LAB MANUAL

SUBJECT:

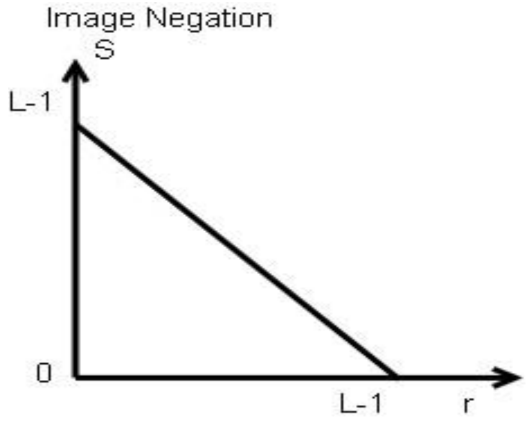
IMAGE PROCESSING

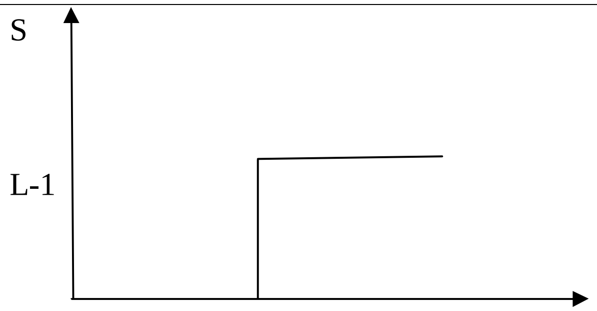
BSc CSIT

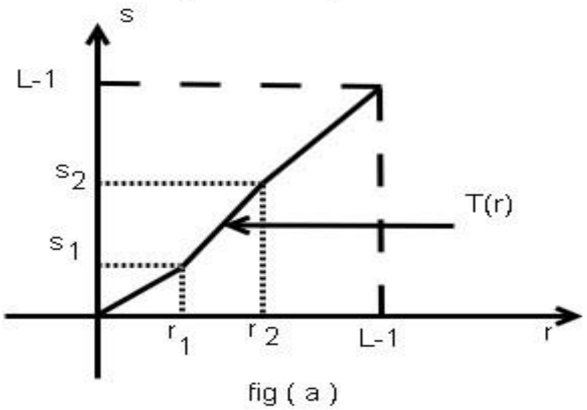
SEM V

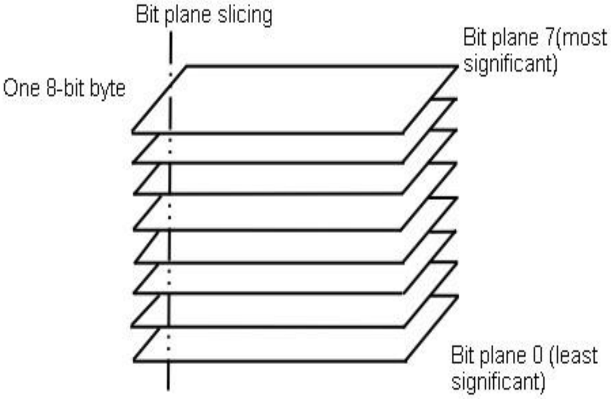
IMAGE PROCESSING

INDEX

Experiment No. 1A	***Negation of an image***
Aim	To study image negative
Tool	PYTHON
Theory	The negative of an image with gray levels in the range [0, L-1] is obtained by using the negative transformation given by the expression $S= L - 1 - r$ (1) This is according to the transformation $S = T (r)$ In above transformation (1) , the intensity of the output image decreases as the intensity of the input increases. The type of processing is particularly suited for enhancing white or gray detail embedded in dark regions of an image especially when black areas are dominants in site. 
Algorithm	1. Read i/p image 2. Read maximum gray level pixel of i/p image 3. Replace input image by (maximum – i/p) = o/p 4. Display o/p image
Questions	1. Explain application of image negation.

Experiment No. 1B	***Thresholding of an Image***
Aim	To study thresholding of the image
Tool	PYTHON
Theory	 Thresholding is a simple process to separate the interested object from the background. It gives the binary image. The formula for achieving thresholding is as follows s = 0 if r <= t s = L-1 if r > t t L-1 r
Algorithm	1. Read input image 2. Enter thresholding value t 3. If image pixel is less than t replace it by zero. 4. If image pixel is > t replace it by 255 5. Display input image 6. Display threshold image 7. Write input image 8. Write threshold image
Conclusion	Thresholding separate out the object from the background
Questions	1. Explain local & global thresholding 2. Discuss some application of thresholding.

