Experiment No: 4

Title: Apply different thresholds &filter using MATLAB and DIPLAB- 1.0

Software Used:

Programming Language – MATLAB® (Version 7 or higher), VisualDSP++

Operating System – Microsoft Windows (XP 32-bit or higher)(Preferable)

Theory:

Thresholding

If in original contrast stretching function r1=r2=th, s1=00 and s2=L-1, the transformation function becomes a thresholding function, that creates a binary image. This function is shown as below.

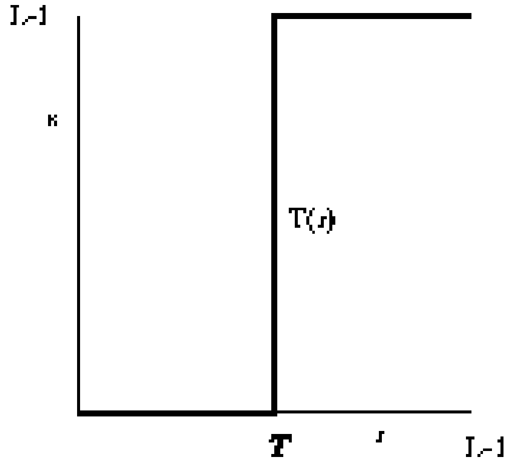


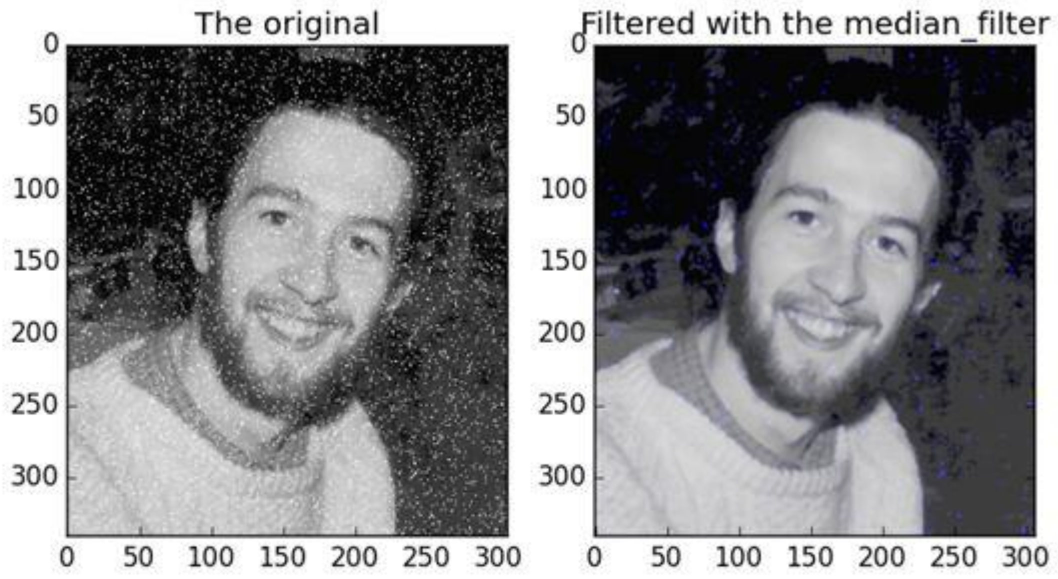
Fig. Thresholding mapping transform function

In computer vision and image processing. Otsu's method is used to automatically perform clustering-based image thresholding, or, the reduction of a grayscale image to a binary image. The algorithm assumes that the image contains two classes of pixels following bi-modal histogram (foreground pixels and background pixels), it then calculates the optimum threshold separating the two classes so that their combined spread (intra-class variance) is minimal. The extension of the original method to multi-level thresholding is referred to as the Multi Otsu method.

Median Filter

Median filtering is a nonlinear method used to remove noise from images. It is widely used as it is very effective at removing noise while preserving edges. It is particularly effective at removing 'salt and pepper' type noise.

