# 

# EXPERIMENT – 1

## Aim:

Introduction to MATLAB

## Software used:

MATLAB

## Theory:

MATLAB (matrix laboratory) is a fourth-generation high-level programming language and interactive environment for numerical computation, visualization and programming.

MATLAB is developed by MathWorks.

It allows matrix manipulations; plotting of functions and data; implementation of algorithms; creation of user interfaces; interfacing with programs written in other languages, including C, C++, Java, and FORTRAN; analyse data; develop algorithms; and create models and applications.

It has numerous built-in commands and math functions that help you in mathematical calculations, generating plots, and performing numerical methods.

MATLAB's Power of Computational Mathematics

MATLAB is used in every facet of computational mathematics. Following are some commonly used mathematical calculations where it is used most commonly −

* Dealing with Matrices and Arrays
* 2-D and 3-D Plotting and graphics
* Linear Algebra
* Algebraic Equations
* Non-linear Functions
* Statistics
* Data Analysis
* Calculus and Differential Equations
* Numerical Calculations
* Integration
* Transforms
* Curve Fitting
* Various other special functions

Features of MATLAB

Following are the basic features of MATLAB −

* It is a high-level language for numerical computation, visualization and application development.
* It also provides an interactive environment for iterative exploration, design and problem solving.
* It provides vast library of mathematical functions for linear algebra, statistics, Fourier analysis, filtering, optimization, numerical integration and solving ordinary differential equations.
* It provides built-in graphics for visualizing data and tools for creating custom plots.
* MATLAB's programming interface gives development tools for improving code quality maintainability and maximizing performance.
* It provides tools for building applications with custom graphical interfaces.
* It provides functions for integrating MATLAB based algorithms with external applications and languages such as C, Java, .NET and Microsoft Excel.

Uses of MATLAB

MATLAB is widely used as a computational tool in science and engineering encompassing the fields of physics, chemistry, math and all engineering streams. It is used in a range of applications including −

* Signal Processing and Communications
* Image and Video Processing
* Control Systems
* Test and Measurement
* Computational Finance
* Computational Biology

## Result:

Hence studied introduction to MATLAB.

|  |  |  |  |
| --- | --- | --- | --- |
| Criteria | Total Marks | Marks Obtained | Comments |
| Concept(A) | 2 |  |  |
| Implementation(B) | 2 |  |  |
| Performance(C) | 2 |  |  |
| Total | 6 (To be scaled down to 1) |  |  |

# EXPERIMENT – 2

## Aim:

To perform vector operations, arithmetic operations, matrices operations and plotting graph using MATLAB command.

## Software used:

MATLAB

## Theory:

MATLAB has two different types of arithmetic operations: array operations and matrix operations. You can use these arithmetic operations to perform numeric computations, for example, adding two numbers, raising the elements of an array to a given power, or multiplying two matrices.

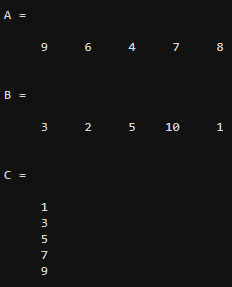
## Code:

**% 1D dimension**

A=[9 6 4 7 8]

B=[3 2 5 10 1]

C=[1;3;5;7;9]



**%ADD**

sum=A+B;

display(sum)

**%SUB**

diff=A-B;

display(diff)

**%MUL**

pro=A\*C;

display(pro)

**%DIV**

quo=A/B;

display(quo)

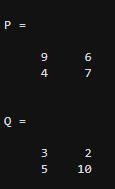
A screenshot of a computer screen

Description automatically generated with medium confidence

**% 2D dimension**

P=[9 6; 4 7]

Q=[3 2; 5 10]



**%ADD**

sum=P+Q;

display(sum)

**%SUB**

diff=P-Q;

display(diff)

**%MUL**

pro=P\*Q;

display(pro)

**%DIV**

quo=P/Q;

display(quo)

A screenshot of a computer screen

Description automatically generated with medium confidence

**%TRANSPOSE**

transpose(A)

transpose(B)

transpose(C)

transpose(P)

transpose(Q)

A screenshot of a computer

Description automatically generated with low confidence

**%INVERSE**

R=[5 3 1; 8 1 9; 5 4 5]

inv(R)

Text

Description automatically generated

**%RANDOM MATRIX**

randn(3)

rand(3)

A picture containing text

Description automatically generated

**%PLOT**

X=[9 6 3 5]

Y=[1 2 4 7]

xlabel('X-axis','FontSize',12,'Color','r')

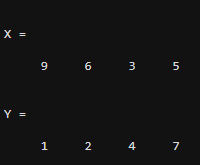
ylabel('Y-axis','FontSize',12,'Color','g')

title('X-Y Plot')

plot(X,Y)

grid on

legend



Chart, line chart

Description automatically generated

X=linspace(0,2\*pi,100);

Y1=cos(X)

hold on

plot(X,Y1,'color','c')

xlabel('X-axis','FontSize',12,'Color','r')

ylabel('Y-axis','FontSize',12,'Color','g')

title('X-Y Plot','Color','b')

grid on

Y2=sin(X)

plot(X,Y2,'color','m')

grid on

legend('cos(x)','sin(x)')

hold off

Chart, line chart

Description automatically generated

## Result:

Hence, performed vector operations, arithmetic operations, matrices operations and plotting graph using MATLAB command.

|  |  |  |  |
| --- | --- | --- | --- |
| Criteria | Total Marks | Marks Obtained | Comments |
| Concept(A) | 2 |  |  |
| Implementation(B) | 2 |  |  |
| Performance(C) | 2 |  |  |
| Total | 6 (To be scaled down to 1) |  |  |

# EXPERIMENT – 3

## Aim:

To read and display an image and find it centroid. Also, to extract its metadata.

## Software used:

MATLAB

## Theory:

A = imread (filename) reads the image from the file specified by filename, inferring the format of the file from its contents. If filename is a multi-image file, then imread reads the first image in the file.

imshow([I](https://in.mathworks.com/help/matlab/ref/imshow.html#bvmnrxi-1-I)) displays the grayscale image I in a figure. imshow uses the default display range for the image data type and optimizes figure, axes, and image object properties for image display. imshow([RGB](https://in.mathworks.com/help/matlab/ref/imshow.html#d123e740497)) displays the TrueColor image RGB in a figure. imshow([BW](https://in.mathworks.com/help/matlab/ref/imshow.html#d123e740549)) displays the binary image BW in a figure. For binary images, imshow displays pixels with the value 0 (zero) as black and 1 as white.

STATS = regionprops (L, properties) measures a set of properties for each labelled region in the label matrix L. Positive integer elements of L correspond to different regions. For example, the set of elements of L equal to 1 corresponds to region 1; the set of elements of L equal to 2 corresponds to region 2; and so on.

The image info function to create an Image Information tool. The tool displays information about the basic attributes and metadata of the target image in a separate figure. image info creates an Image Information tool associated with the image in the current figure.

## Code:

%read and display the image

image = imread("/MATLAB Drive/927310.jpg")

imshow(image)



%centroid of image

s = regionprops(image,'centroid');

centroids = cat(1,s.Centroid);

imshow(image)

hold on

plot(centroids(:,1),centroids(:,2),'b\*')

hold off



%metadata of image

info = imfinfo('/MATLAB Drive/927310.jpg') Text

Description automatically generated

Text

Description automatically generated

## Result:

Hence, read and displayed an image and found it centroid. Also, extracted its metadata.

|  |  |  |  |
| --- | --- | --- | --- |
| Criteria | Total Marks | Marks Obtained | Comments |
| Concept(A) | 2 |  |  |
| Implementation(B) | 2 |  |  |
| Performance(C) | 2 |  |  |
| Total | 6 (To be scaled down to 1) |  |  |

# EXPERIMENT – 4

## Aim:

Perform various point processing operations like negative, log and power law transformation on an image

## Software used:

MATLAB

## Theory:

Negative transformation:

The second linear transformation is negative transformation, which is invert of identity transformation. In negative transformation, each value of the input image is subtracted from the L-1 and mapped onto the output image.

## Logarithmic transformations:

Logarithmic transformation further contains two type of transformation. Log transformation and inverse log transformation.

The log transformations can be defined by this formula

s = c log(r + 1).

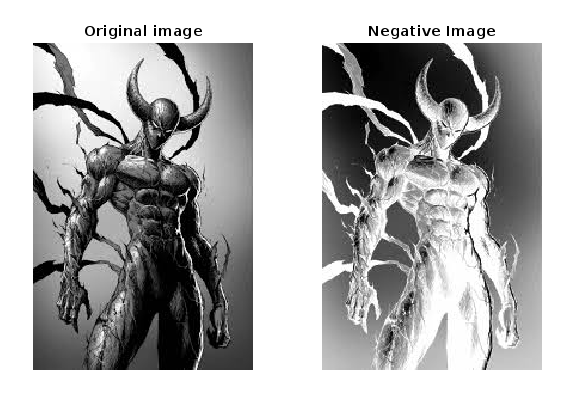
## Power – Law transformations:

There are further two transformation is power law transformations, that include nth power and nth root transformation. These transformations can be given by the expression:

s=cr^γ

This symbol γ is called gamma, due to which this transformation is also known as gamma transformation.

**Output:**

Image Negative:

Code:

Neg = imread("test1.jfif");

subplot(1, 2, 1),

imshow(neg);

title("Original image");

L = 2 ^ 8;

neg = (L - 1) - neg;

subplot(1, 2, 2),

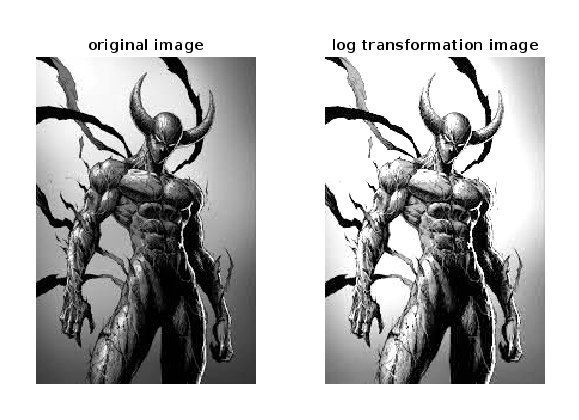
imshow(neg);

title("Negative Image")

Text

Description automatically generated

Log Transformation:



Code:

img1 = imread('test1.jfif'); % Read the image

d = double(img1)/255; % Normalized Image

c = 2; % Constant

t = c\*log(1 + (d)); % Log Transform

subplot(1,2,1)

imshow(img1)

title('original image');

subplot(1,2,2)

imshow((t))

title('log transformation image');

Text

Description automatically generated

Power Law Transformation:

Chart

Description automatically generated A picture containing text

Description automatically generated

Code:

I=imread('card9.png');

% To read image

figure,imshow(I)

p=rgb2gray(I);

% To convert RGB image to gray image(normalised image)

p=double(p);

figure,imshow(p/255)

[rowi,coli]=size(p);

r=0:1:255;

gamma=0.5;

c=1.5;

s=c\*r.^gamma;

out=zeros(rowi,coli);

plot(r,s)

for k=1:256

for i=1:rowi

for j=1:coli

if p(i,j)==r(k)

out(i,j)=s(k);

end

end

end

end

figure,imshow(uint8(out))

Text

Description automatically generated

|  |  |  |  |
| --- | --- | --- | --- |
| Criteria | Total Marks | Marks Obtained | Comments |
| Concept(A) | 2 |  |  |
| Implementation(B) | 2 |  |  |
| Performance(C) | 2 |  |  |
| Total | 6 (To be scaled down to 1) |  |  |

# EXPERIMENT – 5

## Aim:

To perform Gray level slicing with and without background on an image.

## Software used:

MATLAB

## Theory:

One of the most valuable points for Gray-level image enhancement is that in this technique, the Gray-level image enhancement techniques are directly performed on the specific pixel of an image. One of the most valuable points for Gray-level image enhancement is that the process is directly performed on the particular pixel of an image in this technique. The modification value of every single pixel of the processed image is dependent on the original pixel value.

