IAT3 :
DIGITAL IMAGE PROCESSING: 18CS741-7 SEM
(PROFESSIONAL ELECTIVE)

Q1:With a neat block diagram, Explain Transform Coding System.(diagram- 4 marks & explanation- 6 marks)

Ans

Digital Image Processing allows you to apply powerful transform domain techniques to your data. Transform methods are typically used in filtering, compression, and image texture analysis. Transform coding compresses image data by representing the original signal with a small number of transform coefficients.

Procedure

Divide the image into n × n sub-images.

Transform each sub-image using a reversible transform (e.g., the Hotelling transform, the discrete Fourier transform (DFT) or the discrete cosine transform (DCT)).

Quantify, i.e., truncate the transformed image (e.g., by using DFT,and DCT frequencies with small amplitude can be removed without much information

loss). The quantification can be either image dependent (IDP) or image independent (IIP).

Code the resulting data, normally using some kind of "variable length coding", e.g., Huffman code.

The coding is not reversible (unless step 3 is skipped). Divide the image into n × n sub-images.

Transform coding, is a form of block coding done in the transform domain. The image is divided into blocks, or subimages, and the transform is calculated for each block

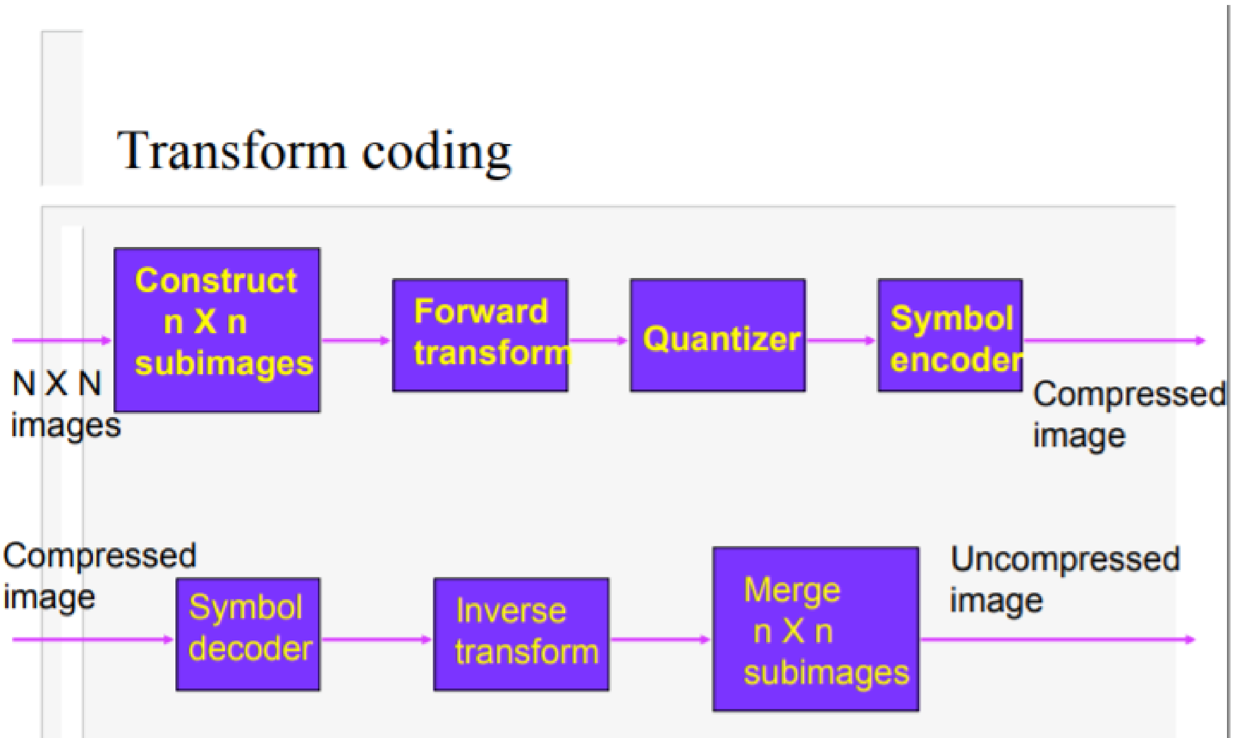
- **Any of the previously defined transforms can be used, frequency (e.g. Fourier) or sequency (e.g. Walsh/Hadamard), but it has been determined that the discrete cosine transform (DCT) is optimal for most images**
- **The newer JPEG2000 algorithms uses the wavelet transform, which has been found to provide even better compression**
-
- **After the transform has been calculated, the transform coefficients are quantized and coded**
- **This method is effective because the frequency/sequency transform of images is very efficient at putting most of the information into relatively few coefficients, so many of the high frequency coefficients can be quantized to 0 (eliminated completely)**
- **This type of transform is a special type of mapping that uses spatial frequency concepts as a basis for the mapping**
- **The main reason for mapping the original data into another mathematical space is to pack the information (or energy) into as few coefficients as possible**