The median filter works by moving through the image pixel by pixel, replacing each value with the median value of neighbouring pixels. The pattern of neighbours is called the "window", which slides, pixel by pixel, over the entire image. The median is calculated by first sorting all the pixel values from the window into numerical order, and then replacing the pixel being considered with the middle (median) pixel value.



Procedure:

Using MATLAB

1. Open an image using 'imread' and 'imshow' functions. Convert it to grayscale if required.

2. For Global Single thresholding (Manual), choose a threshold value‘t’. Obtain the image segmentation result by using "im2bw function using 't’. Display the result.
3. For Global Dual thresholding (Manual), choose two threshold values "t1" and "t2".Obtain the image segmentation result by designing an algorithm using conditional statements. Display the result.
4. For Global Single thresholding (Automated), use function 'graythresh' to obtain the threshold value‘t’ automatically by Otsu's method. Obtain the image segmentation result by using ‘im2bw’ function using‘t’. Display the result.

Using DIPLAB

1. Load the image in processor memory
2. Note down the height and width of image which is loaded.
3. Replace the value of variable iImgWdt and iImgHgt with image width and height
4. Compile the program
5. Run the program with multi-processor Run Mode
6. Read the output image from the processor memory defined in the program

<u>Conclusion</u>:

Experiment No: 2

Title: Implement RGB to Gray scale Image Processing using DIPLAB-1.0 and MATLAB

Software Used:

Programming Language – MATLAB® (Version 7 or higher), VisualDSP++

Operating System – Microsoft Windows (XP 32-bit or higher)(Preferable)

Theory:

RGB Image

RGB stands for red, green and blue, RGB image format is the most widely used color format.

In RGB model each color image is formed by three different images. Red image, Green image and Blue image. A normal grayscale image can be defined by only one 2D matrix, but a color image is actually composed of three different 2D matrices. Individually all R, G. B Matrices are Gray Image, but on combining it forms color image.

One color image matrix = Red matrix + Green matrix + Blue matrix

RGB is 24bit color format in which is equally divided on 8, so it has been distributed equally between three different color channels. Red 8 bits, Green 8 bits, Blue 8 bits.

Grayscale Image

Gray image has 8-bit color format, which is one of the most famous image format. It has 256 different shades of (Black and White) colors in it. The range of the colors in 8-bit vary from 0-255. Where @ stands for black, and 255 stands for white, and 127 stands for gray color. This format was used initially by early models of the operating systems UNIX and the early color Macintoshes.

Fig. Color shade in Grayscale

RGB to Grayscale Conversion

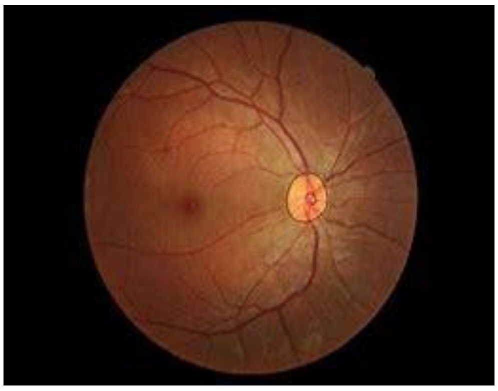
RGB colour model can be converted to gray scale format. There are two methods to convert it. Both has their own merits and demerits.

1] Average method:

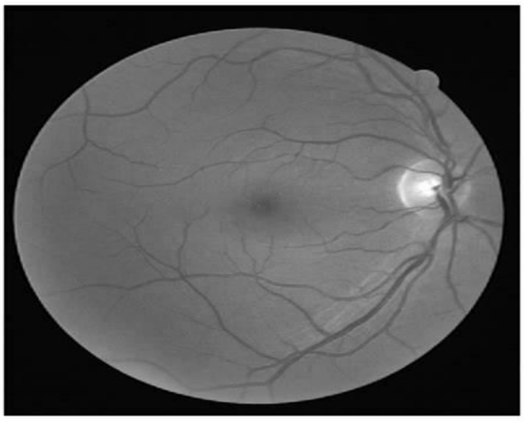
Average method is the simplest one. In this method, average of three colours is taken. Since it's an RGB image, so it means that take addition of R, G and B and then divide it by 3, it will give desired grayscale image.