Processing the Gray-level image is comparatively more efficient than working on the true colour image, so it is more focused and striking for researchers, like point process, etc. In this experiment, the pixel we are looking at has a two-dimensional Gray image with 256 levels, ranging from 0 to 255. The horizontal axis will be from 0 to 255, but the vertical axis is reliant on the number of pixels and the distribution of Gray-level values of an image.

Text

Description automatically generated

Code: -

clc; clear all;

i=imread('test1.jfif');

j=double(i);

k=double(i);

[row,col]=size(j);

T1=input('Enter the Lowest threshold value:');

T2=input('Enter the Highest threshold value:');

for x=1:row

for y=1:col

if((j(x,y)>T1) && (j(x,y)<T2))

j(x,y)=i(x,y);

k(x,y)=255;

else

j(x,y)=0;

k(x,y)=0;

end

end

end

subplot(311), imshow(i), title('Original image')

subplot(312), imshow(uint8(j)), title('Graylevel slicing with background')

subplot(313), imshow(uint8(k)), title('Graylevel slicing without background')

## Result:

Timeline

Description automatically generated

|  |  |  |  |
| --- | --- | --- | --- |
| Criteria | Total Marks | Marks Obtained | Comments |
| Concept(A) | 2 |  |  |
| Implementation(B) | 2 |  |  |
| Performance(C) | 2 |  |  |
| Total | 6 (To be scaled down to 1) |  |  |

# EXPERIMENT – 6

## Aim:

To display all bit planes of an image

## Software used:

MATLAB

## Theory:

Image is basically combination of individual pixel (dots) information. When we write that image is of 620 X 480 size, it means that image has 620 pixels in horizontal direction and 480 pixels in vertical direction. So, altogether there is 620 X 480 pixels and each pixels contains some information about image.

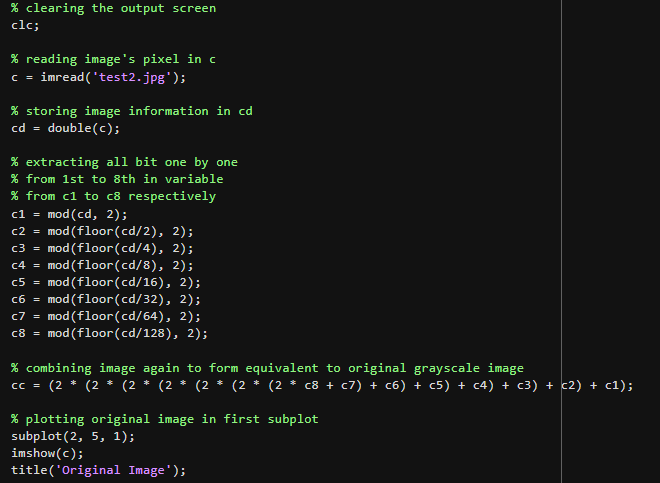
Grayscale image are basically those images which we say black and white image. Each pixel of grayscale image has a value lies in between 0 – 255 which decides at which position, the image will be black and at which position, it will be white.

If pixel value is 0, it means that pixel colour will be fully black and if pixel value is 255, then that pixel will be fully white and pixel having intermediate value will be having shades of black and white.

We will take a Grayscale Image. Since pixel value of grayscale image lies between 0 -255, so its information is contained using 8 bits. So, we can divide those images into 8 planes (8 Binary Image). Binary image are those images whose pixel value can be either 0 or 1. So, our task is to extract each bit planes of original image to make 8 binary images.

Let particular pixel of grayscale image has value 212. So, its binary value will be 11010100. So, its 1st bit is 0, 2nd is 0, 3rd is 1, 4rth is 0, 5th is 1, 6th is 0, 7th is 1, 8th is 1. In this manner, we will take these 8 bits of all pixels and will draw 8 binary images. We have to do this to all the pixels and generate new images.

Below is the implementation of above theory in MATLAB.



Text

Description automatically generated

Code: -

clc;

c = imread('test2.jpg');

cd = double(c);

c1 = mod(cd, 2);

c2 = mod(floor(cd/2), 2);

c3 = mod(floor(cd/4), 2);

c4 = mod(floor(cd/8), 2);

c5 = mod(floor(cd/16), 2);

c6 = mod(floor(cd/32), 2);

c7 = mod(floor(cd/64), 2);

c8 = mod(floor(cd/128), 2);

cc = (2 \* (2 \* (2 \* (2 \* (2 \* (2 \* (2 \* c8 + c7) + c6) + c5) + c4) + c3) + c2) + c1);

subplot(2, 5, 1);

imshow(c);

title('Original Image');

subplot(2, 5, 2);

imshow(c1);

title('Bit Plane 1');

subplot(2, 5, 3);

imshow(c2);

title('Bit Plane 2');

subplot(2, 5, 4);

imshow(c3);

title('Bit Plane 3');

subplot(2, 5, 5);

imshow(c4);

title('Bit Plane 4');

subplot(2, 5, 6);

imshow(c5);

title('Bit Plane 5');

subplot(2, 5, 7);

imshow(c6);

title('Bit Plane 6');

subplot(2, 5, 8);

imshow(c7);

title('Bit Plane 7');

subplot(2, 5, 9);

imshow(c8);

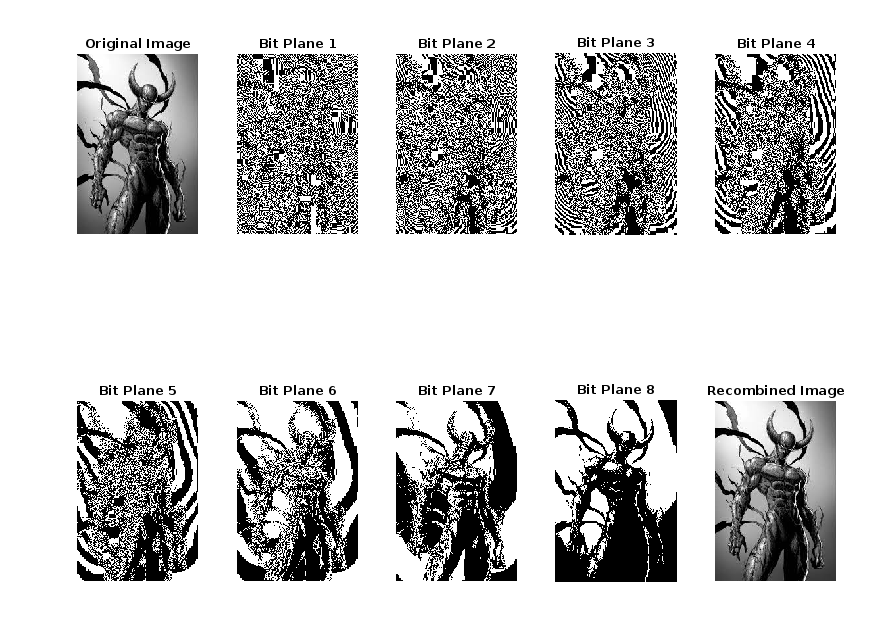
title('Bit Plane 8');

subplot(2, 5, 10);

imshow(uint8(cc));

title('Recombined Image');

## Result:



|  |  |  |  |
| --- | --- | --- | --- |
| Criteria | Total Marks | Marks Obtained | Comments |
| Concept(A) | 2 |  |  |
| Implementation(B) | 2 |  |  |
| Performance(C) | 2 |  |  |
| Total | 6 (To be scaled down to 1) |  |  |

# EXPERIMENT – 7

## Aim:

To display a histogram of an image and to perform histogram equalization on an image. Software used

## Software used:

MATLAB

## Theory:

A histogram is a display of statistical information that uses rectangles to show the frequency of data items in successive numerical intervals of equal size. In the most common form of histogram, the independent variable is plotted along the horizontal axis and the dependent variable is plotted along the vertical axis. The histogram is a popular graphing tool. It is used to summarize discrete or continuous data that are measured on an interval scale. It is often used to illustrate the major features of the distribution of the data in a convenient form.

Histogram of An Image: -

The graphical representation of an image's pixel intensity values is called a histogram. It can be understood as the data structure that keeps track of all the frequencies of the image's pixel intensity levels.

Chart, histogram

Description automatically generated

Image 1: [Histogram of an Image](http://www.doc.gold.ac.uk/~mas02fl/MSC101/ImageProcess/HIPR/histgram.htm)

The pixel intensity levels of the image are shown on the X-axis in the figure above. Typically, the intensity level runs from 0 to 255. There is just one histogram for a grayscale image, whereas there are three 2-D histograms—one for each colour—for an RGB coloured image. The histogram's Y-axis displays the frequency or quantity of pixels with a certain intensity value.

Text

Description automatically generated

Code: -

clc;

I = imread("test.png");

figure

subplot(1,2,1)

imshow(I)

title('original image')

subplot(1,2,2)

imhist(I, 64)

title('before hist eq')

J = histeq(I);

figure

subplot(1,2,1)

imshow(J)

title('histogram eq')

subplot(1,2,2)

imhist(J,64)

title('after hist eq')

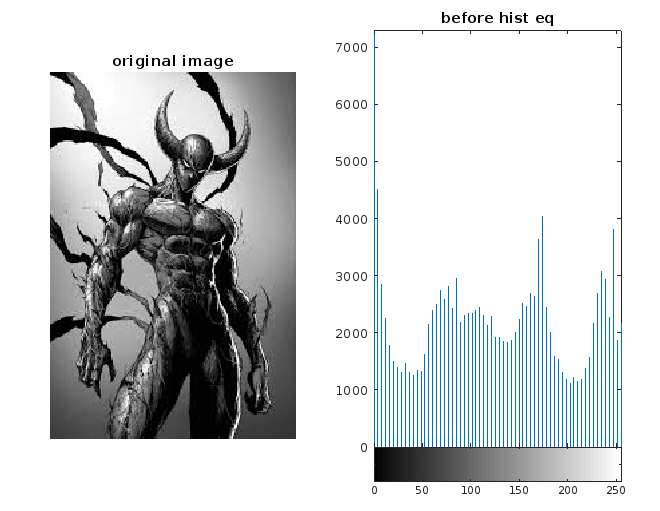


Image 2: before histogram equalization.

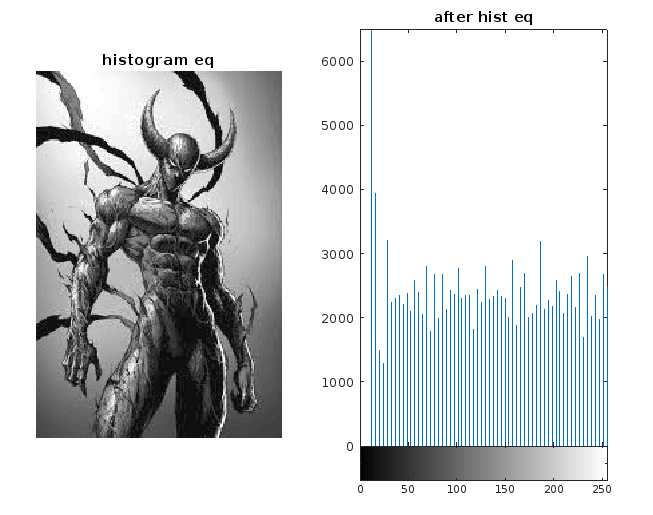


Image 3: after histogram equalization

|  |  |  |  |
| --- | --- | --- | --- |
| Criteria | Total Marks | Marks Obtained | Comments |
| Concept(A) | 2 |  |  |
| Implementation(B) | 2 |  |  |
| Performance(C) | 2 |  |  |
| Total | 6 (To be scaled down to 1) |  |  |

# EXPERIMENT – 8

## Aim:

Perform various colour transformations on an image.