• The simplest form of transform coding is achieved by filtering by eliminating some of the high frequency coefficients

• However, this will not provide much compression, since the transform data is typically floating point and thus 4 or 8 bytes per pixel (compared to the

original pixel data at 1 byte per pixel), so quantization and coding is applied to the reduced data
• Quantization includes a process called bit allocation, which determines the number of bits to be used to code each coefficient based on its importance

• Typically, more bits are used for lower frequency components where the energy is concentrated for most images, resulting in a variable bit rate or nonuniform quantization and better resolution
• Two particular types of transform coding have been widely explored:
1. Zonal coding
2. Threshold coding
• These two vary in the method they use for selecting the transform coefficients to retain (using ideal filters for transform coding selects the coefficients based on their location in the transform domain)



Q2:A:Explain the three types of Gray Level discontinuities in image processing- 5 Marks

There are 3 basic types of discontinuities: **points, lines and edges**.
The three basic types of discontinuities in a digital image are point, line and edge.

Point Detection:

- The detection of isolated point different from constant background image can be done using the following mask:

-1	**-1**	**-1**
-1	8	-1
-1	-1	-1

- A point has been detected at the location on which the mask is centered if |R|>T where T is non negative threshold and.
- The idea is that the gray level of an isolated point will be quite different from the gray levels of its neighbors.
- The mask operation measures the weighted differences between the centre point and its neighbors. The differences that are large enough are determined by T considered isolated points in the image of interest.

Line Detection:

- The various masks present for line detection are :

a) **Horizontal mask**

-1	**-1**	**-1**
2	2	2
-1	-1	-1

b) **-45 degrees**

-1	**-1**	**2**

-1	2	-1
2	-1	-1

c) **45 degrees**

2	**-1**	**-1**
-1	2	-1
-1	-1	2

d) **Vertical mask**

-1	**2**	**-1**
-1	2	-1
-1	2	-1

- If the horizontal mask is moved around an image it would respond more strongly to lines oriented horizontally.
- With constant background the maximum response would result when the line is passing through the middle row of the mask.

Edge Detection:

- Edges characterize object boundaries are therefore useful for segmentation and identification of objects in scene.

- Edge point can be thought of as pixel location of abrupt gray levels. It is the boundary between two regions with relatively distinct gray level properties.
- There are two types of edges step and ramp edge.
- The step edges are detected using first order derivative filters like Robert, Sobel, Frichen and Prewit.
- The ramp edges can be detected using second order derivative line Laplacian filter.

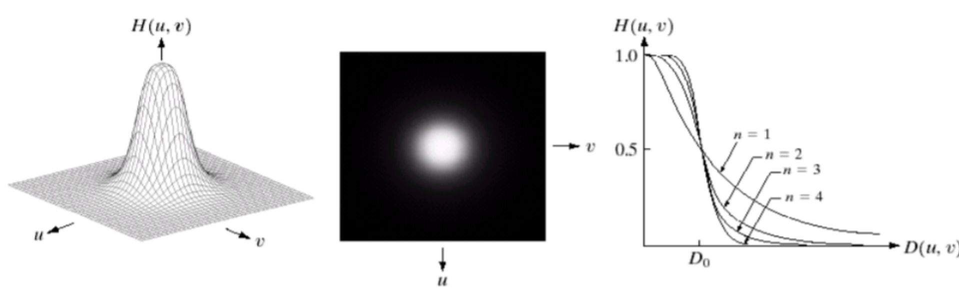
Q2:B:Explain the Butterworth Low pass filter used in frequency domain for image enhancement.is it used for sharpening or smoothing the image. (5 marks- 3+2)