Experiment No. 1C	***Contrast Stretching of an Image***
Aim	To study Contrast Stretching of an image
Tool	PYTHON
Theory	Low contrast images can result from poor illumination, lack of dynamic range in the imaging sensor etc. The idea behind contrast stretching is to increase the dynamic range of the gray levels in the image being processed. The transformation function for contrast stretching is given by r 0 r r_1 = $(r-r_1)+s_1$ r_1 r r_2 $(r-r_2)+ s_2$ r_2 r L-1 Stretching of an image  fig (a) tion of the points (r_1 , s_1) & (r_2 , s_2) control the shape of the transformation
Algorithm	1. Read input image 2. Enter values r1,r2,s1,s2 3. Calculate alpha,beta and gamma slopes. 3. if input pixel value is <= r1 then o/p = alpha x input 5. If input pixel is > r1and <=r2 then o/p = beta x (r-r1)+s1 6. otherwise o/p = gamma x (r-r2)+s2 7. Display i/p image 8. Display o/p image.
Conclusion	Contrast stretching increases the contrast of the image.
Questions	1. Explain difference between contrast stretching & histogram equalization.

Experiment No. 2	***Bit Plane Slicing***
Aim	To study Bit Plane Slicing
Tool	PYTHON
Theory	This transformation involves determining the number of usually significant bits in an image. In case of a 8 bit image each pixel is represented by 8 bits. Imagine that the image is composed of eight 1 bit planes ranging from bit plane 0 for the least significant bit to bit plane 7 for the most significant bit. Plane 0 contains all the lowest order bits in the bytes comprising the pixels in the image & plane 7 contains all the high order bits. The higher order bits contain usually significant data and the other bit planes contribute to more subtle details in the iamge. Separating a digital image into its bit planes is useful for analyzing the relative importance played by each bit of the image.  Bit plane slicing Bit plane 7(most significant) One 8-bit byte Bit plane 0 (least significant)
Algorithm	1.Read i/p image 2. Use bitand operation to extract each bit 3. Do the step 2 for every pixel. 4. Display the original image and the biplanes formed by bits extracted
Conclusion	Higher order bit planes carries maximum visual information

Questions	1. Explain the importance of bit plane slicing in image enhancement & image compression.

Experiment No. 3	***Histogram Equalization***
Aim	To implement histogram equalization.
Tool	PYTHON
Theory	Histogram of a digital image with gray levels in range [0,L-1] is a discrete function h($_k$) = n_k where $_k$ k^{th} gray level and n_k = no. of pixels of an image having gray level r_k In histogram there are 3 possibilities as follows, 1. For a dark image the components of histogram on the low (dark) side. 2. For a bright image the component are on high (bright) side & 3. For an image with low contrast they are in the middle of gray side. Histogram equalization is done to spread there component uniformly over the gray scale as far as possible. This is obtained by function Sk = ∑ (limit k to i=0) h_i /n; k = 0,1,2,3,...i-l Thus processed image is obtained by mapping each pixel with level r_k into a corresponding pixel with level s_k in o/p image. This transformation is called Histogram equalization
Algorithm	1. Read the i/p image & its size. 2. Obtain the gray level values of each pixel & divide them by total number of gray level values. 3. Implement the function Sk 4. Plot the equalized histogram and original histogram. 5. Display the original and the new image.
Conclusion	Digital histogram enhances image but it does not generate a flat histogram
Questions	1. What information one can get by observing histogram.

Experiment No. 4	***Histogram Specification***
Aim	To implement histogram specification
Tool	PYTHON
Theory	Histogram equalization automatically determines a transformation function that seeks to produce an output image that has a uniform histogram. But it is useful sometimes to be able specify the shape of the histogram that we wish the processed image to have. The method used to generate a processed image that has a specified histogram is called histogram specification. $Sk = T(r_k)=\sum Pr(r_j)$ $k = 0,1,2,3,...L\text{-}l$ $Vk= G(z_k) = \sum Pz(z_j)$ $k = 0,1,2,3,...L\text{-}l$ $Zk = G^{-1}(T(r_k))$ $k = 0,1,2,3,...L\text{-}l$ Map each pixel with level r_k into a corresponding pixel with level s_k . Obtain the transformation function G from a given histogram Pz(z). For any Zq this transformation function yields a corresponding value Vq. We would find the corresponding value Zq from G^{-1}.
Algorithm	. Obtain the histogram of the given image. 2. Map each level r_k to s_k 3. Obtain the transformation function G from the given Pz(z) 4. Calculate z_k for each value of s_k 5. For each pixel in the original image, if the value of that pixel is r_k , map this value to its corresponding level s_k ,then map level s_k into the final value z_k 6. Display the modified image and its histogram
Questions	Explain the histogram specification in continuous domain.