Software used: MATLAB

## Theory:

Colour can be described by its red (R), green (G) and blue (B) coordinates (the well-known RGB system), or by some its linear transformation as XYZ, CMY, YUV, IQ, among others. The CIE adopted systems CIELAB and CIELUV, in which, to a good approximation, equal changes in the coordinates result in equal changes in perception of the colour. Nevertheless, sometimes it is useful to describe the colours in an image by some type of cylindrical-like coordinate system, it means by its hue, saturation and some value representing brightness. If the RGB coordinates are in the interval from 0 to 1, each colour can be represented by the point in the cube in the RGB space. Let us imagine the attitude of the cube, where the body diagonal linking” black” vertex and” white” vertex is vertical. Then the height of each point in the cube corresponds to the brightness of the colour, the angle or azimuth corresponds to the hue and the relative distance from the vertical diagonal corresponds to the saturation of the colour.

**Code:**

RGB = imread('MATLAB Drive/927310.jpg');

subplot(2,3,1);

imshow(RGB);

title('Original image')

% 1

HSV = rgb2hsv(RGB);

[h,s,v] = imsplit(HSV);

saturationFactor = 2;

s\_sat = s\*saturationFactor;

HSV\_sat = cat(3,h,s\_sat,v);

RGB\_sat = hsv2rgb(HSV\_sat);

subplot(2,3,2);

imshow(RGB\_sat)

title('HSV to RGB image')

% 2

RGB = reshape(ones(64,1)\*reshape(jet(64),1,192),[64,64,3]);

HSV = rgb2hsv(RGB);

[h,s,v] = imsplit(HSV); %Split the HSV version of the synthetic image into its component planes:

%hue, saturation, and value.

subplot(2,3,3);

montage({h,s,v,RGB},"BorderSize",10,"BackgroundColor",'w'); %Display the individual HSV color

%planes with the original image

%Split RGB Image into Its Component Channels

I = imread('MATLAB Drive/927310.jpg');

imshow(I)

[r,g,b] = imsplit(I);

subplot(2,3,4);

montage({r,g,b},'Size',[1 3])

%Split Image in HSV Colorspace into Its Component Channels

rgbImage = imread('MATLAB Drive/927310.jpg');

imshow(rgbImage)

hsvImage = rgb2hsv(rgbImage);

[h,s,v] = imsplit(hsvImage);

subplot(2,3,5);

title('final image');

montage({h,s,v},'Size',[1 3])

**Output:**

Graphical user interface

Description automatically generated with medium confidence

|  |  |  |  |
| --- | --- | --- | --- |
| Criteria | Total Marks | Marks Obtained | Comments |
| Concept(A) | 2 |  |  |
| Implementation(B) | 2 |  |  |
| Performance(C) | 2 |  |  |
| Total | 6 (To be scaled down to 1) |  |  |

# EXPERIMENT – 9

## Aim:

Enhance an Image and remove noise and other artifacts from an Image**.**

Software used: MATLAB

## Theory:

Image noise is random variation of brightness or colour information in images and is usually an aspect of electronic noise. It can be produced by the image sensor and circuitry of a scanner or digital camera. Image noise can also originate in film grain and in the unavoidable shot noise of an ideal photon detector.

The main types of image noise are random noise, fixed pattern noise, and banding noise.

There are typically three types of digital images: Binary images, Gray scale Images, Colour Images

**Code:**

I = imread('MATLAB Drive/927310.jpg');

subplot(2,3,1);

imshow(I);

se = strel('disk',15);

background = imopen(I,se);

imshow(background)

title('strel of the image');

I2 = I - background;

subplot(2,3,2);

imshow(I2);

title('rgb of the image');

gbird = rgb2gray(I2);

I3 = imadjust(gbird);

subplot(2,3,3);

imshow(I3);

title('gray of the image')

bw = imbinarize(I3);

bw = bwareaopen(bw,50);

subplot(2,3,4);

imshow(bw);

title('binary of the image');

cc = bwconncomp(bw,4);

cc.NumObjects;

grain = false(size(bw));

grain(cc.PixelIdxList{50}) = true;

subplot(2,3,5);

imshow(grain);

title('black of the image');

labeled = labelmatrix(cc);

whos labeled;

RGB\_label = label2rgb(labeled,'spring','c','shuffle');

subplot(2,3,6);

imshow(RGB\_label);

title('final image');

**Output:**

Graphical user interface, timeline

Description automatically generated

|  |  |  |  |
| --- | --- | --- | --- |
| Criteria | Total Marks | Marks Obtained | Comments |
| Concept(A) | 2 |  |  |
| Implementation(B) | 2 |  |  |
| Performance(C) | 2 |  |  |
| Total | 6 (To be scaled down to 1) |  |  |

# EXPERIMENT – 10

## Aim:

Perform colour-based segmentation on an Image.