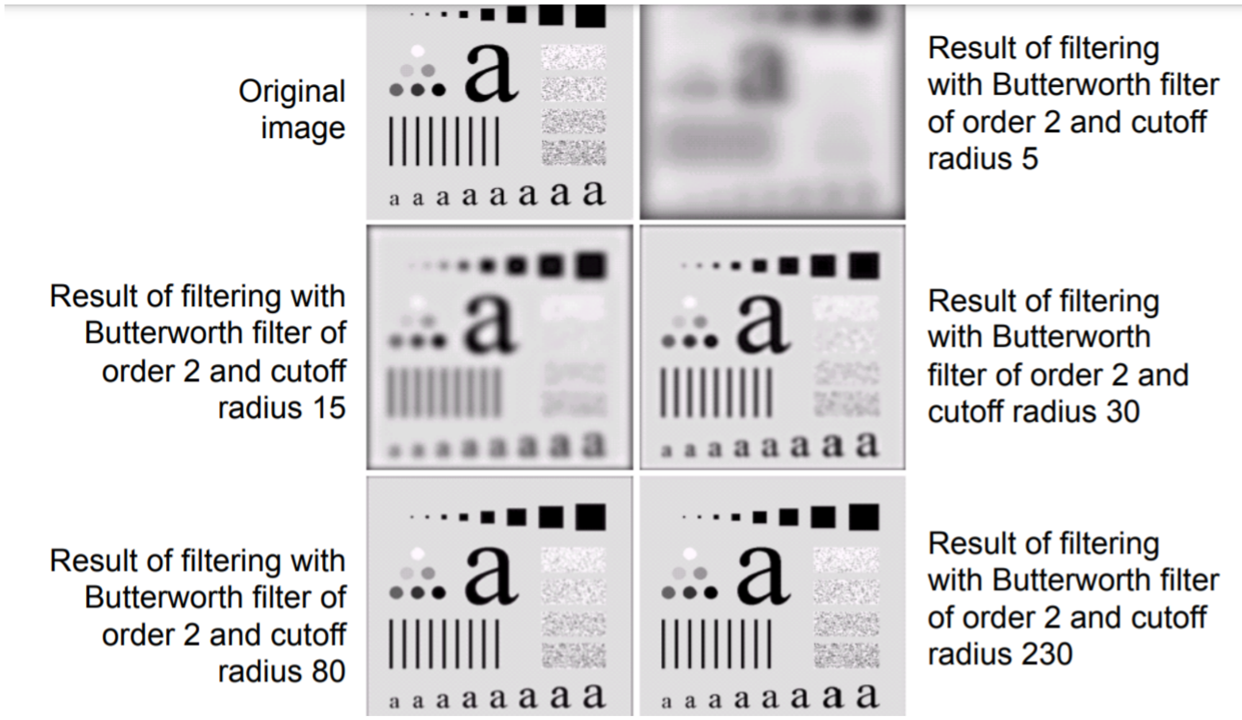
Image **can be smoothed** in the frequency domain by attenuating the high-frequency content of its Fourier transform. This would be a low pass filter. Frequency response does not have a sharp transition as ideal LPF. ... A two-dimensional butterworth low pass filter has transfer function.

In the field of Image Processing, Butterworth Lowpass Filter (BLPF) is **used for image smoothing in the frequency domain**. It removes high-frequency noise from a digital image and preserves low-frequency components.

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

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





Q3:Explain LZW Compression Technique. Is it a Lossy or lossless compression justify your answer with example.(5 marks- 3+2

There are two categories of compression techniques, lossy and lossless. Whilst each uses different techniques to compress files, both have the same aim: To look for duplicate data in the graphic (GIF for LZW) and use a much more compact data representation. Lossless compression reduces bits by identifying and eliminating statistical redundancy. No information is lost in lossless compression. On the other hand, Lossy compression reduces bits by removing unnecessary or less important information. So we need Data Compression mainly because:

- Uncompressed data can take up a lot of space, which is not good for limited hard drive space and internet download speeds.
- While hardware gets better and cheaper, algorithms to reduce data size also help technology evolves.
- Example: One minute of uncompressed HD video can be over 1 GB. How can we fit a two-hour film on a 25 GB Blu-ray disc?