Experiment No. 5	***Zooming by interpolation and replication***
Aim	To implement the magnification by replication and interpolation

Tool	PYTHON
Theory	Zooming can be done in two ways. 1)Replication : In replication we simply replicate each pixel and then replicate each row. Hence image of size n x n is zoomed to 2n x2n. Zooming by replication gives the final image a patchy look since clusters of grey levels are formed. This can be substantially reduced by using a better method of zooming known as interpolation. 2) Interpolation : In this method instead of replicating each pixel, average of two adjacent pixels along the rows is taken and placed between two pixels. The same operation is then performed along the columns. The patchiness that was present in the replicated image is much less in the interpolated image.
Algorithm	Replication: 1. Read i/p image. 2. Replicate each pixel 3. Replicate each row 4. Display o/p image Interpolation 1. Read i/p image 2. Average of two adjacent pixels along the rows is taken and placed between two pixels. 3. Do the same along columns 4. Display o/p image.
Conclusion	Zooming by interpolation is more effective than zooming by replication
Questions	1. Explain the methods of zooming by using convolution mask

Experiment No. 6A	***Filtering in spatial domain: Low pass filtering***
Aim	To implement low pass filtering in spatial domain

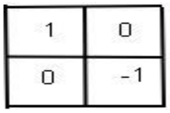
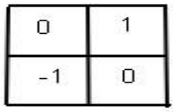
<table>
<tr><td>Tool</td><td>PYTHON</td></tr>
<tr><td>Theory</td><td>Low pass filtering as the name suggests removes the high frequency content from the image. It is used to remove noise present in the image. Mask for the low pass filter is :
<table>
<tr><td>1/9</td><td>1/9</td><td>1/9</td></tr>
<tr><td>1/9</td><td>1/9</td><td>1/9</td></tr>
<tr><td>1/9</td><td>1/9</td><td>1/9</td></tr>
</table>
One important thing to note from the spatial response is that all the coefficients are positive. We could also use 5 x 5 or 7 x 7 mask as per our requirement. We place a 3 x 3 mask on the image . We start from the left hand top corner. We cannot work with the borders and hence are normally left as they are. We then multiply each component of the image with the corresponding value of the mask. Add these values to get the response. Replace the center pixel of the o/p image with these response. We now shift the mask towards the right till we reach the end of the line and then move it downwards.</td></tr>
<tr><td>Algorithm</td><td>1. Read I/p image
2. Ignore the border pixel
3.Apply low pass mask to each and every pixel.
4. Display the o/p image</td></tr>
<tr><td>Conclusion</td><td>Low pass filtering makes the image blurred.</td></tr>
<tr><td>Questions</td><td>1. Explain weighted average filter.</td></tr>
</table>

Experiment No. 6B	***Filtering in spatial domain: High pass filtering***
Aim	To implement high pass filtering in spatial domain
Tool	PYTHON

<table>
<tr><td>Theory</td><td>High pass filtering as the name suggests removes the low frequency content from the image. It is used to highlight fine detail in an image or to enhance detail that has been blurred. Mask for the high pass filter is :
<table>
<tr><td>-1/9</td><td>-1/9</td><td>-1/9</td></tr>
<tr><td>-1/9</td><td>8/9</td><td>-1/9</td></tr>
<tr><td>-1/9</td><td>-1/9</td><td>-1/9</td></tr>
</table>

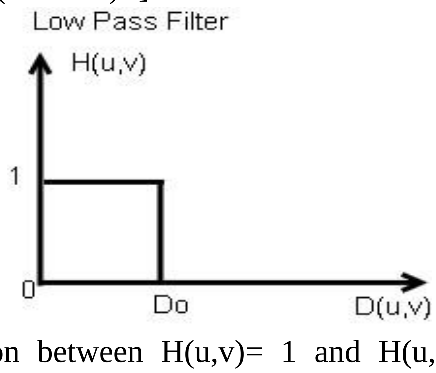
One important thing to note from the spatial response is that sum of all the coefficients is zero. We could also use 5 x 5 or 7 x 7 mask as per our requirement. We place a 3 x 3 mask on the image . We start from the left hand top corner. We cannot work with the borders and hence are normally left as they are. We then multiply each component of the image with the corresponding value of the mask. Add these values to get the response. Replace the center pixel of the o/p image with these response. We now shift the mask towards the right till we reach the end of the line and then move it downwards.</td></tr>
<tr><td>Algorithm</td><td>1. Read I/p image
2. Ignore the border pixel
3.Apply high pass mask to each and every pixel.
4. Display the o/p image</td></tr>
<tr><td>Conclusion</td><td>High pass filtering makes the image sharpened.</td></tr>
<tr><td>Questions</td><td>1. Show that high pass= Original – low pass</td></tr>
</table>