Software used: MATLAB

## Theory:

Image segmentation is a method in which a digital image is broken down into various subgroups called Image segments which helps in reducing the complexity of the image to make further processing or analysis of the image simpler. Segmentation in easy words is assigning labels to pixels. All picture elements or pixels belonging to the same category have a common label assigned to them. For example: Let’s take a problem where the picture must be provided as input for object detection. Rather than processing the whole image, the detector can be inputted with a region selected by a segmentation algorithm. This will prevent the detector from processing the whole image thereby reducing inference time.

**Image Segmentation Techniques:**

1. Threshold Based Segmentation
2. Edge Based Segmentation
3. Region-Based Segmentation
4. Clustering Based Segmentation
5. Artificial Neural Network Based Segmentation

**Approaches in Image Segmentation**

1. **Similarity approach:** This approach is based on detecting similarity between image pixels to form a segment, based on a threshold. ML algorithms like clustering are based on this type of approach to segment an image.
2. **Discontinuity approach:** This approach relies on the discontinuity of pixel intensity values of the image. Line, Point, and Edge Detection techniques use this type of approach for obtaining intermediate segmentation results which can be later processed to obtain the final segmented image.

**Code:**

%Read Image

x = imread("MATLAB Drive/927310.jpg");

subplot(2,4,1);

imshow(x);

title("Original Image");

%Classify Colors in RBG Color Space Using K-Means Clustering

numColors = 3;

L = imsegkmeans(x,numColors);

B = labeloverlay(x,L);

subplot(2,4,2);

imshow(B);

title("Labeled Image RGB");

%Convert Image from RGB Color Space to L\*a\*b\* Color Space

lab\_he = rgb2lab(x);

%Classify Colors in a\*b\* Space Using K-Means Clustering

ab = lab\_he(:,:,2:3);

ab=im2single(ab);

pixel\_labels=imsegkmeans(ab,numColors,NumAttempts=3);

B2 = labeloverlay(x,pixel\_labels);

subplot(2,4,3);

imshow(B2);

title("Labeled Image a\*b\*");

%Create Images that Segment H&E Image by Color

mask1 = pixel\_labels == 1;

cluster1 = x.\*uint8(mask1);

subplot(2,4,4);

imshow(cluster1);

title("Objects in Cluster 1");

mask2 = pixel\_labels == 2;

cluster2 = x.\*uint8(mask2);

subplot(2,4,5);

imshow(cluster2)

title("Objects in Cluster 2");

mask3 = pixel\_labels == 3;

cluster3 = x.\*uint8(mask3);

subplot(2,4,6);

imshow(cluster3)

title("Objects in Cluster 3");

%Segment Nuclei

L=lab\_he(:,:,1);

L\_blue = L.\*double(mask3);

L\_blue = rescale(L\_blue);

idx\_light\_blue = imbinarize(nonzeros(L\_blue));

blue\_idx = find(mask3);

mask\_dark\_blue = mask3;

mask\_dark\_blue(blue\_idx(idx\_light\_blue)) = 0;

blue\_nuclei = x.\*uint8(mask\_dark\_blue);

subplot(2,4,7);

imshow(blue\_nuclei);

title("Blue Nuclei");

**Output:**

Graphical user interface, timeline

Description automatically generated

|  |  |  |  |
| --- | --- | --- | --- |
| Criteria | Total Marks | Marks Obtained | Comments |
| Concept(A) | 2 |  |  |
| Implementation(B) | 2 |  |  |
| Performance(C) | 2 |  |  |
| Total | 6 (To be scaled down to 1) |  |  |