Session 3

1. Intensity Transformations and Spatial filter

OBJECTIVES:

The objective of this lab is to understand & implement

1. Image enhancement in spatial domain through Gray level Transformation function
2. Linear Transformation
    - Image Negation function
    - Identity function
3. Logarithmic Transformation
4. Power Law Transformation
5. Piece Wise Linear Transformation

BACKGROUND MATERIAL:

Image Enhancement in Spatial Domain -Basic Grey Level Transformations Image enhancement is a very basic image processing task that defines us to have a better subjective judgement over the images. And Image Enhancement in spatial domain (that is, performing operations directly on pixel values) is the very simplistic approach. Enhanced images provide better contrast of the details that images contain. Image enhancement is applied in every field where images are ought to be understood and analysed. For example, Medical Image Analysis, Analysis of images from satellites, etc.

Image enhancement simply means, transforming an image f into image g using T. Where T is the transformation. The values of pixels in images f and g are denoted by r and s, respectively. As said, the pixel values r and s are related by the expression,

$$s = T(r)$$

where T is a transformation that maps a pixel value r into a pixel value s. The results of this transformation are mapped into the grey scale range as we are dealing here only with grey scale digital images. So, the results are mapped back into the range $[0, L-1]$, where $L = 2k$, k being the number of bits in the image being considered. So, for instance, for an 8-bit image the range of pixel values will be $[0, 255]$.
There are three basic types of functions (transformations) that are used frequently in image enhancement. They are,

1. Linear
2. Logarithmic
3. Power-Law

The transformation map plot shown in Figure 3.1 depicts various curves that fall into the above three types of enhancement techniques.
The Identity and Negative curves fall under the category of linear functions. Identity curve simply indicates that input image is equal to the output image. The Log and Inverse-Log curves fall under the category of Logarithmic functions and n^{th} root and n^{th} power transformations fall under the category of Power-Law functions.

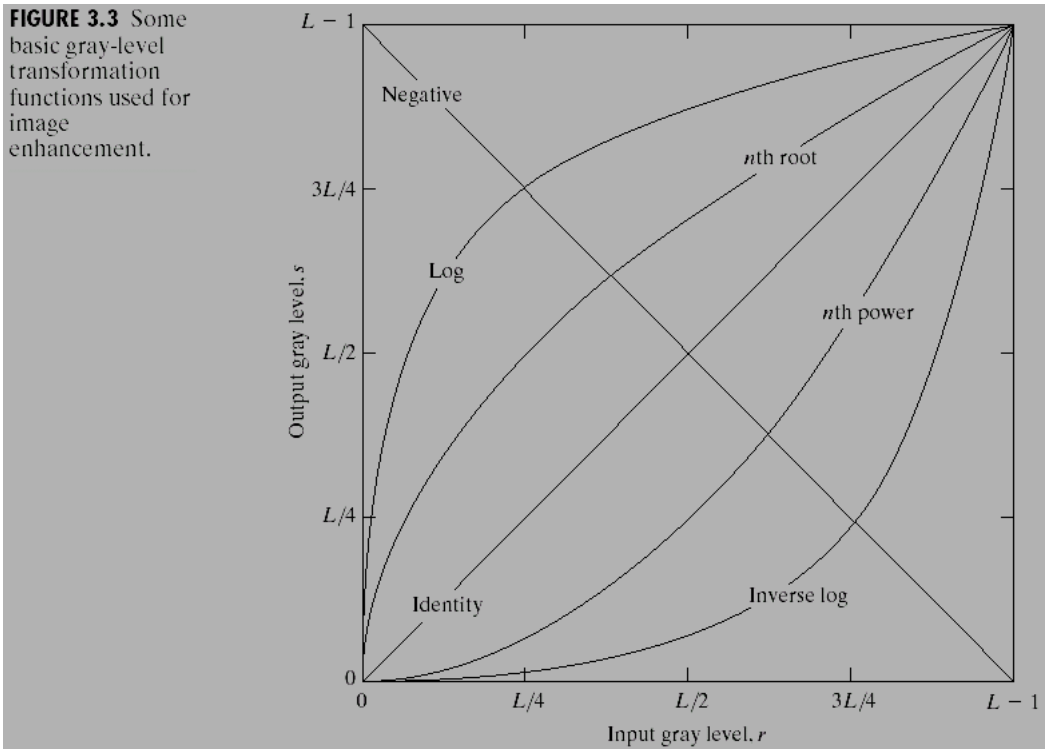

Figure 3.1: Plot of various transformation functions

IMAGE NEGATION FUNCTION:

The negative of an image with grey levels in the range $[0, L-1]$ is obtained by the negative transformation shown in Figure 3.2, which is given by the expression,

$$s = L - 1 - r$$

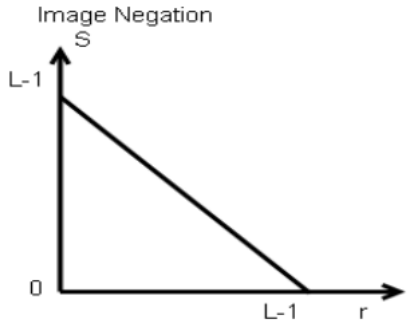

Figure 3.2: Image Negation Transformation

This expression results in reversing of the grey level intensities of the image thereby producing a negative like image. The output of this function can be directly mapped into the grey scale lookup table consisting values from 0 to $L-1$.