Experiment No. 6C	***Filtering in spatial domain: Median filtering***
Aim	To implement median filtering in spatial domain
Tool	PYTHON
Theory	Median filtering is a signal processing technique developed by tukey that is useful for noise suppression in images. Here the input pixel is replaced by the median of the pixels contained in the window around the pixel. The median filter disregards extreme values and does not allow them to influence the selection of a pixel value which is truly representative of the neighborhood.
Algorithm	1. Read i/p image 2. Add salt and pepper noise in the image 3. use 3 x 3 window. 4. Arrange the pixels in the window in ascending order. 5. Select the median. 6 Replace the center pixel with the median. 7. Do this process for all pixels. 8. Display the o/p image.
Conclusion	Median filtering works well for impulse noise but performs poor for Gaussian noise
Questions	1. Explain”Median filter removes the impulse noise” with example.

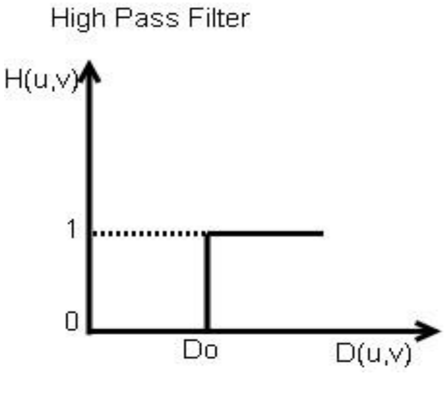
Experiment No. 7	***Edge Detection***
Aim	To implement Image segmentation using edge detection technique.
Tool	PYTHON
Theory	Image segmentation can be achieved in two ways, 1. Segmentation based on discontinuities of intensity. 2. Segmentation based on similarities in intensity edge detection form an important part. An edge can be defined as a set of disconnected pixels that form a boundary between 2 disjoint regions. Edge detection is achieved through various masks. 1. Roberts Masks : Roberts Masks F = Z5 – Z9 + Z6 – Z8 $\begin{bmatrix} Z1Z2Z3 \\ \\ Z7Z8Z9 \end{bmatrix}$ Therefore masks are  $\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$  $\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$ *Z4 Z5 Z6* There are masks along x&y gradient. The sum of two Roberts Masks $\begin{bmatrix} 1 & 1 \\ -1 & -1 \end{bmatrix}$

Algorithm	1. Read i/p image & its size 2. apply prewitt, sobel & laplacian edge masks on i/p image 3. Display i/p image & edge detected image. Use prewitt mask m1 = 0 $\begin{bmatrix} -1 & -1 & -1 \\ & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$ m2 $\begin{bmatrix} - & 1\ 0\ 1 \\ - & = 1\ 0 \\ - & 1\ 0\ 1 \end{bmatrix}$ 1 sobel mask m1 = $\begin{bmatrix} +12 & & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$ m2 1 0 $\begin{bmatrix} + & 1\ 0 & -1 \\ & = 2 & -2 \\ & & -1 \end{bmatrix}$ 0 laplacian m = $\begin{bmatrix} 0 & -10 & \\ - & & -1 \\ 0 & -10 & \end{bmatrix}$ 1 4
Conclusion	Prewitt is simpler to implement but sobel gives the better result. Laplacian is more sensitive to noise.
Questions	1. Give the difference between first order derivative filter and second order derivative filter 2. What is compass gradient mask

Experiment No. 8	***Data compression using Huffman coding***
Aim	To implement data compression using Huffman coding
Tool	PYTHON
Theory	It is used to reduce the space that an image uses on disk or in transit. It is the most popular technique to remove the coding redundancy. When coding the symbols of an information source individually Huffman coding yields the smallest possible number of code symbols per source symbol. It is lossless coding technique.
Algorithm	1. Order the gray levels according to their frequency of use, most frequent first 2. Combine the two least used gray levels into one group, combine their frequencies and reorder the gray levels 3. Continue to do this until only two gray levels are left 4. Now allocate a '0' to one of these gray level groups and '1' to the other 5. Work back through the groupings so that where two groups have been combined to form a new , larger, group which is currently coded as 'ccc' , code one of the smaller groups as 'ccc0' and the other as 'cccc1'.
Conclusion	Huffman code is an instantaneous uniquely decodable block code.
Questions	1. What is uniquely decodable code? 2. Give the formulas to calculate entropy, average length, compression ratio, coding efficiency.