Grayscale = (R+G+B)/3

Input Image:



Output Image:



Drawbacks of Average method:

It gives more black coloured image, this problem arises due to the fact, and that average of the three colours is taken. Since the three different colours have three different wavelengths and have their own contribution in the formation of image, it should take average according to their contribution, but in average method, it considered that all have equal contribution 33% of each. But in reality, it is not the case. The solution to this has been given by weighted method.

2] Weighted method or luminosity method:

The problem that occur in the average method can be overcome by weighted method. Since red colour (7000A0) has more wavelength among all the three colours and blue colour (4700A0) has less wavelength, so contribution of blue should less and red should high, but the green colour gives more soothing effect as the human perception is more sensitive to green Colour than other Colours, so green is weighted most heavily than Red and Blue. Hence, the Equation is:

Grayscale image = (0.2989 R+0.587 G+0.114. B)

According to this equation, Red has contributed 29.89%, Green has contributed 58.7% which is greater in all three colours and Blue has contributed 11.4%.

Output Image:

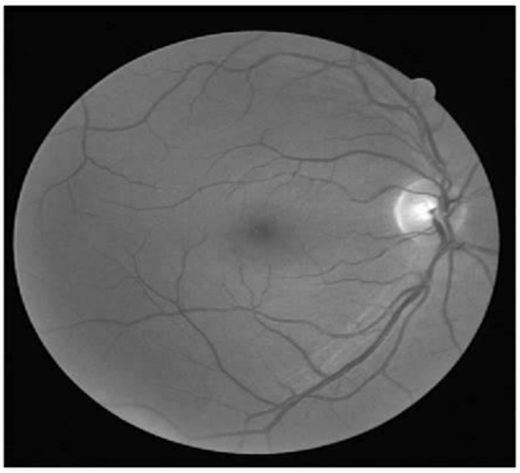


Fig : Grayscale retina image

<u>Procedure:</u>

Using MATLAB

1. Read the input image contents by ‘imread’ function and display
2. Calculate gray scale by using weighted average method
3. Display gray scale image

Using DIPLAB

1. Load the Image in processor memory
2. Note down the height and width of the Image which is loaded
3. Replace the value of the variable iImgWdt and iImgHgt with image width and height
4. Compile the programme.
5. Run the programme with Multi-Processor Run Mode
6. Read the output image from the processor memory defined in the programme

<u>Conclusion</u>:

Experiment No: 6

Title: Perform Histogram Processing and Equalization on an image using MATLAB

Software Used:

Programming Language – MATLAB® (Version 7 or higher)

Operating System – Microsoft Windows (XP 32-bit or higher)(Preferable)

Theory:

Statistical properties of an image

Mean:

The mean is an average if a set of numbers i.e., it is the central tendency (the way in which the quantitative data tend to cluster around some value) of a collection of numbers taken as the sum of the numbers divided by the size if the collection.

Variance:

The mean is not always the representative of the data, and other measures are needed to analyse the spread of the data. The variance is the measure of the distance of each number from the mean. In statistical approach, the variance is a measure of how far a set of number is spread out. It is one of the several descriptors of the probability distribution, describing how far the numbers lie from the mean (expected value). In particular, the variance is one of the moments of a distribution.

Standard Deviation:

The standard deviation is considerably larger than the mean. Standard deviation shows how much variation or dispersion exists from the average (mean or the expected value). A low standard deviation indicates that the data points tend to be very close to the mean, whereas high standard deviation indicates that the data points are spread out over a large range of values. It is equivalent with square root of variance.

Profile

Profile is useful in obtaining detail of the amplitude variations in an image of a selected path. In short, profile is the grey values representation along a line.