Lossy compression methods include DCT (Discrete Cosine Transform), Vector Quantisation, and Transform Coding while Lossless compression methods include RLE (Run Length Encoding), string-table compression, LZW (Lempel Ziff Welch), and zlib. There Exist several compression Algorithms, but we are concentrating on LZW.

What is Lempel–Ziv–Welch (LZW) Algorithm ?

The LZW algorithm is a very common compression technique. This algorithm is typically used in GIF and optionally in PDF and TIFF. Unix's 'compress' command, among other uses. It is lossless, meaning no data is lost when compressing. The algorithm is simple to implement and has the potential for very high throughput in hardware implementations. It is the algorithm of the widely used Unix file compression utility compress and is used in the GIF image format.

The Idea relies on reoccurring patterns to save data space. LZW is the foremost technique for general-purpose data compression due to its simplicity and versatility. It is the basis of many PC utilities that claim to "double the capacity of your hard drive".

How does it work?

LZW compression works by reading a sequence of symbols, grouping the symbols into strings, and converting the strings into codes. Because the codes take up less space than the strings they replace, we get compression. Characteristic features of LZW includes,

- LZW compression uses a code table, with 4096 as a common choice for the number of table entries. Codes 0-255 in the code table are always assigned to represent single bytes from the input file.
- When encoding begins the code table contains only the first 256 entries, with the remainder of the table being blanks. Compression is achieved by using codes 256 through 4095 to represent sequences of bytes.
- As the encoding continues, LZW identifies repeated sequences in the data and adds them to the code table.
- Decoding is achieved by taking each code from the compressed file and translating it through the code table to find what character or characters it represents.

Compression using LZW

Example 1: Use the LZW algorithm to compress the string: **BABAABAAA**

The steps involved are systematically shown in the diagram below.

BABAABAAA ↑ P=A, C = empty

Encoder Output		String Table	
Output Code	representing	codeword	string
66	B	256	BA

LZW compression step 1

BABAABAAA ↑ P=B, C = empty

Encoder Output		String Table	
Output Code	representing	codeword	string
66	B	256	BA
65	A	257	AB

LZW compression step 2

BABAABAAA ↑ P=A, C = empty

Encoder Output		String Table	
Output Code	representing	codeword	string
66	B	256	BA
65	A	257	AB
256	BA	258	BAA

LZW compression step 3

BABAABAAA ↑ P=A, C = empty

Encoder Output		String Table	
Output Code	representing	codeword	string
66	B	256	BA
65	A	257	AB
256	BA	258	BAA
257	AB	259	ABA

LZW compression step 4

BABAABAAA ↑ P=A, C = A

Encoder Output		String Table	
Output Code	representing	codeword	string
66	B	256	BA
65	A	257	AB
256	BA	258	BAA
257	AB	259	ABA
65	A	260	AA

LZW compression step 5

BABAABAAA ↑ P=AA, C = empty

Encoder Output		String Table	
Output Code	representing	codeword	string
66	B	256	BA
65	A	257	AB
256	BA	258	BAA
257	AB	259	ABA
65	A	260	AA
260	AA		

LZW compression step 6

LZW Decompression

The LZW decompressor creates the same string table during decompression. It starts with the first 256 table entries initialized to single characters. The string table is updated for each character in the input stream, except the first one. Decoding is achieved by reading codes and translating them through the code table being built.

Q4:Define 2D DFT. With respect to 2D DFT of an Image State the following properties:
(A) Translation
(B) Rotation
(C) Periodicity
(D) Convolution theorem
(**10 marks- 2D DFT- 2 MARKS, PROPERTIES-1-1 MARK)**

Fourier transform of a 2D set of samples forming a bidimensional. sequence. • As in the 1D case, 2D-DFT, though a self-consistent transform, can be considered as a mean of calculating the transform of a 2D sampled signal defined over a discrete grid.