Experiment No. 9A	***Filtering in frequency domain: low pass filtering***
Aim	To study Low Pass Filtering
Tool	PYTHON
Theory	Low pass filters attenuate or eliminate high frequency components while leaving low frequencies untouched. High frequency components characterize edges and other sharp details in an image so that the net effect of low pass filtering is image blurring. The transfer function for an ideal low pass filter is given by $H (u,v) = 1$ if $D(u,v) \quad D_0$ & 0 if $D(u,v)>D_0$ Where D_0 is a specified non-negative quality and D(u,v) is the distance from point (u,v) to the origin of the frequency plane. In case of a NxN image , $D(u,v) = [(u – N/2)^2 + (V – N/2)^2]^{1/2}$  The point of transition between H(u,v)= 1 and H(u,v)=0 is called cut off frequency. In this case it is D_o .

Algorithm	1. Read the i/p image & its size. 2. Read the cutoff frequency fc 3. Implement the function d = $[(u – N/2)^2 + (V – N/2)^2]$ 4. Find impulse response such that if d<fc IR=1 else IR=0 for LPF 5. Find EFT 2- DFT of i/p image . 6. Shift 2D FFT image 7. Multiply IR with shifted 2DFFT o/p element by element. 8. Take absolute multiple value of image 9. Display Low pan image.
Conclusion	As cutoff frequency goes on decreasing we get more and more blurring effect.
Questions	1. Why ideal low pass filter gives rise to ringing effect?

Experiment No. 9B	***Filtering in frequency domain: High pass filtering***
Aim	To study High Pass Filtering
Tool	PYTHON
Theory	This class of filters can be designed by their effect of emphasizing or strengthening the edges within an image. A high pass filter has the inverse characteristic of a low pass filter, it will not change the high frequency component of the signal but will attenuate the low frequencies and eliminate any constant background intensity. The transfer function for an ideal high pass filter is given by $H (u,v) = 0$ if $D(u,v) \quad D_0$ & 1 if $D(u,v)>D_0$ Where Do is the cutoff distance measured from the origin of the frequency plane. D(u,v) is the distance from the point (u,v) to the origin of frequency plane for NxN image $D(u,v) = [(u – N/2)^2 + (V – N/2)^2]^{1/2}$ 

Algorithm	1. Read the i/p image and size 2. Enter the cutoff frequency d_0 3. Implement function d = $[(u – N/2)^2 + (V – N/2)^2]$ 4. Make impulse response H=0 if $d<d_0$ else H=1 5. Take two dimensional f_T of i/p image 6. Shift ff_T image 7. Multiply shifted ff_T values pixel by pixel with IR(H) & obtain x image. 8. Take absolute value of x 9. Display HPF & original image.
Conclusion	High pass filter produces sharpening of the image
Questions	1.Explain butterworth high pass filter.

Experiment No. 10	***Hadamard Transform***
Aim	To implement Hadamard transform
Tool	PYTHON

Theory	The Hadamard transform is based on the Hadamard matrix which is a square array having entries of +1 or -1 only. The Hadamard matrix of order 2 is given by $H(2)= \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$ The rows and columns are orthogonal. For orthogonality of vectors the dot product has to be zero. We get H(4) from the Kronecker product of H(2) H(4) = H(2) X H(2) So we know that the Hadamard matrices of order 2^n can be recursively generated $H(2^n) = H(2) X H(2^{n-1})$ The rows of Hadamard matrix can be considered to be samples of rectangular waves with sub-periods of 1/N units. If x(n) is N-point 1 dimensional sequence of finite valued real numbers arranged in a column then the Hadamard transformed sequence is given by X = T.x X[n] = [H(N) x(n)] The inverse Hadamard transform is given by x(n) = 1/N H(N) X(n) For a two dimensional sequence f of size N X N, we compute the Hadamard transform using equation F = T f T F = [H(N) f H(N)]
Algorithm	1. Read i/p image 2. Divide the image into 8 x 8 blocks. 3. Apply Hadamard transform to the blocks 4. Merge the blocks and display the transformed o/p image. 5. Apply inverse transform and display the image.
Conclusion	Hadamard transform is the simple to implement. It is non sinusoidal, orthogonal function. Transforms are used in image compression.
Questions	Explain Haar, Walsh transform.