ALGORITHM:

1. Read input image
2. Read maximum gray level pixel of input image
3. Replace input image by $(\mathrm{m}aximum - \mathrm{i}nput) = \mathrm{o}utput$
4. Display output image

Question 1. Explain application of image negation.

THRESHOLDING OF AN IMAGE:

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

$$s = 0; if r \leq t$$

$$s = L - 1; if r > t$$

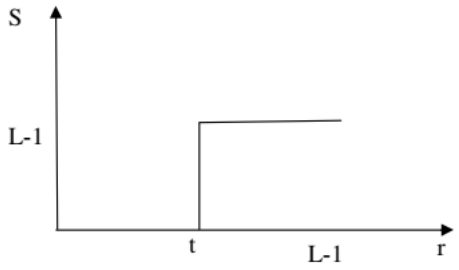

Figure 3.3: Thresholding

ALGORITHM:

1. Read input image
2. Enter thresholding value t
3. If image pixel is less than t replace it by 0.
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

Question 2. Explain local & global thresholding

Question 3. Discuss some application of thresholding.

LOG TRANSFORMATIONS:

The log transformation curve shown in Figure 3.1 is given by the expression,

$$s = c \log(1 + r)$$

where c is a constant and it is assumed that $r \geq 0$. The shape of the log curve in Figure 3.1 tells that this transformation maps a narrow range of low-level grey scale intensities into a wider range of output values. And similarly maps the wide

range of high-level grey scale intensities into a narrow range of high level output values. The opposite of this applies for inverse-log transform. This transform is used to expand values of dark pixels and compress values of bright pixels.

POWER-LAW TRANSFORMATIONS:

The n^{th} power and n^{th} root curves shown in Figure 3.1 can be given by the expression,

$$s = cr^{\gamma}$$

This transformation function is also called as gamma correction. For various values of γ different levels of enhancements can be obtained. This technique is quite commonly called as Gamma Correction. If you notice, different display monitors display images at different intensities and clarity. That means, every monitor has built-in gamma correction in it with certain gamma ranges and so a good monitor automatically corrects all the images displayed on it for the best contrast to give user the best experience.

The difference between the log-transformation function and the power-law functions is that using the power-law function a family of possible transformation curves can be obtained just by varying the λ.

POWER-LAW TRANSFORMATION PROGRAM:

```
image=imread('pout.tif');
    figure;
    imshow(image);
    image_double=im2double(image);
  [r c]=size(image_double);
  cc=input('Enter the value for c==>');
  ep=input('Enter the value for gamma==>');
  for i=1:r
    for j=1:c
      imout(i,j)=cc*power(image_double(i,j),ep);
    end
  end
  figure,imshow(imout);
```

Program 3.1: Power-law transformation

Enter value $c = 1$ and $\gamma = .2\%$, it will make input image brighter as shown in Figure 3.4

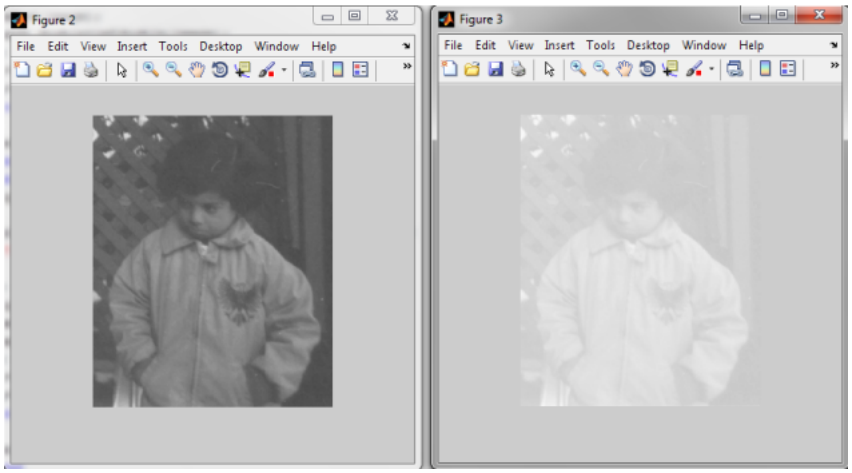

Figure 3.4: Power-Law Transformation with $\gamma < 1$

Enter value $c = 1$ and $\gamma = 5\%$, it will make input image darker as shown in Figure 3.5

Figure 3.5: Power-Law Transformation with $\gamma > 1$

BIT PLANE SLICING:

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. 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 image. Separating a digital image into its bit planes is useful for analyzing the relative importance played by each bit of the image.