Property	Spatial domain	Frequency domain
Periodicity Where k_1 and k_2 are integers	$f(x, y) = f(x + k_1M, y) = f(x, y + k_2N) = f(x + k_1M, y + k_2N)$	$F(u, v) = F(u + k_1M, v) = F(u, v + k_2N) = F(u + k_1M, v + k_2N)$
Convolution	$f(x, y) * h(x, y)$	$F(u, v)H(u, v)$
Multiplication	$f(x, y)h(x, y)$	$F(u, v) * H(u, v)$
Correlation	Corr($f(x, y)$, $h(x, y)$)	$F^*(u, v)H(u, v)$
	$f^*(x, y)h(x, y)$	Corr($F(u, v)$, $H(u, v)$)
Linearity	$af(x, y) + bg(x, y)$	$aF(u, v) + bG(u, v)$ Where $DFT(f(x, y)) = F(u, v)$, $DFT(g(x, y)) = G(u, v)$

Property	Spatial domain	Frequency domain
Translation	$f(x, y)e^{j2\pi(\frac{u_0x}{M}+\frac{v_0y}{N})}$	$F(u - u_0, v - v_0)$
	$f(x - x_0, y - y_0)$	$F(u, v)e^{-j2\pi(\frac{ux_0}{M}+\frac{vy_0}{N})}$
	$f(x, y)(-1)^{x+y}$	$F(u - P/2, v - Q/2)$
Rotation	$f(r, \Theta + \Theta_0)$	$F(\omega, \Phi + \Theta_0)$
	$x = r\cos\Theta, y = r\sin\Theta$	$u = \omega\cos\Phi, v = \omega\sin\Phi$
Differentiation	$\left(\frac{\partial}{\partial t}\right)^m\left(\frac{\partial}{\partial z}\right)^n f(t, z)$	$(j2\pi\mu)^m(j2\pi v)^n F(\mu, v)$
Gaussian	$A2\pi\sigma^2 e^{-2\pi^2\sigma^2(t^2+z^2)}$	$Ae^{-(u^2+v^2)/2\sigma^2}$

Q5 A:Explain the concept of Homomorphic filtering in image enhancement. 5 MARKS

Homomorphic filtering is a **generalized technique for signal and image processing**, involving a nonlinear mapping to a different domain in which linear filter techniques are applied, followed by mapping back to the original domain.

Homomorphic Filtering

Homomorphic filtering:

The illumination-reflectance model can be used to develop a frequency domain procedure for improving the appearance of an image by simultaneous gray-level range compression and contrast enhancement. An image f(x, y) can be expressed as the product of illumination and reflectance components:

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

Equation above cannot be used directly to operate separately on the frequency components of illumination and reflectance because the Fourier transform of the product of two functions is not separable; in other words,

$$\Im\{f(x, y)\} \neq \Im\{i(x, y)\}\Im\{r(x, y)\}.$$

Suppose, however, that we define

$$\begin{aligned} z(x, y) &= \ln f(x, y) \\ &= \ln i(x, y) + \ln r(x, y). \end{aligned}$$

Then

$$\begin{aligned} \Im\{z(x, y)\} &= \Im\{\ln f(x, y)\} \\ &= \Im\{\ln i(x, y)\} + \Im\{\ln r(x, y)\} \end{aligned}$$

or

$$Z(u, v) = F_i(u, v) + F_r(u, v)$$

where Fi (u, v) and Fr (u, v) are the Fourier transforms of ln i(x, y) and ln r(x, y), respectively. If we process Z (u, v) by means of a filter function H (u, v) then, from

$$\begin{aligned} S(u, v) &= H(u, v)Z(u, v) \\ &= H(u, v)F_i(u, v) + H(u, v)F_r(u, v) \end{aligned}$$

where S (u, v) is the Fourier transform of the result. In the spatial domain,

$$s(x, y) = \Im^{-1}\{S(u, v)\}$$
$$= \Im^{-1}\{H(u, v)F_i(u, v)\} + \Im^{-1}\{H(u, v)F_r(u, v)\}.$$

By letting

$$i'(x, y) = \Im^{-1}\{H(u, v)F_i(u, v)\}$$

and

$$r'(x, y) = \Im^{-1}\{H(u, v)F_r(u, v)\},$$

Now we have

$$s(x, y) = i'(x, y) + r'(x, y).$$

Finally, as z (x, y) was formed by taking the logarithm of the original image f (x, y), the inverse (exponential) operation yields the desired enhanced image, denoted by g(x, y); that is,

$$g(x, y) = e^{s(x, y)}$$
$$= e^{i'(x, y)} \cdot e^{r'(x, y)}$$
$$= i_0(x, y)r_0(x, y)$$

where

$$i_0(x, y) = e^{i'(x, y)}$$

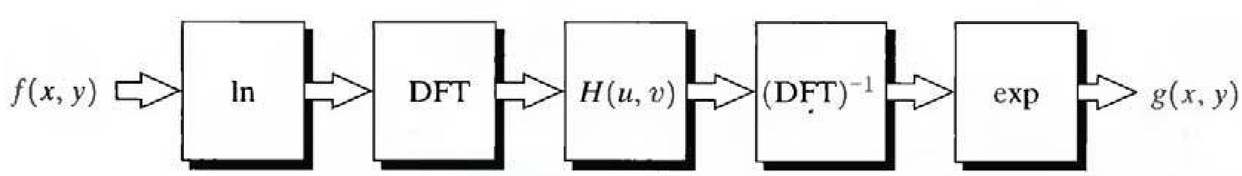


Fig.9.1 Homomorphic filtering approach for image enhancement

and

$$r_0(x, y) = e^{r'(x, y)}$$

are the illumination and reflectance components of the output image. The enhancement approach using the foregoing concepts is summarized in Fig. 9.1. This method is based on a special case of a class of systems known as homomorphic systems. In this particular application, the key to the approach is the separation of the illumination and reflectance components achieved. The homomorphic filter function H (u, v) can then operate on these components separately.

The illumination component of an image generally is characterized by slow spatial variations, while the reflectance component tends to vary abruptly, particularly at the junctions of dissimilar objects. These characteristics lead to associating the low frequencies of the Fourier transform of the logarithm of an image with illumination and the high frequencies with reflectance. Although these associations are rough approximations, they can be used to advantage in image enhancement.

A good deal of control can be gained over the illumination and reflectance components with a homomorphic filter. This control requires specification of a filter function H (u, v) that affects the low- and high-frequency components of the Fourier transform in different ways. Figure 9.2 shows a cross section of such a filter. If the parameters γL and γH are chosen so that $\gamma L < 1$ and $\gamma H > 1$, the filter function shown in Fig. 9.2 tends to decrease the contribution made by the low frequencies (illumination) and amplify the contribution made by high frequencies (reflectance). The net result is simultaneous dynamic range compression and contrast enhancement.

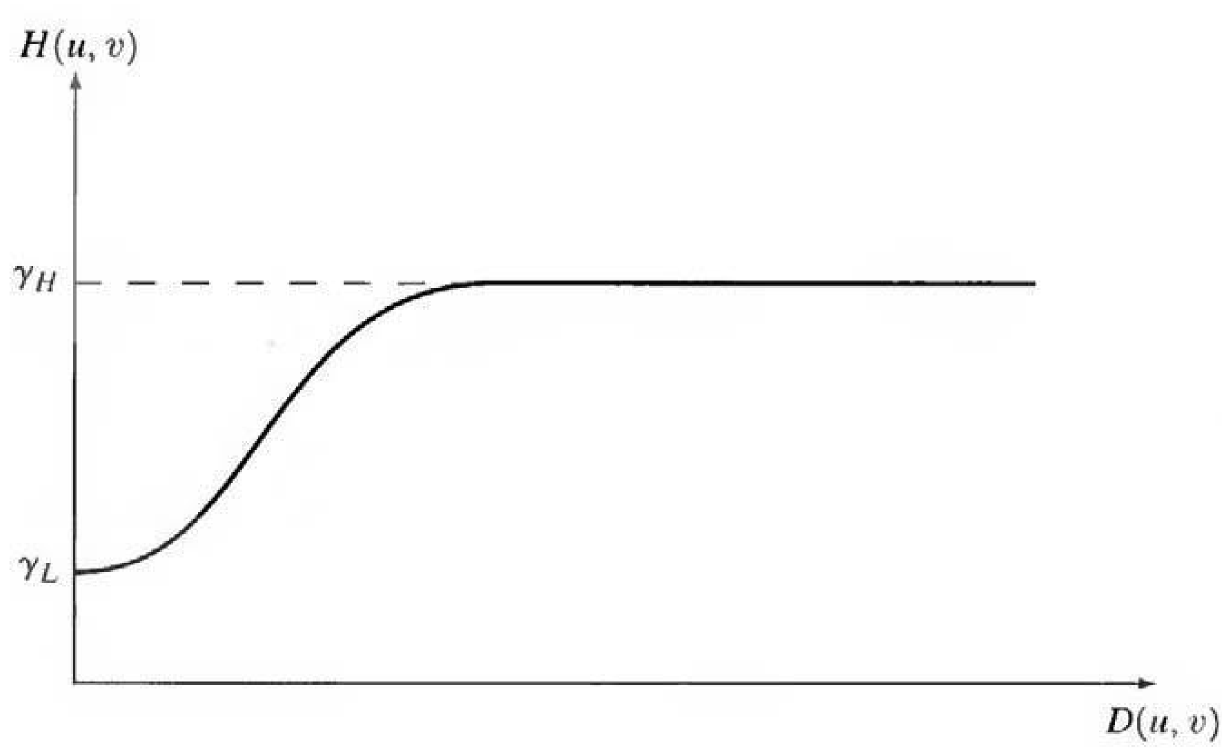


Fig.9.2 Cross section of a circularly symmetric filter function D (u. v) is the distance from the origin of the centered transform.

Q5:B:Explain Run Length Coding in image compression 5 MARKS

Run-length encoding (RLE) is a form of lossless data compression in which runs of data (sequences in which the same data value occurs in many consecutive data elements) are stored as a single data value and count, rather than as the original run. This is most useful on data that contains many such runs.

Run-length encoding (RLE) is a form of lossless data compression in which *runs* of data (sequences in which the same data value occurs in many consecutive data elements) are stored as a single data value and count, rather than as the original run. This is most efficient on data that contains many such runs, for example, simple graphic images such as icons, line drawings, Conway's Game of Life, and animations. For files that do not have many runs, RLE could increase the file size.

RLE may also be used to refer to an early graphics file format supported by CompuServe for compressing black and white images, but was widely supplanted by their later Graphics Interchange Format (GIF). RLE also refers to a little-used image format in Windows 3.x, with the extension `rle`, which is a run-length encoded bitmap, used to compress the Windows 3.x startup screen.

Consider a screen containing plain black text on a solid white background, over hypothetical scan line, it can be rendered as follows:

```
12W1B12W3B24W1B14W
```

This can be interpreted as a sequence of twelve Ws, one B, twelve Ws, three Bs, etc., and represents the original 67 characters in only 18. While the actual format used for the storage of images is generally binary rather than ASCII characters like this, the principle remains the same. Even binary data files can be compressed with this method; file format specifications often dictate repeated bytes in files as padding space. However, newer compression methods such as DEFLATE often use LZ77-based algorithms, a generalization of run-length encoding that can take advantage of runs of strings of characters (such as `BWWBWWBWWBWW`).

Run-length encoding can be expressed in multiple ways to accommodate data properties as well as additional compression algorithms. For instance, one popular method encodes run lengths for runs of two or more characters only, using an "escape" symbol to identify runs, or using the character itself as the escape, so that any time a character appears twice it denotes a run. On the previous example, this would give the following:

```
WW12BWW12BB3WW24BWW14
```

This would be interpreted as a run of twelve Ws, a B, a run of twelve Ws, a run of three Bs, etc. In data where runs are less frequent, this can significantly improve the compression rate.

One other matter is the application of additional compression algorithms. Even with the runs extracted, the frequencies of different characters may be large, allowing for further compression; however, if the run lengths are written in the file in the locations where the runs occurred, the presence of these numbers interrupts the normal flow and makes it harder to compress. To overcome this, some run-length encoders separate the data and escape symbols from the run lengths, so that the two can be handled independently. For the example data, this would result in two outputs, the string "`WWBWWBBWWBWW`" and the numbers (`12,12,3,24,14`).