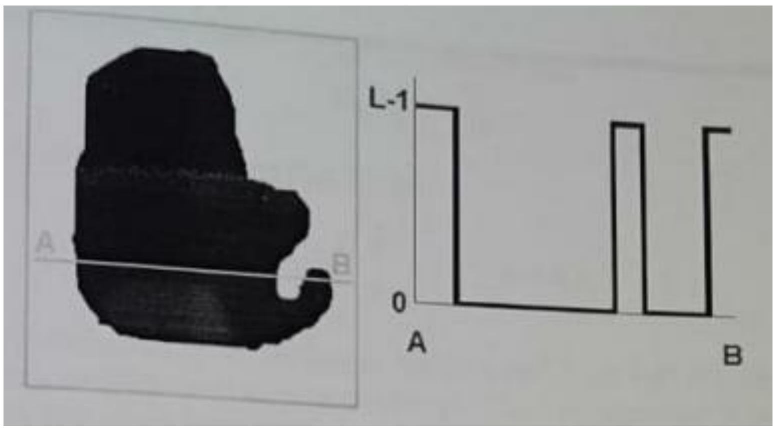


Fig. Profile of an image along a line

Histogram

Histogram of a digital image with gray levels in range (0, L-1) is a discrete function. It can be represented as:

$$h(r_k) = n_k$$

Where,

rk= kth Gray level, and

nk=no. of pixels of that image having intensity value(Gray level) rk.

In histogram there are 4 primary possibilities as follow:

1. For a dark image the components of histogram on the low side.
2. For a bright image the components are on high side.
3. For an image with low contrast they are in the middle of Gray side.
4. For an image with high contrast they are distributed or scattered.

Histogram manipulations can be used effectively for image enhancement operations.

Histogram Equalization

Histogram equalization is a method in image processing of contrast adjustment using the images histogram.

Histogram equalization is done to spread the histogram components uniformly over the Gray scale as far as possible.

For an image of size M*N, Histogram equalization is obtained by the following mapping function:

$$\boldsymbol{S_k} = \mathbf{T}(\boldsymbol{r_k}) = (\mathbf{L-1}) \sum_{j=0}^{k} \boldsymbol{P_r}(\boldsymbol{r_j}), \qquad \mathbf{k = 0,1,2,\ldots,L-1}$$

Where, $\boldsymbol{S_k}$ = output intensity level

$\boldsymbol{r_k}$= input intensity level after mapping

If n_k = total no. of pixels of intensity $\boldsymbol{r_k}$, we can conclude that,

$$\boldsymbol{P_r}\ (\boldsymbol{r_k}) = \frac{\boldsymbol{n_k}}{\boldsymbol{MN}}$$

And thus, we can modify the above function as:

$$\boldsymbol{S_k} = \mathbf{T}(\boldsymbol{r_k}) = \frac{\boldsymbol{L-1}}{\boldsymbol{MN}} \sum_{j=0}^{k} \boldsymbol{n_j}$$

Thus, processed image is obtained by mapping each pixel with intensity level $\boldsymbol{r_k}$ into a corresponding pixel with output image. This transformation is called Histogram equalization.

<u>Procedure:</u>

1. Read the input image contents by ‘imread’ function. Convert it to grayscale if necessary.

2. Obtain and display the histogram of the original image using ‘imhist’ function.
3. Perform histogram equalization on the original image by using ‘histeq’ function and display.
4. Compare the histogram of the original image and its histogram equalization version. Comment your observation in conclusion.
5. Perform histogram equalization respectively on the input image. Check histogram for each iteration of the histogram equalization and comment your observation in the conclusion.

<u>Conclusion</u>:

Experiment No: 7

Title: Perform morphological operations such as Dilation, Erosion, Opening and Closing on an image using MATLAB

Software Used:

Programming Language – MATLAB® (Version 7 or higher)

Operating System – Microsoft Windows (XP 32-bit or higher)(Preferable)

Theory:

Morphological Operations

Morphology is a tool for extracting image components that are useful in the representation and description of region shape such as boundaries, skeletons and the convex hull.

Some morphological operations

Basic definitions:

Let, A, B : sets in 2-D integer space i.e. Z^2 with components a = (a1,a2) and b=(b1,b2).