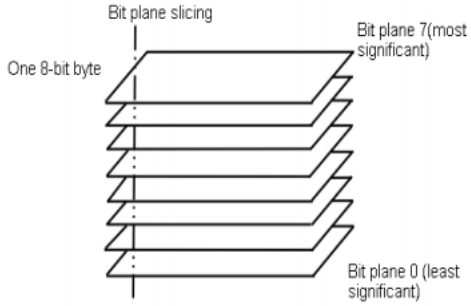

Figure 3.6: Bit plane splicing

ALGORITHM:

1. Read i/p image
2. Use bit and 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

Question 4. Explain the importance of bit plane slicing in image enhancement & image compression.

Task 1. Implement negation transform.

Task 2. Implement Logarithmic transform.

Task 3. Implement Piece wise linear transform.

2. Filtering in spatial domain

LOW PASS FILTERING:

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 : One important thing to note from the spatial response is that all the coefficients are positive. We could also use 5×5 or 7×7 mask as per our requirement. We place a 3×3 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. Low-pass filter kernel is:

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

ALGORITHM:

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

CONCLUSION:

Low pass filtering makes the image blurred.

Question 5. Explain weighted average filter.

HIGH PASS FILTERING:

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:

$$\frac{1}{9}\begin{bmatrix}-1 & -1 & -1\\ -1 & 8 & -1\\ -1 & -1 & -1\end{bmatrix}$$ One important thing to note from the spatial response is that sum of all the coefficients is zero. We could also use 5×5 or 7×7 mask as per our requirement. We place a 3×3 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.

ALGORITHM:

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

CONCLUSION:

High pass filtering makes the image sharpened.

Question 6. Show that high pass $=$ Original $-$ low pass .

MEDIAN FILTERING:

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 3×3 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.

Question 7. Explain "Median filter removes the impulse noise" with example

Task 4. Write a program to implement Smoothing Spatial filter and note the effects on given images.

Task 5.
Write a program to implement order statistics filters and write down your observations.

3. Noise filtering

OBJECTIVES:

- Learn about different kinds of noises
- Learn about different kind of filters
- Learn about different kinds of noise reduction techniques using filters

DIFFERENT KIND OF NOISE:

Salt and pepper noise Random dark (pepper) and bright (salt) pixels in an image

Gaussian Noise Noises that are distributed with Gaussian probability distribution

DIFFERENT KINDS OF FILTERS:

Max Filter Chooses the maximum pixel value from the window as the center value. Good for reducing pepper noises (why?)

Min Filter Chooses the minimum pixel value from the window as the center value. Good for reducing salt noises (why?)

Median Filter Chooses the median pixel value from the window as the center value. Good for reducing both salt and pepper noises (why?)

Average Filter Chooses the average pixel value from the window as the center value. Good for reducing noises and sharpness caused by random pixels- but reduces the pixel intensities. It is a linear filter. The kernel is:

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

Gaussian Filter Chooses the Gaussian weighted average pixel value from the window as the center value. Good for reducing noises and sharpness caused by random pixels- gives better pefromance than average filter. It is a linear filter. The kernel is:

$$\begin{bmatrix}0.0000 & 0.0004 & 0.0017 & 0.0004 & 0.0000\\ 0.0004 & 0.0275 & 0.1102 & 0.0275 & 0.0004\\ 0.0017 & 0.1102 & 0.4421 & 0.1102 & 0.0017\\ 0.0004 & 0.0275 & 0.1102 & 0.0275 & 0.0004\\ 0.0000 & 0.0004 & 0.0017 & 0.0004 & 0.0000\end{bmatrix}$$

Task 6. Learn the use of `imnoise()` function of Matlab and introduce noises in an image.

Task 7. Run min, max, median filters on noisy images and notice the output

Task 8. Run average and Gaussian filters on noisy images and notice the differences.

4. Template matching

OBJECTIVES:

1. Learn about cosine distance
2. Learn about correlation
3. Use correlation and template matching for finding patterns in a image.

NORMALIZED CROSS CORRELATION:

Finds the similarity between two collection of numbers/ vectors/ images. It helps to find a template within a given image by running a sliding window same as the template and calculating the normalized cross correlation. The normalized cross correlation coefficient between two image windows u, v is defined as:

$$\gamma(u,v) = \frac{\sum_{x,y}[f(x,y) - \bar{f}_{u,v}][t(x-u,y-v) - \bar{t}]}{\sqrt{\{\sum_{x,y}[f(x,y) - \bar{f}_{u,v}]^2 \sum_{x,y}[t(x-u,y-v) - \bar{t}]^2\}}}$$

Task 9. Write a function in Matlab which will implement normalized cross correlation between two arrays.

Task 10. Use the idea of normalized cross-correlation two find similar pair of images from a collection of images

5. Convolution

OBJECTIVES:

- Learn about convolution operation
- Use convolution for filtering and edge detection operations

INTUITION:

Convolution denotes the sliding of a function over another function and computing their product sum over the whole interval. In terms of image, it usually involves running a filter/kernel for smoothing/noise reducing/ edge detection/ template matching- as all of them are done using some kernel who slide over the images.

Task 11. Learn the use of Matlab's `im2conv()` function

Task 12. Write a code from scratch which will take an image and a kernel and run convolution

Task 13. Try to perform template matching using your convolution code.