Q6:Explain Image Gradient and various Gradient operators used for Edge Detection.-10 MARKS (2+2+2+2+2)

Vertical edges can be detected by using a horizontal gradient operator followed by a threshold operation to detect the extreme values of the gradient. The gradient produces a doublet of extremes, positive-negative or negative-positive, depending on the direction of the transition.
Edges are significant local changes of intensity in a digital image. An edge can be defined as a set of connected pixels that forms a boundary between two disjoint regions. There are three types of edges:

- **Horizontal edges**
- **Vertical edges**
- **Diagonal edges**

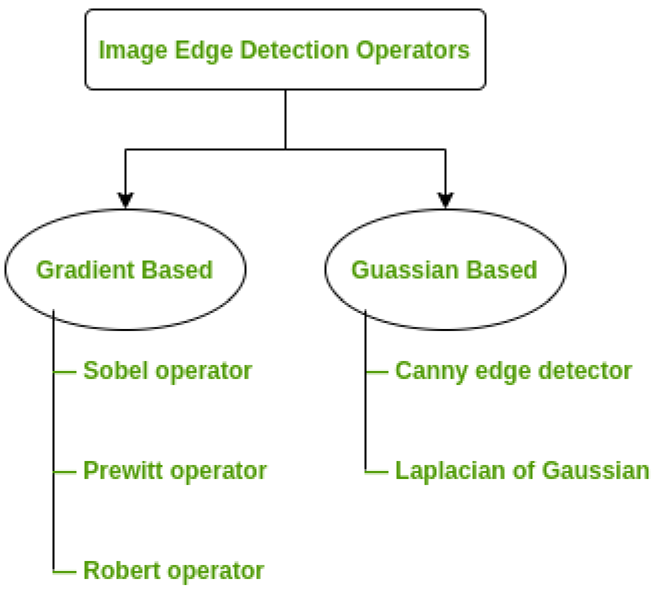
Edge Detection is a method of segmenting an image into regions of discontinuity. It is a widely used technique in digital image processing like

- **pattern recognition**
- **image morphology**
- **feature extraction**

Edge detection allows users to observe the features of an image for a significant change in the gray level. This texture indicating the end of one region in the image and the beginning of another. It reduces the amount of data in an image and preserves the structural properties of an image.

Edge Detection Operators are of two types:

- **Gradient – based operator which computes first-order derivations in a digital image like, Sobel operator, Prewitt operator, Robert operator**
- **Gaussian – based operator which computes second-order derivations in a digital image like, Canny edge detector, Laplacian of Gaussian**



Some Gradient Operator Sobel operator Prewitt operator Robinson mask
Sobel Operator: It is a discrete differentiation operator. It computes the gradient approximation of image intensity function for image edge detection. At the pixels of an image, the Sobel operator produces either the normal to a vector or the corresponding gradient vector. It uses two 3 x 3 kernels or masks which are convolved with the input image to calculate the vertical and horizontal derivative approximations respectively –

$$D_i = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} \quad \text{And} \quad D_j = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix}$$

Advantages:

1. **Simple and time efficient computation**
2. **Very easy at searching for smooth edges**

Limitations:

1. **Diagonal direction points are not preserved always**
2. **Highly sensitive to noise**
3. **Not very accurate in edge detection**
4. **Detect with thick and rough edges does not give appropriate results**

Prewitt Operator: This operator is almost similar to the sobel operator. It also detects vertical and horizontal edges of an image. It is one of the best ways to detect the orientation and magnitude of an image. It uses the kernels or masks –

$$D_i = \begin{bmatrix} -1 & 0 & +1 \\ -1 & 0 & +1 \\ -1 & 0 & +1 \end{bmatrix} \text{ And } D_j = \begin{bmatrix} +1 & +1 & +1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

Advantages:

1. **Good performance on detecting vertical and horizontal edges**
2. **Best operator to detect the orientation of an image**

Limitations:

1. **The magnitude of coefficient is fixed and cannot be changed**
2. **Diagonal direction points are not preserved always**

Robert Operator: This gradient-based operator computes the sum of squares of the differences between diagonally adjacent pixels in an image through discrete differentiation. Then the gradient approximation is made. It uses the following 2 x 2 kernels or masks –

$$D_x = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \text{ And } D_y = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

Advantages:

1. **Detection of edges and orientation are very easy**
2. **Diagonal direction points are preserved**

Limitations:

1. **Very sensitive to noise**
2. **Not very accurate in edge detection**