Then,

Translation of A by point z = (z_1 ,z_2) :

$$(A)_z = \{c \mid c = a+z \text{ , for } a \in A\}$$

Reflection of B:

$$\hat{B} = \{w \mid w = -b, \text{ for } b \in B\}$$

Complement of A:

$$A^c = \{ x \mid x = \bar{\text{I}}\text{A} \}$$

Difference of A & B:

$$\mathbf{A - B = A \cap \mathit{B}^{c}}$$

1. Dilation

It expands an image.

Dilation of A by B can be represented as:

$$A \oplus B = \left\{ z \,\middle|\, (\hat{B})_z \cap A \neq \emptyset \right\}$$

$$A \oplus B = \left\{ z \,\middle|\, [(\hat{B})_z \cap A] \subseteq A \right\}$$

Where, B is the structuring element in dilation and ∅ is empty set.

2. Erosion

It shrinks an image.

Erosion of A by B can be represented as:

$$A \ominus B = \{ z \mid (B)_z \subseteq A \}$$

$$A \ominus B = \{ z \mid (B)_z \cap A = \emptyset \}$$

3. Opening

Generally smoothens the contour of an image, eliminate protrusions.

$$A \circ B = (A \ominus B) \oplus B$$

4. Closing

Generally smoothens section of the contours, but it generally fuses breaks, holes, gaps etc.

$$A \bullet B = (A \oplus B) \ominus B$$

5. Boundary extraction (or edge extraction)

It is the subtraction of the eroded image from the original image.

$$A - (A \ominus B)$$

<u>Procedure:</u>

1. Open a binary or grayscale image by using ‘imread’ function.
2. Select structure element by using ‘strel’ function and store in variable ‘se’.
3. Perform dilation on the image by ‘se’, using ‘imdilate’ function.
4. Perform erosion on the image by ‘se’, using ’imerode’ function.
5. Perform opening on the image by ‘se’, using ‘imopen’ function.
6. Perform closing on the image by ‘se’, using ‘imclose’ function.
7. Perform boundary/edge extraction by subtracting eroded image from the original image.
8. Display all results.

<u>Conclusion</u>:

Experiment No: 8

Title: Perform Edge detection using Prewitt, Sobel, Isotropic, Roberts, Canny gradient operators in MATLAB

Software Used:

Programming Language – MATLAB® (Version 7 or higher)

Operating System – Microsoft Windows (XP 32-bit or higher)(Preferable)

Theory:

Edge Detection using Gradient Operators

The spatial filter are used to detect edges in an image are also called as High pass as these filters have effect on high frequency areas i.e. edges and boundaries in the image . Accuracy of edge detection depends up on the values in the filter and filter size. Edge detection depends up on the sum value of the filter.

All high pass filter has both positive and negative values i.e. zero cross over edge. Edge detection removes the low frequency information with the image leaving only edge visible i.e. sum value of filter must be 0. Also the filter must be positive and negative i.e. inverse symmetric nature i.e. it's both diagonal and central line elements must be zero.

Edge detection is the most common approach for detecting meaningful discontinuity in gray level. An edge is a set of connected pixels that is on the boundary between two regions. We define a point in an image as an edge point if its 2D first order derivative is greater than a specific threshold. A set of such connected points defines an edge.

Many method of edge detection via spatial mask filter with varying degrees of success are available, as listed in following table:

Name	Horizontal (X) Gradient	Horizontal (Y) Gradient
Prewitt	$\begin{bmatrix} \mathbf{-1} & \mathbf{0} & \mathbf{1} \\ \mathbf{-1} & \mathbf{0} & \mathbf{1} \\ \mathbf{-1} & \mathbf{0} & \mathbf{1} \end{bmatrix}$	$\begin{bmatrix} \mathbf{-1} & \mathbf{-1} & \mathbf{-1} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{1} & \mathbf{1} & \mathbf{1} \end{bmatrix}$
Sobel	$\begin{bmatrix} \mathbf{-1} & \mathbf{0} & \mathbf{1} \\ \mathbf{-2} & \mathbf{0} & \mathbf{2} \\ \mathbf{-1} & \mathbf{0} & \mathbf{1} \end{bmatrix}$	$\begin{bmatrix} \mathbf{-1} & \mathbf{-2} & \mathbf{-1} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{1} & \mathbf{2} & \mathbf{1} \end{bmatrix}$
Isotropic	$\begin{bmatrix} \mathbf{-1} & \mathbf{0} & \mathbf{1} \\ -\sqrt{\mathbf{2}} & \mathbf{0} & \sqrt{\mathbf{2}} \\ \mathbf{-1} & \mathbf{0} & \mathbf{1} \end{bmatrix}$	$\begin{bmatrix} \mathbf{-1} & -\sqrt{\mathbf{2}} & \mathbf{-1} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{1} & \sqrt{\mathbf{2}} & \mathbf{1} \end{bmatrix}$
Roberts	$\begin{bmatrix} \mathbf{0} & \mathbf{1} \\ \mathbf{-1} & \mathbf{0} \end{bmatrix}$	$\begin{bmatrix} \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{-1} \end{bmatrix}$

