction and it is because the zeros column in the vertical direction. When you will convolve this mask on an image, it will give you the vertical edges in an image.

How it works

When we apply this mask on the image it prominent vertical edges. It simply works like as first order derivate and calculates the difference of pixel intensities in an edge region. As the center

column is of zero so it does not include the original values of an image but rather it calculates the difference of right and left pixel values around that edge. This increase the edge intensity and it become enhanced comparatively to the original image.

Horizontal Direction

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

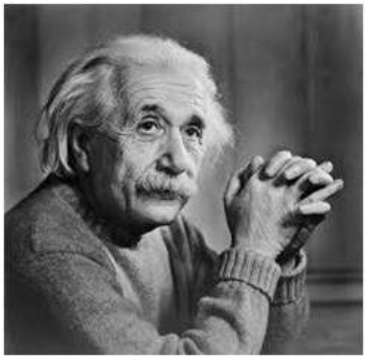
Above mask will find edges in horizontal direction and it is because that zeros column is in horizontal direction. When you will convolve this mask onto an image it would prominent horizontal edges in the image.

How it works

This mask will prominent the horizontal edges in an image. It also works on the principle of above mask and calculates difference among the pixel intensities of a particular edge. As the center row of mask is consist of zeros so it does not include the original values of edge in the image but rather it calculate the difference of above and below pixel intensities of the particular edge. Thus increasing the sudden change of intensities and making the edge more visible. Both the above masks follow the principle of derivate mask. Both masks have opposite sign in them and both masks sum equals to zero. The third condition will not be applicable in this operator as both the above masks are standardize and we can't change the value in them.

Sample Image

Following is a sample picture on which we will apply above two masks one at time.



After applying Vertical Mask

After applying vertical mask on the above sample image, following image will be obtained. This image contains vertical edges. You can judge it more correctly by comparing with horizontal edges picture.

After applying Horizontal Mask

After applying horizontal mask on the above sample image, following image will be obtained.

Comparison

As you can see that in the first picture on which we apply vertical mask, all the vertical edges are more visible than the original image. Similarly in the second picture we have applied the horizontal mask and in result all the horizontal edges are visible. So in this way you can see that we can detect both horizontal and vertical edges from an image.

Sobel Operator

The sobel operator is very similar to Prewitt operator. It is also a derivate mask and is used for edge detection. Like Prewitt operator sobel operator is also used to detect two kinds of edges in an image:

- Vertical direction
- Horizontal direction

Difference with Prewitt Operator

The major difference is that in sobel operator the coefficients of masks are not fixed and they can be adjusted according to our requirement unless they do not violate any property of derivative masks.

Following is the vertical Mask of Sobel Operator:

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

This mask works exactly same as the Prewitt operator vertical mask. There is only one difference that is it has “2” and “-2” values in center of first and third column. When applied on an image this mask will highlight the vertical edges.

How it works

When we apply this mask on the image it prominent vertical edges. It simply works like as first order derivate and calculates the difference of pixel intensities in a edge region.

As the center column is of zero so it does not include the original values of an image but rather it calculates the difference of right and left pixel values around that edge. Also the center values of both the first and third column is 2 and -2 respectively.

This give more weight age to the pixel values around the edge region. This increase the edge intensity and it become enhanced comparatively to the original image.

Following is the horizontal Mask of Sobel Operator

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

Above mask will find edges in horizontal direction and it is because that zeros column is in horizontal direction. When you will convolve this mask onto an image it would prominent

horizontal edges in the image. The only difference between it is that it have 2 and -2 as a center element of first and third row.

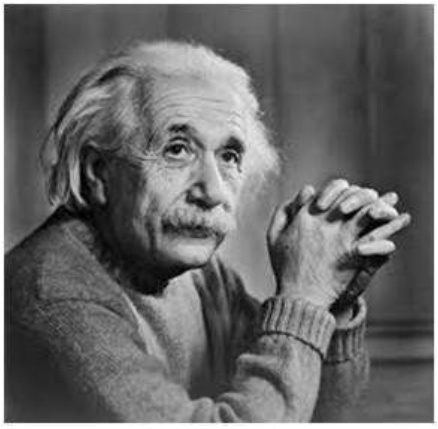
How it works

This mask will prominent the horizontal edges in an image. It also works on the principle of above mask and calculates difference among the pixel intensities of a particular edge. As the center row of mask is consist of zeros so it does not include the original values of edge in the image but rather it calculate the difference of above and below pixel intensities of the particular edge. Thus increasing the sudden change of intensities and making the edge more visible.

Now it's time to see these masks in action:

Sample Image

Following is a sample picture on which we will apply above two masks one at time.



After applying vertical mask and horizontal mask on the above sample image, pictures become

Roberts Cross Edge Detector

It was one of the first edge detectors and was initially proposed by Lawrence Roberts in 1963. As a differential operator, the idea behind the Roberts cross operator is to approximate the gradient of an image through discrete differentiation which is achieved by computing the sum of the squares of the differences between diagonally adjacent pixels.

The Roberts Cross operator performs a simple, quick to compute, 2-D spatial gradient measurement on an image. It thus highlights regions of high spatial frequency which often correspond to edges. In its most common usage, the input to the operator is a grayscale image, as is the output. Pixel values at each point in the output represent the estimated absolute magnitude of the spatial gradient of the input image at that point.

How It Works

In theory, the operator consists of a pair of 2×2 convolution kernels as shown in Figure. One kernel is simply the other rotated by 90°. This is very similar to the Sobel operator.

+1	0
0	-1

Gx

0	+1
-1	0

Gy

These kernels are designed to respond maximally to edges running at 45° to the pixel grid, one kernel for each of the two perpendicular orientations. The kernels can be applied separately to the input image, to produce separate measurements of the gradient component in each orientation (call these *Gx* and *Gy*). These can then be combined together to find the absolute magnitude of the gradient at each point and the orientation of that gradient. The gradient magnitude is given by:

$$|G| = \sqrt{Gx^2 + Gy^2}$$

although typically, an approximate magnitude is computed using:

$$|G| = |Gx| + |Gy|$$

which is much faster to compute.