--Some common gradient operators--

(Bold elements indicates the location of the origin)

Mask size of 2 cross 2 are not useful to implement because they do not have clean centre So, it is tradition to use filter with odd size like 3 cross 3.

Prewitt and Sobel operators are most used in practice for computing digital gradient. Prewitt and Isotropic masks are somewhat similar to Sobel mask but the Sobel has slightly superior noise suppression characteristics.

Procedure:

1. Read the input image contents by imread function and display. Convert it in to grayscale if necessary.
2. Apply the gradient operator of selects spatial operator on the image by using ‘edge’ function.
3. Display the gradient images.

Conclusion:

Experiment No: 5

Title: Demonstration of convert between color spaces and perform filtering on color images in MATLAB

Software Used:

Programming Language – MATLAB® (Version 7 or higher)

Operating System – Microsoft Windows (XP 32-bit or higher)(Preferable)

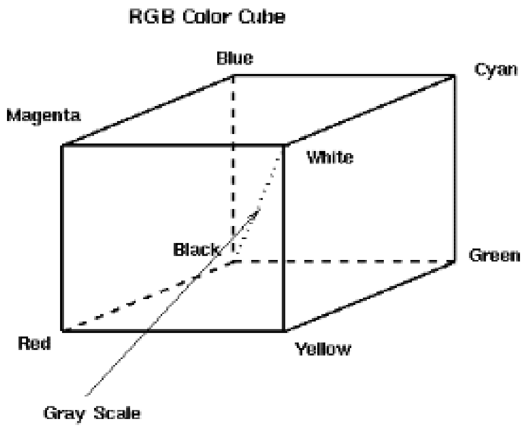
Theory:

The various color spaces exist because they present color information in ways that make certain calculations more convenient or because they provide a way to identify colors that is more intuitive. For example, the RGB color space defines a color as the percentages of red, green, and blue hues mixed together. Other color models describe colors by their hue (shade of color), saturation (amount of gray or pure color), and luminance (intensity, or overall brightness).

Color spaces are different types of color modes, used in image processing and signals and system for various purposes. Some of the common color spaces are:

- RGB
- HSI
- YCbCr

RGB Color Model



In the RGB model, an image consists of three independent image planes, one in each of the primary colours: red, green and blue. Specifying a particular colour is by specifying the amount of each of the primary components present. Figure shows the geometry of the RGB colour model for specifying colours using a Cartesian coordinate system. The greyscale spectrum, i.e. those colours made from equal amounts of each primary, lies on the line joining the black and white vertices.

RGB to HSI Conversion

In HSI model, color can be specified by three attributes/quantities namely hue, saturation and intensity.

Hue is a color attribute which is used to describe a pure form of color (may be pure yellow, red or orange). Saturation is a measure of degree to which a pure color is diluted by white light.

Conversion formula:

$$\theta = \cos^{-1}\left\{ \frac{\frac{1}{2}\left[(R-G)+(R-B)\right]}{\left[(R-G)^2+(R-B)(G-B)\right]^{1/2}} \right\}$$

$$\text{H(Hue)} = \begin{cases} \theta & \text{If } B<=G \\ 360-\theta & \text{If } B>G \end{cases}$$

$$\text{S(Saturation)} = 1- \frac{3}{(R+G+B)}\left[\min(R,G,B)\right]$$

$$\text{I(Intensity)} = \frac{1}{3}(R+G+B)$$

RGB to YCbCr Conversion

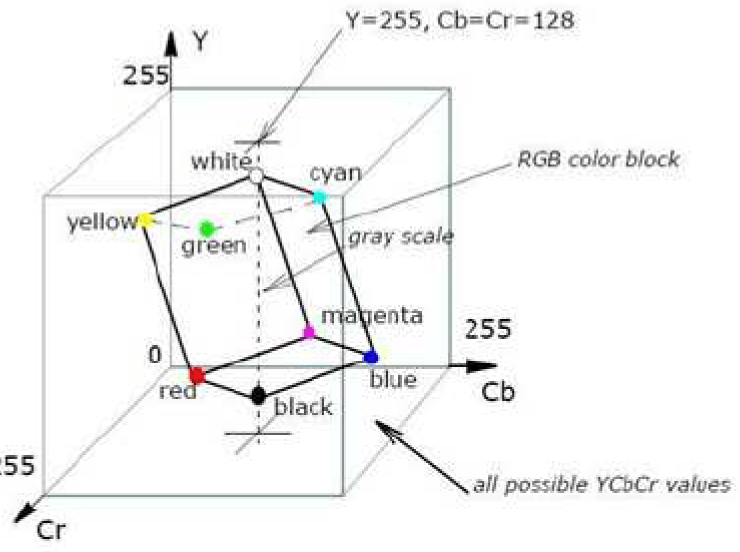


Fig. YCbCr Color Spaces

The YCbCr color space is widely used for digital video. In this format, luminance information is stored as a single component (*Y*) and chrominance information is stored as two color-difference components (*Cb* and *Cr*). Cb and Cr represent the difference between a reference value and the blue or red component, respectively.

The basic equation to convert between RGB and YCbCr are:

$$\mathbf{Y = 0.299R + 0.587G + 0.114B}$$

$$\mathbf{Cb = 128 - 0.168R - 0.3313G + 0.5B}$$

$$\mathbf{Cr = 128 + 0.5R - 0.4187G - 0.0813B}$$

Median Filter

Median filtering is a nonlinear method used to remove noise from images. It is widely used as it is very effective at removing noise while preserving edges. It is particularly effective at removing ‘salt and pepper’ type noise.

The median filter works by moving through the image pixel by pixel, replacing each value with the median value of neighboring pixels. The pattern of neighbors is called the "window", which slides, pixel by pixel, over the entire image. The median is calculated by first sorting all the pixel values from the window into numerical order, and then replacing the pixel being considered with the middle (median) pixel value.

Procedure:

1. Read the input RGB image contents by ‘imread’ function and display.
2. Convert input image to gray image and display.
3. Convert input image to HSI image by implementation of formulae of H, S, I and display.
4. Convert input image to YCbCr color spaces and display
5. Apply Median filter on RGB image

Conclusion:

Experiment No: 3

Title: Demonstration negation of image using MATLAB and DIPLAB- 1.0

Software Used:

Programming Language – MATLAB® (Version 7 or higher), VisualDSP++

Operating System – Microsoft Windows (XP 32-bit or higher)(Preferable)

Theory:

The principle objective behind image enhancement is to process a given image so that the result is more suitable than original image for specific application. Negation is one of the image enhancement technique which is used to print photographic negative of the image or to make features in an image appear clearer to the human observer .It can also be useful for medical images where objects often appear in black or white background. Inverting the image makes object appear makes the objects appear in white on a dark background, which is often more suitable for human eye.

The transformation function has been given below:

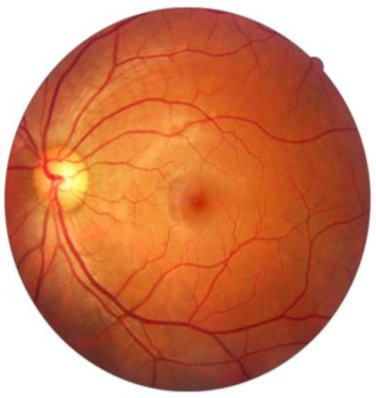
$$Q=T(P)$$

Where P is the pixels of input image and (Q) pixels of the output image . T is the transformation function that maps each value of (P) to each value of (Q).Image enhancement can be done through gray level transformation which is discussed below.

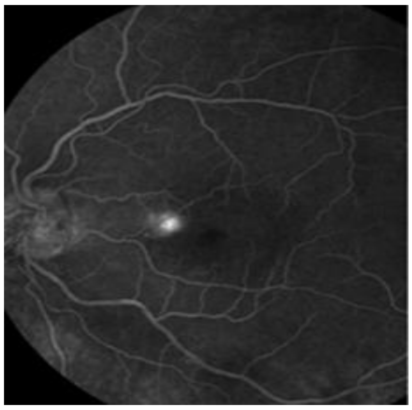
Negative Transformation

This is negative image transformation which is invert of the identity transformation. In negative transformation each value of the input image is inverted i.e. it is subtracted from max. Value is that colour space and mapped onto the output image.

Input Image:



Output Image:



In this case following transmission has been done

$$Q = (L-1) - P$$

Since the input image is 8 bit image so the number of levels in this image are 256 (L-1)=255, so equation becomes

$$Q= 255-P$$

Hence each value is subtracted by 255 and the result image has been shown above .It can be seen that the lighter pixels becomes dark and the darker pixels becomes light and it results in image negative.

Procedure:

Using MATLAB

1. Read the input image contents by ‘imread’ function and display
2. Perform negative operation on image
3. Display input and output images

Using DIPLAB

1. Load the image in processor memory
2. Note down the height and width of image which is loaded.
3. Replace the value of variable iImgWdt and iImgHgt with image width and height
4. Compile the program
5. Run the program with multi-processor Run Mode
6. Read the output image from the processor memory defined in the program

Conclusion:

Experiment No: 1

Title: Demonstration of Image Smoothing operation

Software Used:

Programming Language – MATLAB® (Version 7 or higher)

Operating System – Microsoft Windows (XP 32-bit or higher) (Preferable)

Theory:

Smoothing Spatial Filters

Smoothing filters are used for blurring and for noise reduction. Blurring is used for the pre-processing steps such as removal of small details from an image prior to object extraction. These filters are also called as low pass filters or averaging filters.

The idea behind smoothing is by replacing the value of every pixel in an image by average of the gray levels is the neighborhood defined by the smoothing filter mask. This process result in an image with reduced sharp transformation in gray levels.

Because random noise typically consists of sharp transitions in gray levels, the most obvious application of the smoothing is noise reduction. Averaging filters have the undesirable side effects that they blur edges. Another application of this type of process induces the smoothing of false contour that results from using insufficient number of gray levels.

$$\frac{1}{9}\begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline\end{array}$$

Fig. Standard average smoothing spatial filter

Smoothing using Gaussian Filter

The effect of Gaussian smoothing is to blur an image, in a similar fashion to the mean filter. The degree of smoothing is determined by the standard deviation of the Gaussian. (Larger standard deviation Gaussians, of course, require larger convolution kernels in order to be accurately represented.)

The Gaussian outputs a `weighted average' of each pixel's neighborhood, with the average weighted more towards the value of the central pixels. This is in contrast to the mean filter's uniformly weighted average. Because of this, a Gaussian provides gentler smoothing and preserves edges better than a similarly sized mean filter.

In 2-D, an isotropic (*i.e.* circularly symmetric) Gaussian has the form:

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Increasing the standard deviation continues to reduce/blur the intensity of the noise, but also attenuates high frequency detail (*e.g.* edges) significantly.

<u>Procedure:</u>

1. Read the input image contents by ‘imread’ function and display. Convert it into grayscale if necessary.
2. For smoothing, apply Standard Average Smoothing Spatial Filter with different sizes
3. Also apply Gaussian filter with different standard deviation values.
4. Display smoothened images

<u>Conclusion</u>: