# Certificate

This is to certify that

this Lab Work in the subject of

**Image Processing and Machine Vision (IPMV)**

of Semester **VI** of **Electronics and Telecommunication** (EXTC) course

(Academic Year 2023-2024)

submitted by

Mr. **Aniket Karkala** Roll No. **21104A0020**

is accepted by the Department

**Professor In charge**

(Dr. Varsha Turkar)

**Head of the Department**

(Dr. Varsha Turkar)

# Index

**Practical Session No.:** 1

**Date:** 16/01/2024

## Objective:

To perform gray level slicing on the image based on user requirements.

## Methodology/Algorithm/Related Theory:

- Read the image using imread() and extract their R, G and B layers. Convert the image into greyscale using the extracted layers.
- For enhancing the section, take the input from the user for their required co-ordinates on the greyscale intensity (i.e. r1, r2).
- Check whether it satisfies the condition if the input image is in the region r1<= R <= r2.
- Take a choice from the user whether he wants the image with or without background.
- Apply for loop and if-else statements to compute the image with or without background satisfying the above conditions and other conditions.

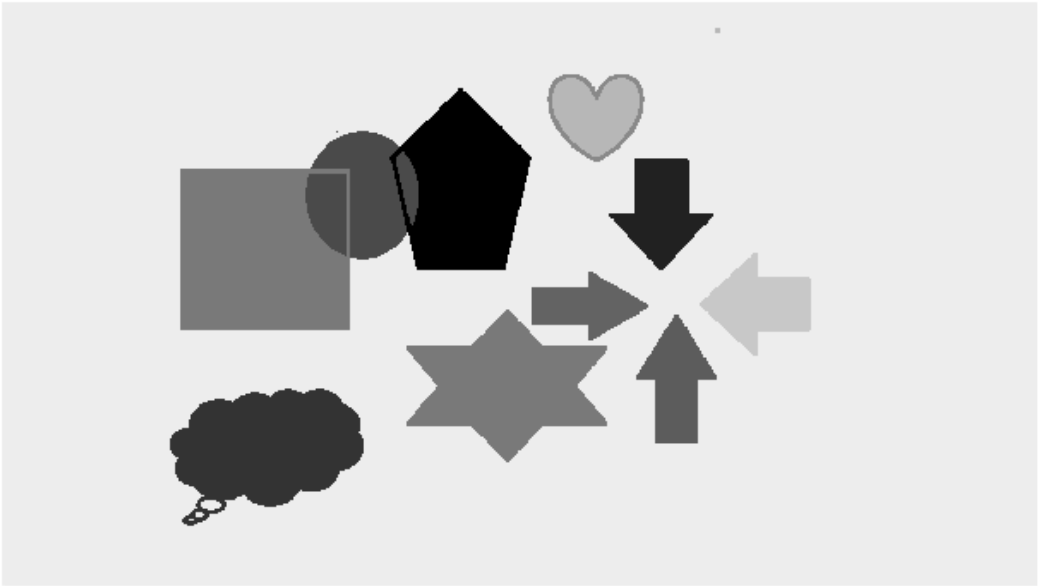
## Code:

```
clear;
clc;
A=imread("Image002.bmp"); %reading the image

%Extracting the RGB layers
r=A(:,:,1);
g=A(:,:,2);
b=A(:,:,3);

%Converting to Gray Scale
C= (0.229*r)+(0.587*g)+(0.114*b);
s=size(C)
imshow(C)

%Taking input from the user as co_ordinates of thresholding points
r1=input("Enter the value of r1:")
r2=input("Enter the value of r2:")
choice=0

%Do the required operations from user based choices
for i=1:100
    if choice==1
        for i=1:s(:,1)
            for j=1:s(:,2)
                if r1<=C(i,j) && C(i,j)<=r2
                    F(i,j)=255;
                else
                    F(i,j)=C(i,j);
                end
            end
```

```
        end
    imshow(F)
break

    elseif choice==2
        for i=1:s(:,1)
            for j=1:s(:,2)
                if r1<=C(i,j) && C(i,j)<=r2
                    F(i,j)=255;
                else
                    F(i,j)=0;
                end
            end
        end
    imshow(F)
break
    else
        choice=input("Enter your choice: \n Press 1 for with background \n or
Press 2 for without background: ")
    end
end
```

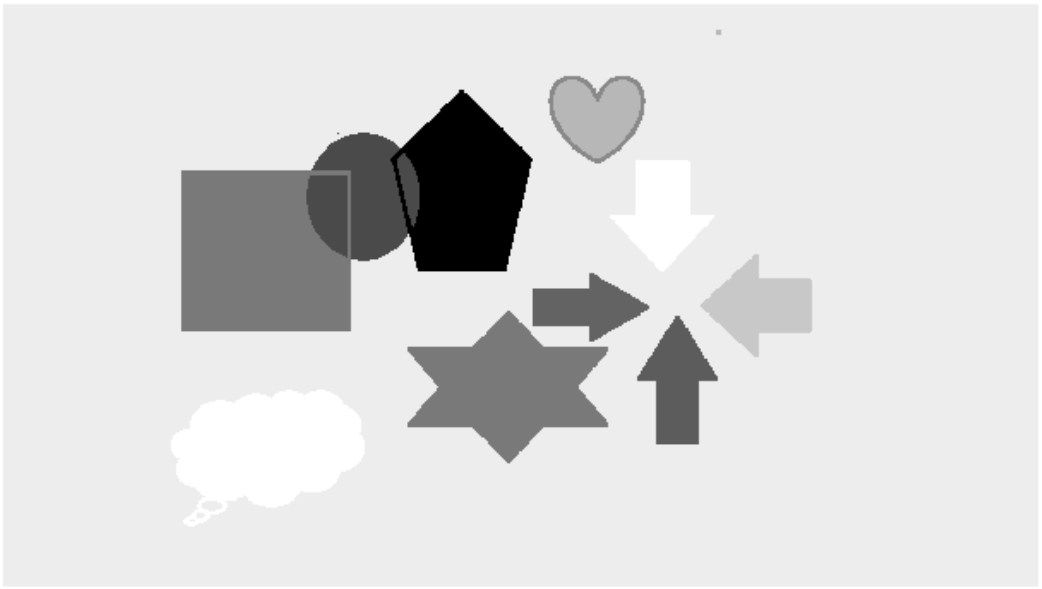
**Output:**

```
s = 1×2
         648        1152
```

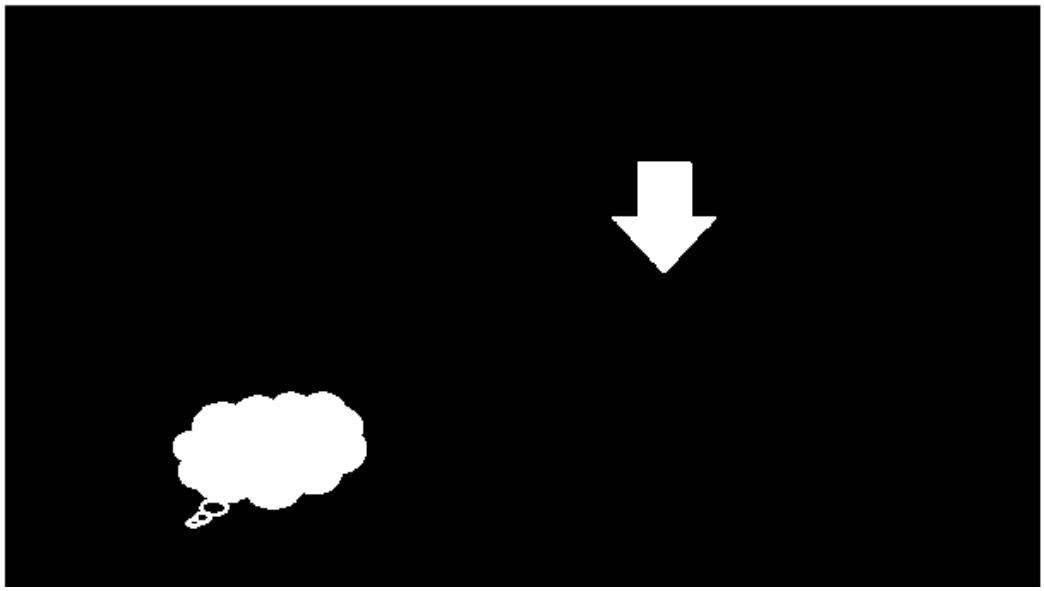


```
r1 = 30
r2 = 55
choice = 0
```

```
choice = 1
```



```
choice = 2
```



**Take away:**

We got a deep knowledge of contrast stretching logic and how it can be used to enhance sections of an image based on their intensity. This logic can be used to enhance the required section of an image to highlight or to brighten the required areas. Doing it so helps us in further processing of image as it parts the way for segmentation.

**Course outcome:**

CO mapped – CO1

After performing this experiment, students will be able to apply appropriate point operations to enhance the image quality.

**Practical Session No:** 2

**Date:** 23/01/2024

**Objective:** To perform Thresholding operation on image

**Methodology/Algorithm/Related Theory:**

For reading an image we use the built-in function imread("filename.bmp"). To show the image we use the function imshow(A).

The ".bmp" is a 24-bit i.e. a colour image. A colour image consists of three layers that are red, green and blue. For extracting the R, G and B component we index into the read image file.

Before performing any operation, we must convert the colour image into a grey-level image. This is performed by multiplying the R, G and B components with some factors (G is given the most weightage). A grey scale image is an 8-bit image having intensity values from 0 to 255.

The thresholding operation binarizes the image based on the given threshold. In this operation if the intensity of a pixel is more than the threshold, it is assigned "1" and if the intensity of a pixel is less than the threshold, it is assigned "0" (vice-versa is possible).

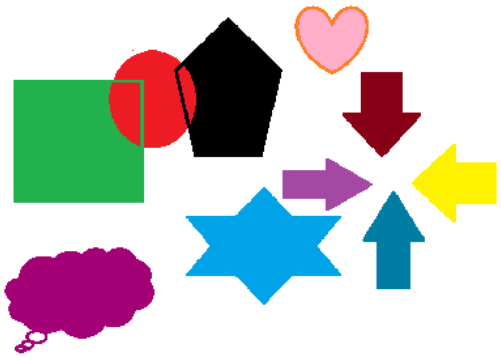
**Code:**

```
clc;
clear;
A = imread("Image002.bmp"); %reading the image
imshow(A)
title("24 bit image")

s = size(A);

%extracting RGB image layers
x=A(:,:,1); %red
y=A(:,:,2); %green
z=A(:,:,3); %blue


C1= (0.229*x)+(0.587*y)+(0.114*z); %converting the image into grayscale
size(C1);
imshow(C1)
title("8 bit image")



T = 100; %Thresholding logic
```

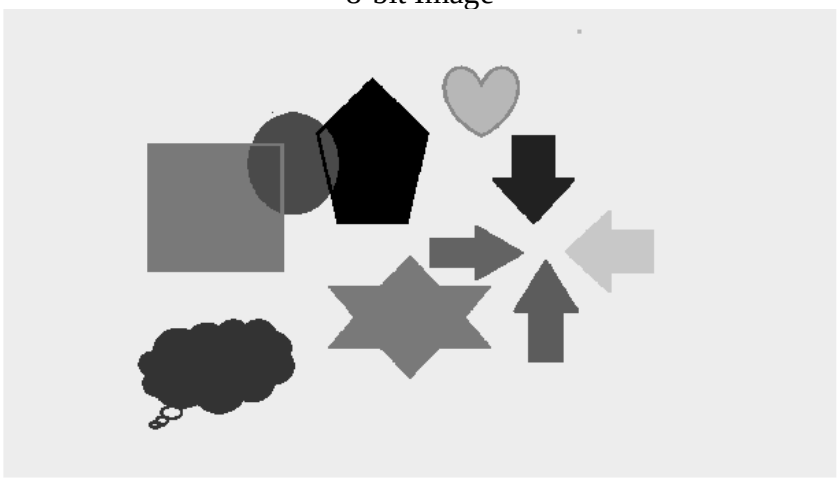
```
for i=1:s(:,1)
    for j=1:s(:,2)
        if C1(i,j)<T
            F(i,j)=0;
        else
            F(i,j)=255;
        end
    end
end
%subplot(3,2,6)
imshow(F)
title("Threshold Image ")
```

**Output:**

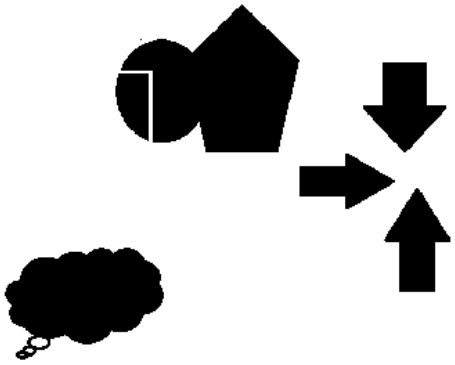
24 Bit image



8-bit Image



Thresholded Image



**Take away:**

By performing this experiment, we were able to read and display the given image. We learnt about the working of a built-in function ‘‘rgb2gray(A)’’ which is used to grey scale an image. The concept of thresholding can be applied on any to convert the given image into binary by specifying the required threshold. Also, the intensity mapping of an 8-bit image from levels 0 to 255 will help in categorizing the images.

**Course outcome:**

CO mapped – CO1

After performing this experiment, students will be able to apply appropriate point operations to enhance the image quality.

**Practical Session No:** 3

**Date:** 30/01/2024

**Objective:** To enhance the brightness of an image

**Methodology/Algorithm/Related Theory:**

For reading an image we use the built-in function imread("filename.bmp"). To show the image we use the function imshow(A).

Before performing any operation, we must convert the colour image into a grey-level image. This is performed by multiplying the R, G and B components with some factors (G is given the most weightage). A grey scale image is an 8-bit image having intensity values from 0 to 255.

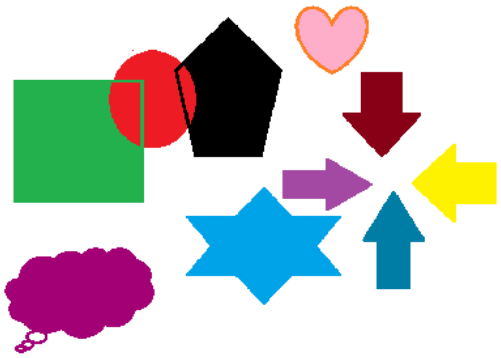
If we multiply the image with a multiplication-factor more than 1, the image is brightened as the intensities are increased by a factor of "(multiplication-factor)-1".

**Code:**

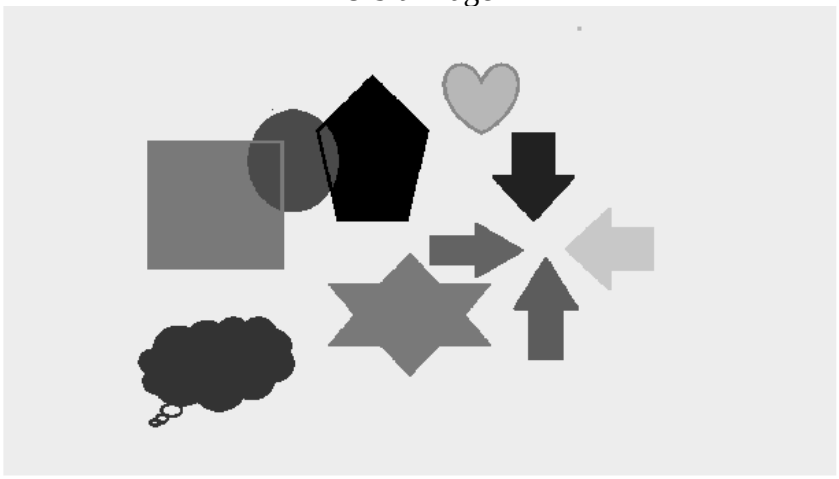
```
clc;
clear;
A = imread("Image002.bmp"); %reading the image
imshow(A)
title("24 bit image")

s = size(A);

%extracting RGB image layers
x=A(:,:,1); %red
y=A(:,:,2); %green
z=A(:,:,3); %blue


C1= (0.229*x)+(0.587*y)+(0.114*z); %converting the image into grayscale
size(C1);
imshow(C1)
title("8 bit image")


B=(1.5*A); %to change the brightness
size(B);
imshow(B)
title("Brightened image")
```

**Output:**

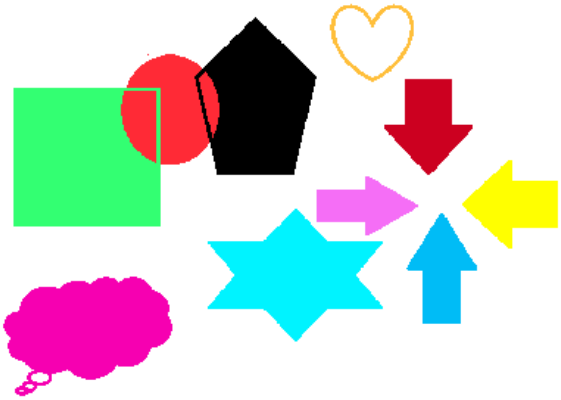
24 Bit image



8-bit Image



Brightened Image



**Take away:**

By performing this experiment, we were able to read and display the given image. We learnt about the working of a built-in function ‘‘rgb2gray(A)’’ which is used to grey scale an image. We observed effects of brightness operation

**Course outcome:**

CO mapped – CO1

After performing this experiment, students will be able to apply appropriate point operations to enhance the image quality.

**Practical Session No.:** 4

**Date:** 06/02/2024

**Objective:** To perform piece-wise contrast stretching on the image based on user requirements.

**Methodology/Algorithm/Related Theory:**

- Read the image using imread() and extract their R, G and B layers. Convert the image into greyscale using the extracted layers.
- For enhancing the particular section, take the input from the user for their required co-ordinates on the greyscale intensity (i.e. r1, r2, S1, S2).
- Check whether it satisfies the condition (r2-r1) < (S2-S1).
- Calculate the slope from user input co-ordinates and formulate a for loop which enhances the section based on the values of "r1" and "r2" and the enhancement factor is slope*image_pixel.

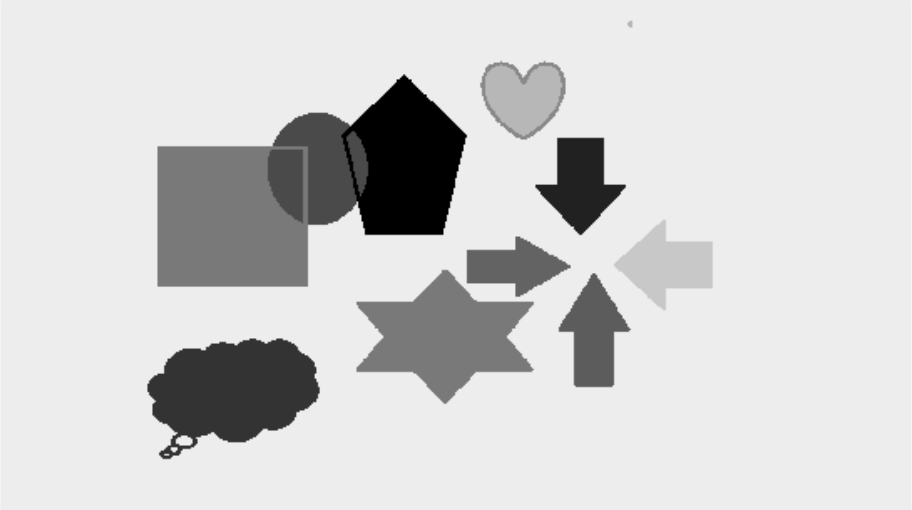
**Code:**

```
A=imread("Image002.bmp"); %reading the image
%Extracting the RGB layers
r=A(:,:,1);
g=A(:,:,2);
b=A(:,:,3);
%Converting to Gray Scale
C= (0.229*r)+(0.587*g)+(0.114*b);
s=size(C)
imshow(C)
%Taking input from the user as co-ordinates of thresholding points
S1=input("Enter the value of S1:")
S2=input("Enter the value of S2:")
r1=input("Enter the value of r1:")
r2=input("Enter the value of r2:")
%Checking whether input arguments satisfy the condition
if (r2-r1)<(S2-S1)
    disp("Inputs are correct")
end
%Calculating slopes for different region
m1=S1/r1
m2=(S2-S1)/(r2-r1)
m3=(255-S2)/(255-r2)
%Applying required operations depending on input values
for i=1:s(:,1)
    for j=1:s(:,2)
        if C(i,j)<r1
            S(i,j)=m1*C(i,j);
         elseif (r1<=C(i,j))<r2
            S(i,j)=m2*(C(i,j)-r1)+S1;
        else
            S(i,j)=m2*(C(i,j)-r2)+S2;
        end
    end
end
```

```
imshow(S)
```

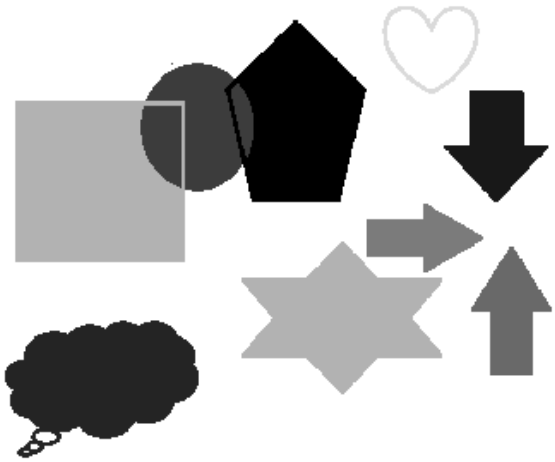
**Output:**

8 Bit Image



```
S1 = 50
S2 = 200
r1 = 70
r2 = 130
Inputs are correct
m1 = 0.7143
m2 = 2.5000
m3 = 0.4400
```

Resultant Image



**Take away:**

We got a deep knowledge of contrast stretching logic and how it can be used to enhance sections of an image based on their intensity. This logic can be used to enhance the required section of an image to highlight or to brighten the required areas. Doing it so helps us in further processing of image as it parts the way for segmentation.

**Course outcome:**

CO mapped – CO1

After performing this experiment, students will be able to apply appropriate point operations to enhance the image quality.

**Practical Session No:** 5

**Date:** 13/02/2024

**Objective:** To perform negation of an image

**Methodology/Algorithm/Related Theory:**

For reading an image we use the built-in function imread("filename.bmp"). To show the image we use the function imshow(A).

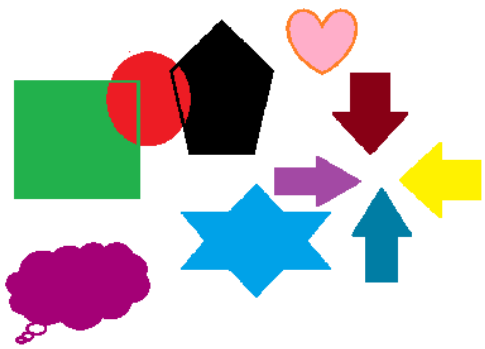
Negation is basically the inversion of the image. This is performed by subtracting the image intensities from 255.

**Code:**

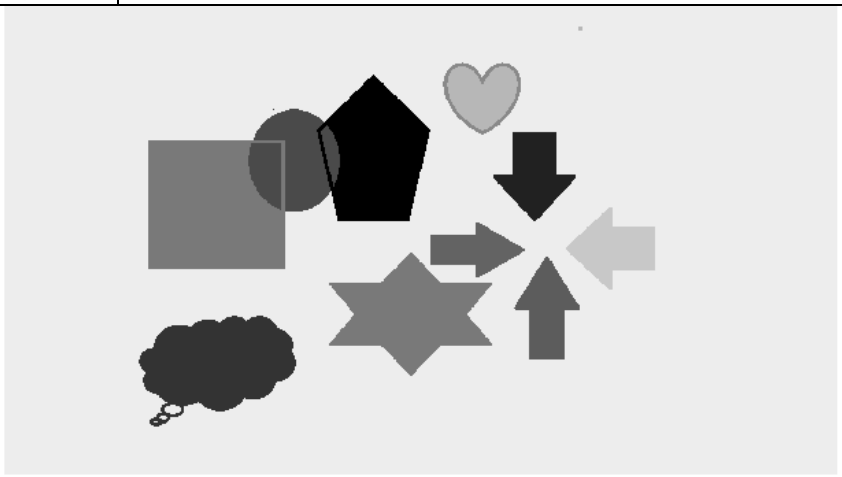
```
clc;
clear;
A = imread("Image002.bmp"); %reading the image
imshow(A)
title("24 bit image")
%extracting RGB image layers
x=A(:,:,1); %red
y=A(:,:,2); %green
z=A(:,:,3); %blue


C1= (0.229*x)+(0.587*y)+(0.114*z); %converting the image into grayscale
size(C1);
imshow(C1)
title("8 bit image")


S=255-C1; %negation or inversion by subtracting gray level value
imshow(S)
title("Negative image")
```

**Output:**

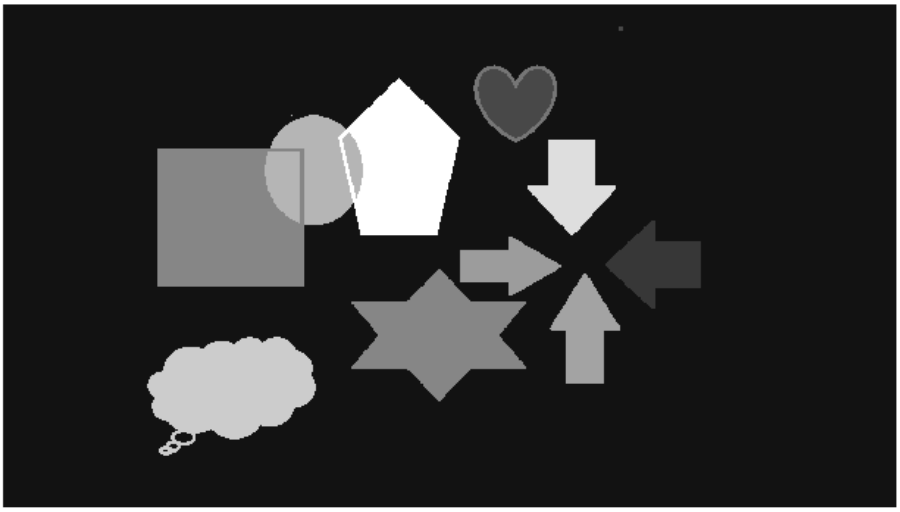
24 Bit image



8-bit Image



Negative Image



**Take away:**

By performing this experiment, we were able to read and display the given image. We learnt about the working of a built-in function ‘‘rgb2gray(A)’’ which is used to grey scale an image. We observed effect of negation on the image.

**Course outcome:**

CO mapped – CO1

After performing this experiment, students will be able to apply appropriate point operations to enhance the image quality.

**Practical Session No.:** 6

**Date:** 20/02/2024

## Objective:

To filter various images using inbuilt filters.

## Methodology/Algorithm/Related Theory:

- Read the image using imread().
- Create a filter mask using the inbuilt function "fspecial()".
- Apply the filter mask on the image using the 'imfilter(image,mask).
- The various inbuilt filters and their description are as follows:

| | |
|---|---|
| 'average' | Averaging filter |
| 'disk' | Circular averaging filter (pillbox) |
| 'gaussian' | Gaussian lowpass filter. |
| 'laplacian' | Approximates the two-dimensional Laplacian operator |
| 'log' | Laplacian of Gaussian filter |
| 'motion' | Approximates the linear motion of a camera |
| 'prewitt' | Prewitt horizontal edge-emphasizing filter |
| 'sobel' | Sobel horizontal edge-emphasizing filter |

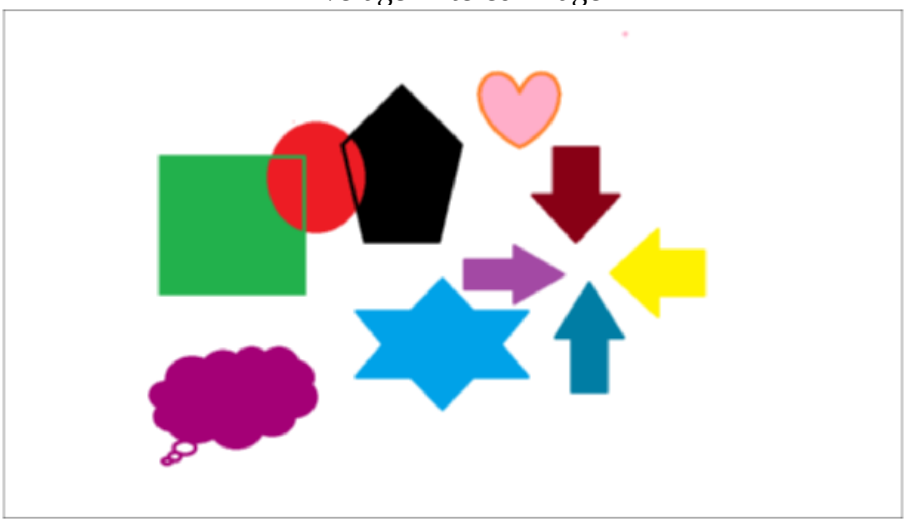
## Code:

```
clear;
clc;
A=imread("Image002.bmp"); %reading the image

F=fspecial("average",[5 5]); % Creating a filter mask of size 5 by 5
filtered_image1=imfilter(A,F); % Applying the filter mask on the image
imshow(filtered_image1)


F=fspecial("gaussian",[5 5],0.4); % Creating a filter mask of size 5 by 5 and
sigma
filtered_image2=imfilter(A,F); % Applying the filter mask on the image
imshow(filtered_image2)


F=fspecial("log",[5 5],0.4); % Creating a filter mask of size 5 by 5 and sigma
filtered_image3=imfilter(A,F); % Applying the filter mask on the image
imshow(filtered_image3)


F=fspecial("disk",5); % Creating a filter mask of radius 5
filtered_image4=imfilter(A,F); % Applying the filter mask on the image
imshow(filtered_image4)


F=fspecial("laplacian",0.3); % Creating a filter mask with alpha value
```

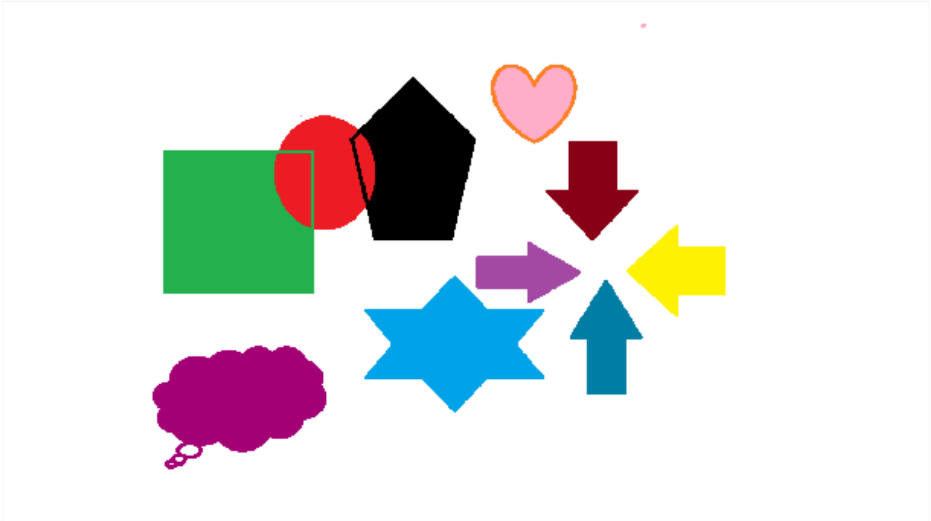
```
filtered_image5=imfilter(A,F); % Applying the filter mask on the image
imshow(filtered_image5)


F=fspecial("motion",10,60); % Creating a motion filter mask of len=5 and angle=60
filtered_image6=imfilter(A,F); % Applying the filter mask on the image
imshow(filtered_image6)


F=fspecial("prewitt"); % Creating a prewitt filter mask
filtered_image7=imfilter(A,F); % Applying the filter mask on the image
imshow(filtered_image7)


F=fspecial("sobel"); % Creating a sobel filter mask
filtered_image8=imfilter(A,F); % Applying the filter mask on the image
imshow(filtered_image8)
```

**Output:**

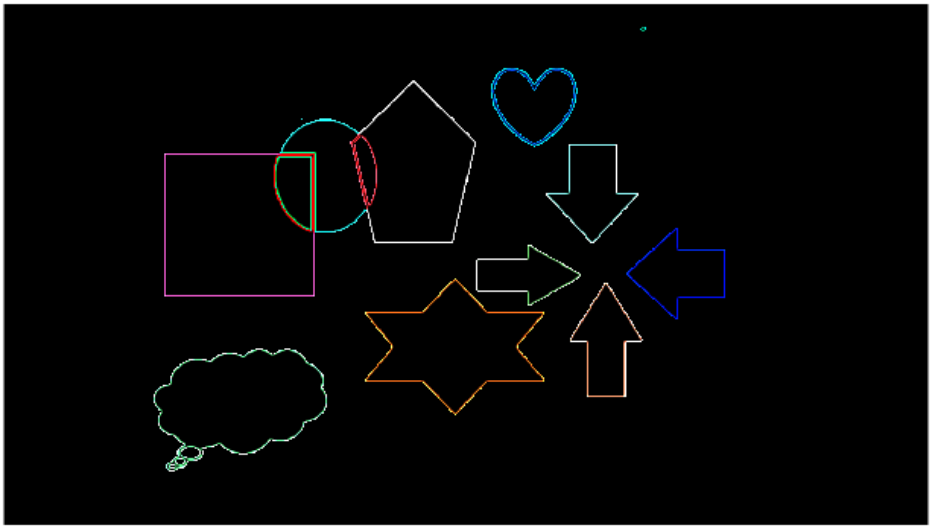
Average Filtered Image



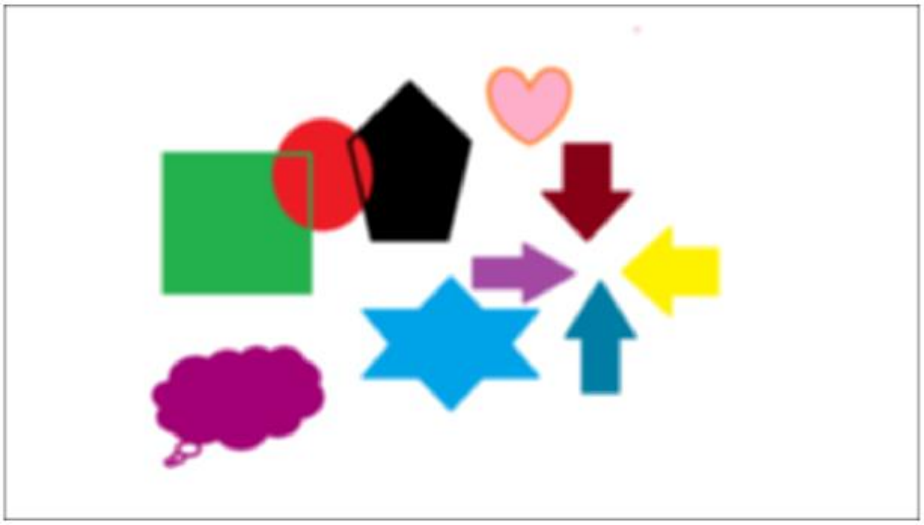
Gaussian Filtered Image



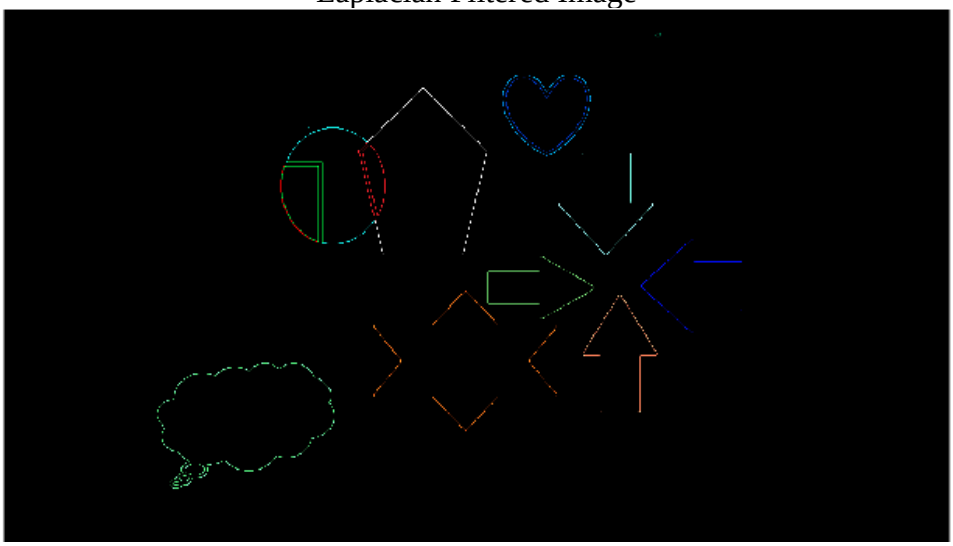
Log Filtered Image



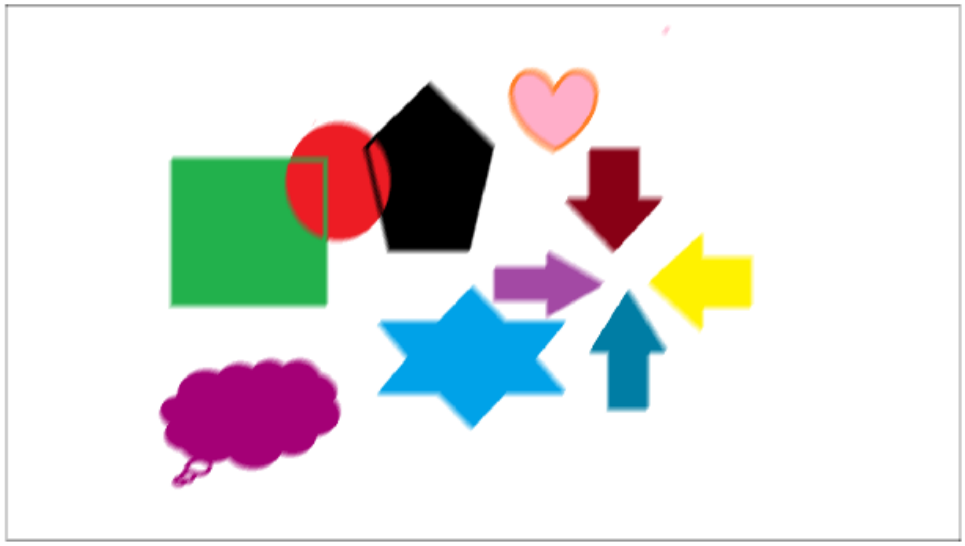
Disk Filtered Image



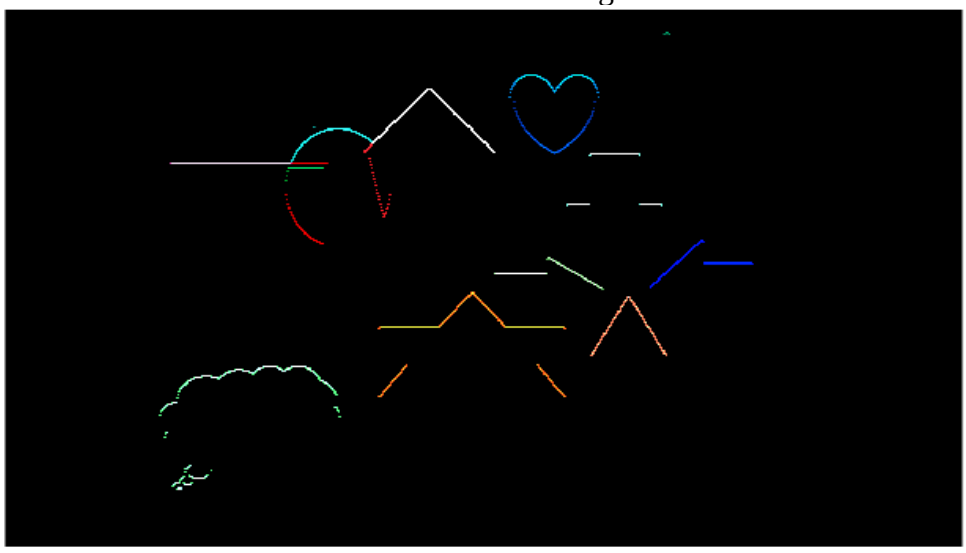
Laplacian Filtered Image



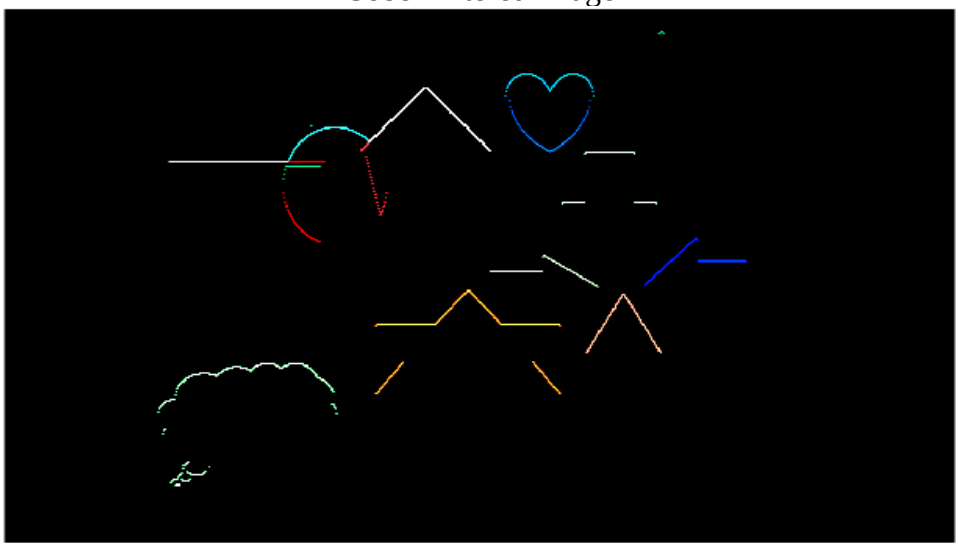
Motion Filtered Image



Prewitt Filtered Image



Sobel Filtered Image



**Take away:**
We learnt about various types of filters in MATLAB and how to apply it on images. We observed the results and understood the mechanisms.

**Course outcome:**

CO mapped – CO2

After performing this experiment, students will be able to apply appropriate filter to enhance image quality.

**Practical Session No.:** 7

**Date:** 27/02/2024

**Objective:** To perform histogram equalisation on the image.

**Methodology/Algorithm/Related Theory:**

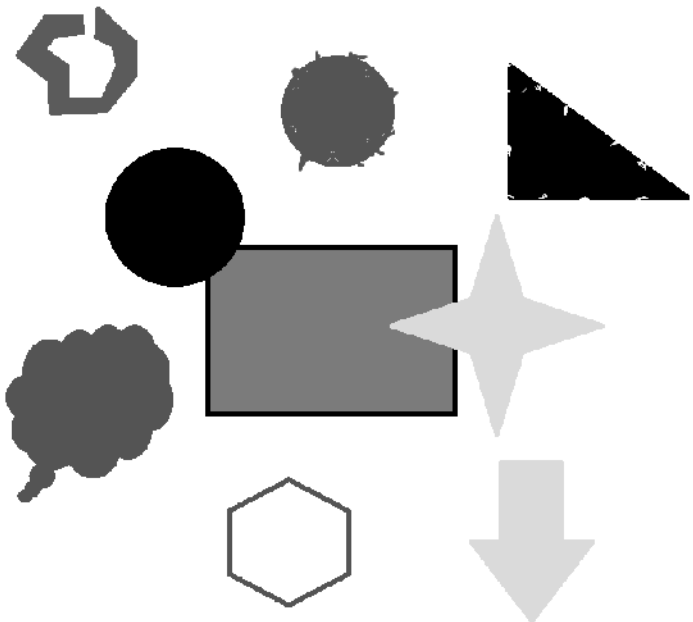
- Read the image using imread(). Convert the image into greyscale using rgb2gray() function.
- We use the imhist() function, this gives us the histogram of the input image.
- We use the histeq() function to equalize the histogram of the input image.

**Code:**

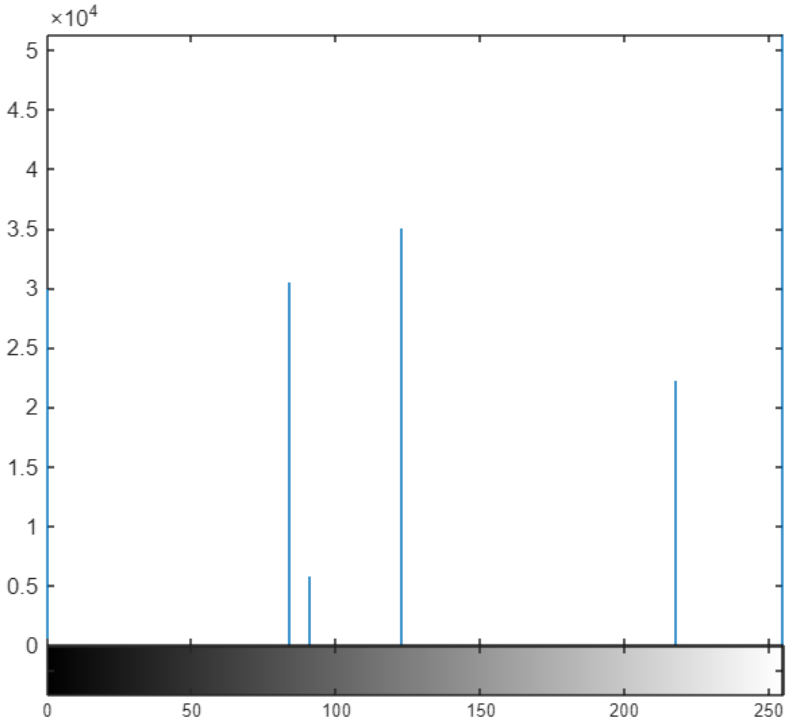
```
clear;
clc;
A=imread("imgshapes.bmp"); % Reading the image
A=rgb2gray(A);             % converting the image from RGB to Gray scale
imshow(A)                  % Displaying the Grayscale image

imhist(A)                  % Histogram of the original Grayscale image

h=histeq(A);               % Equalized histogram of the grayscale image
imhist(h)                  % Histogram of the equalized Grayscale image

imshowpair(A,h,"montage")  % Displaying the original and histogram equalised image
```

**Output:**

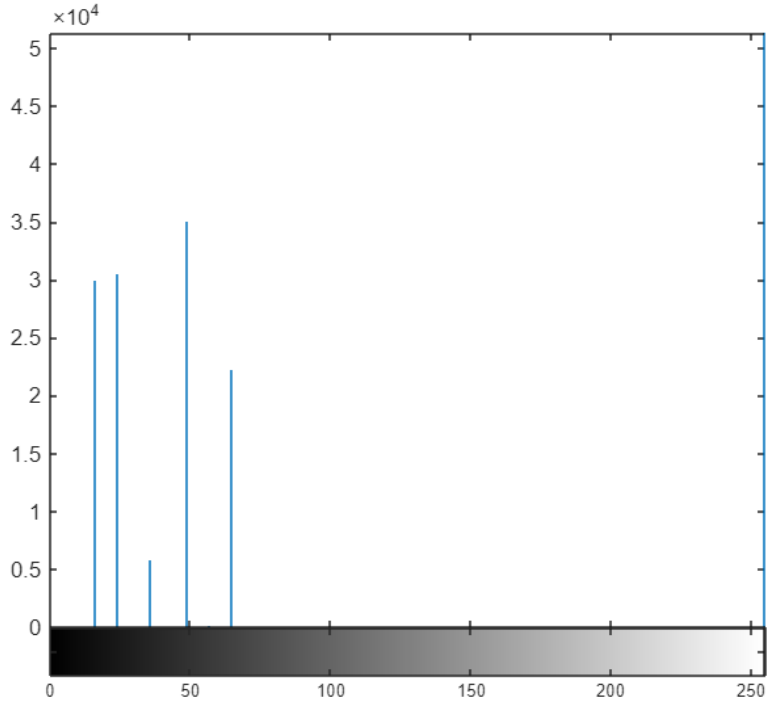
Original grayscale image



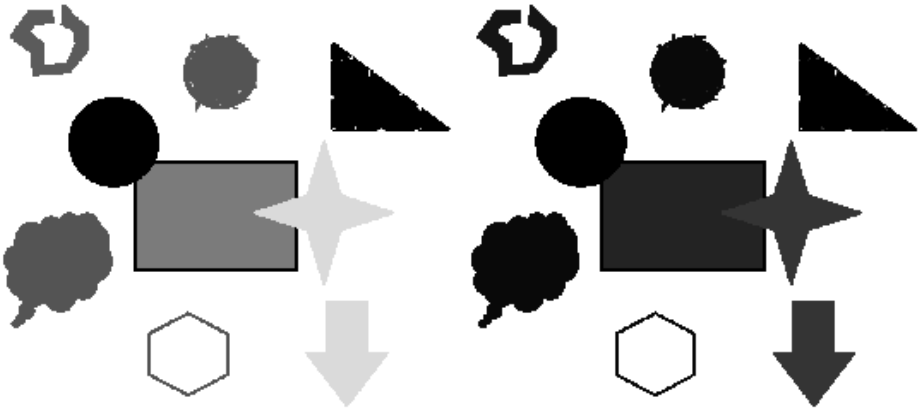
Original Histogram




Equalised Histogram



Original Image Histogram Equalised Image



**Take away:**

We got a deep knowledge of histogram equalization. The histogram equalisation helps in flattening the original histogram, somewhat equalizing to a particular section of intensity values. Doing it so helps us in further processing of image as it balances the intensity distribution of image.

**Course outcome:**

CO mapped – CO1

After performing this experiment, students will be able to apply appropriate point operations to enhance the image quality.

**Practical Session No.:** 8

**Date:** 12/03/2024

**Objective:** To remove the blurriness/fogginess of an image.

**Methodology/Algorithm/Related Theory:**

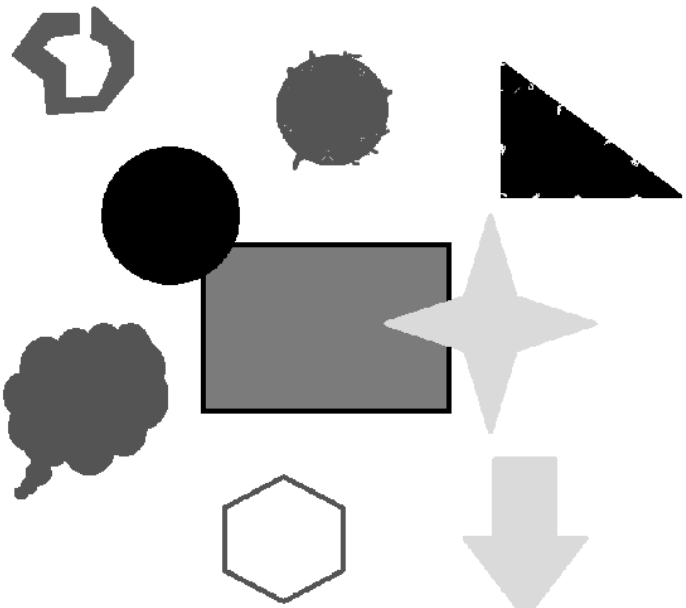
- Read the image using imread(). Convert the image into greyscale using the rgb2gray() function.
- For blurring the image, we use an averaging filter.
- We create a blurring mask of size 8x8 using the fspecial() function.
- For sharpening the image we use the imsharpen() function. It basically uses a gaussian filter as a sharpening filter.
- We need to specify the radius i.e. the standard deviation of a gaussian low pass filter and amount i.e. strength of the sharpening effect.

**Code:**

```
clear;
clc;
A=imread("imgshapes.bmp"); % Reading the image
A=rgb2gray(A);             % converting the image from RGB to Gray scale
imshow(A)                  % Displaying the Grayscale image

% Define the blur Mask (e.g., a 8x8 averaging filter)
blurMask = fspecial('average', [8, 8]);

% Apply the blur mask to the image
blurredImage = imfilter(A, blurMask, 'replicate');

% Displaying the blurred image
imshow(blurredImage)

% Apply Laplacian filter for sharpening specifying the required radius and
% amount
sharpenedImage = imsharpen(blurredImage,"Radius",5,"Amount",1.6);
% Displaying the blurred and sharpened image
imshowpair(blurredImage,sharpenedImage,"montage")
```

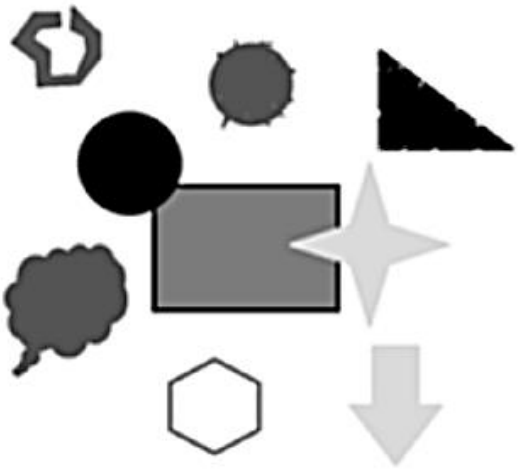
**Output:**

Grayscale Image



Blurred Image

Sharpened Image



**Take away:**

We got a deep knowledge of effect of blurring and sharpening. This logic can be used to enhance any blurred or noisy image, using the sharpening effect with its parameters. Doing it so helps us in further processing of image as it parts the way for pre-processing of noisy/blurry image.

**Course outcome:**

CO mapped – CO2

After performing this experiment, students will be able to apply appropriate filter to enhance image quality.

**Practical Session No.:** 9

**Date:** 19/03/2024

**Objective:** To perform morphological operations on the image.

**Methodology/Algorithm/Related Theory:**

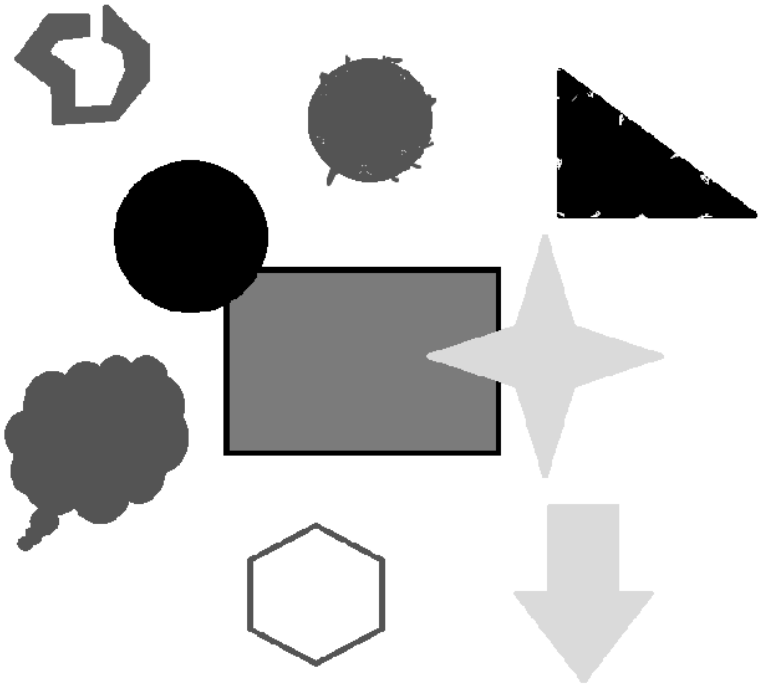
- Read the image using imread(). Convert the image into greyscale using rgb2gray() function.
- Create a structuring element using the "strel()" function. In our case, it is a square matrix of order 5x5.
- Use the imerode() and imdilate() functions to perform erosion and dilation respectively. The imopen() and imclose() functions are used for performing opening and closing morphological operations on the image respectively.
- Extract the inner boundary of the image by subtracting the original image from the dilated image. Extract the outer boundary of the image by subtracting the eroded image from the original image.

**Code:**

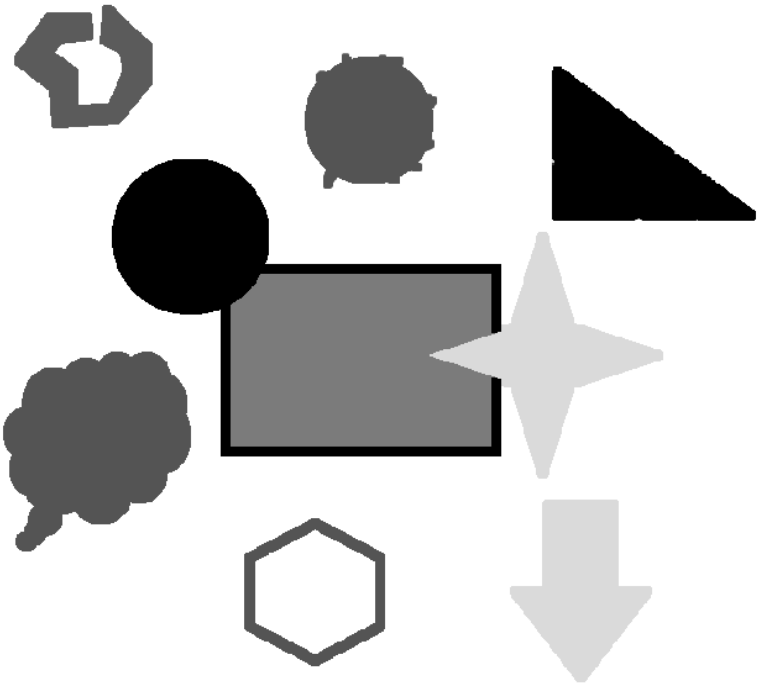
```
clear;
clc;
A=imread("imgshapes.bmp"); % Reading the image
A=rgb2gray(A);             % converting the image from RGB to Gray scale
imshow(A)                  % Displaying the Grayscale image
SE=strel('square',5);      % Creating a square structuring element of size 5x5

e=imerode(A,SE);           % Performing erosion on the image using the SE created
imshow(e)

d=imdilate(A,SE);          % Performing dilation on the image using the SE created
imshow(d)

o=imopen(A,SE);            % Performing opening operation on the image using the
SE created
imshow(o)

c=imclose(A,SE);           % Performing closing operation on the image using the
SE created
imshow(c)

InnerBoundary=d-A;         % Extracting Inner Boundary of image using Arithmetical
operation
imshow(InnerBoundary)

OuterBoundary=A-e;         % Extracting Outer Boundary of image using Arithmetical
operation
imshow(OuterBoundary)
```

**Output:**

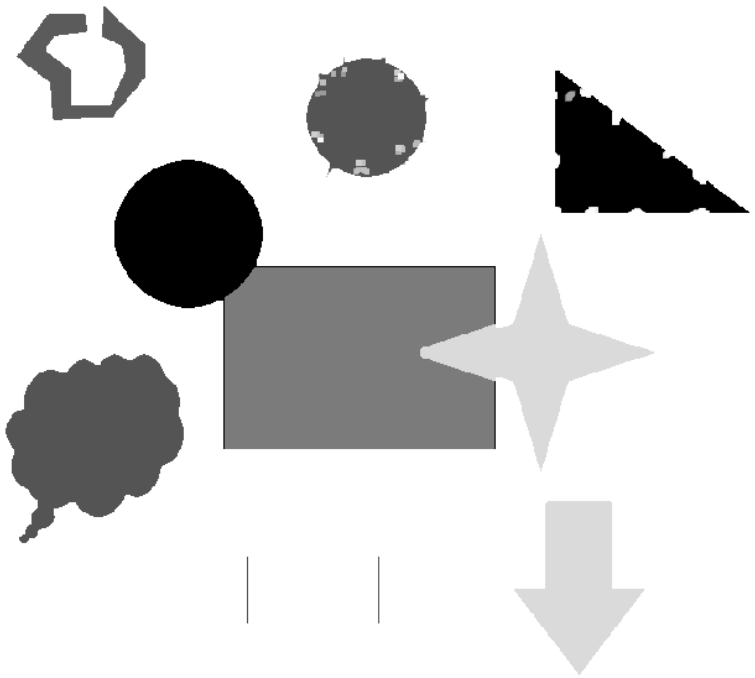
Original grayscale image



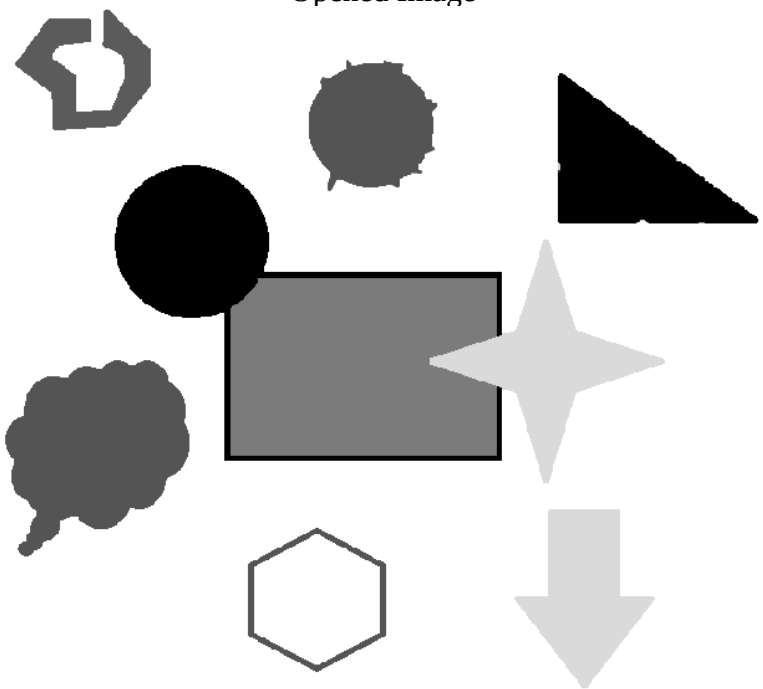
Eroded Image



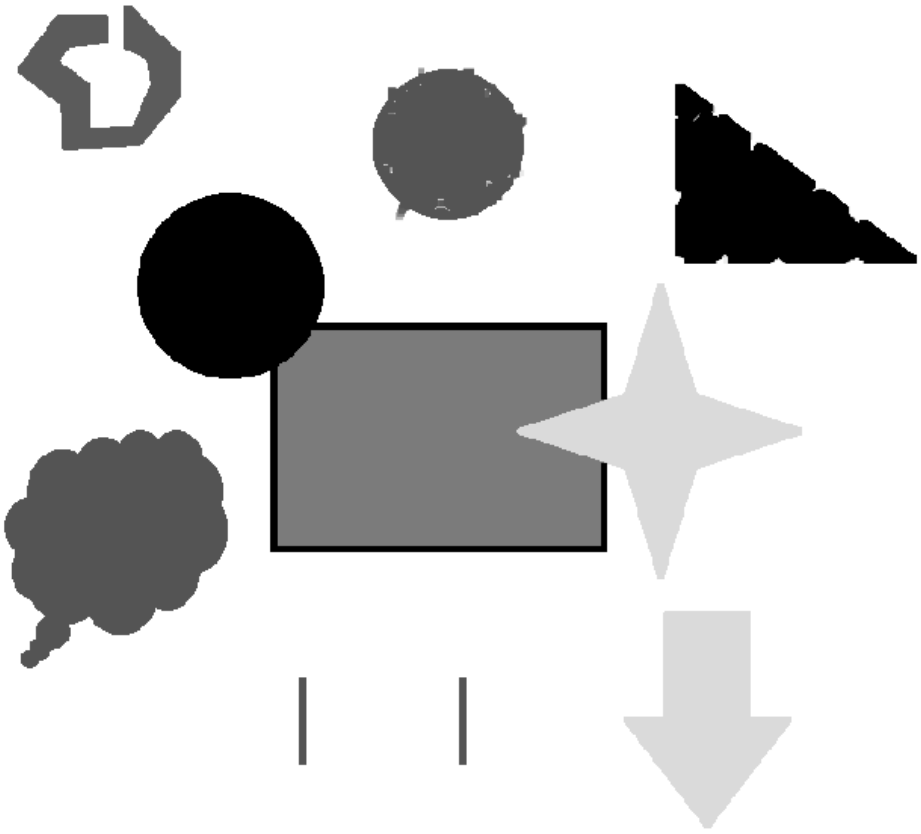
Dilated Image



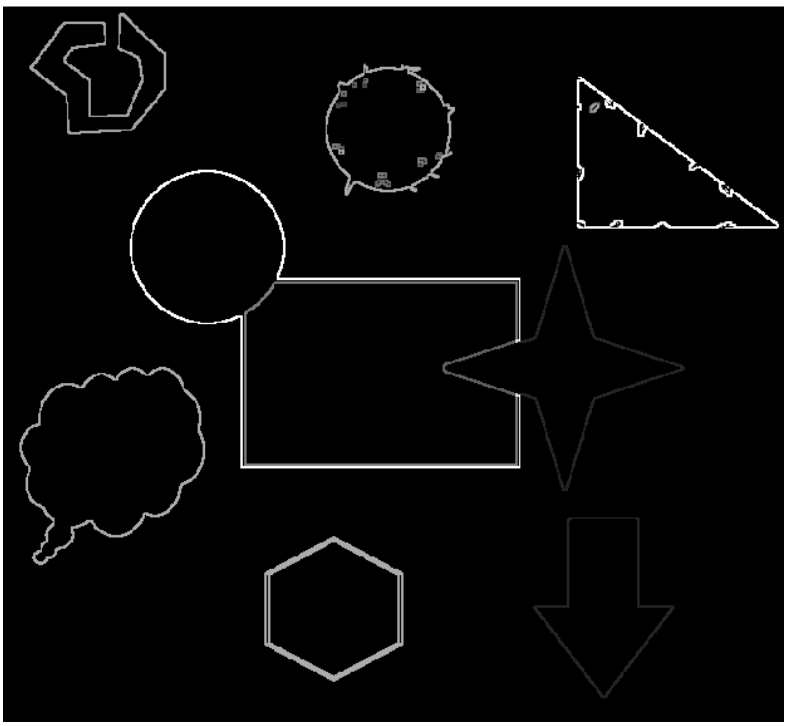
Opened Image



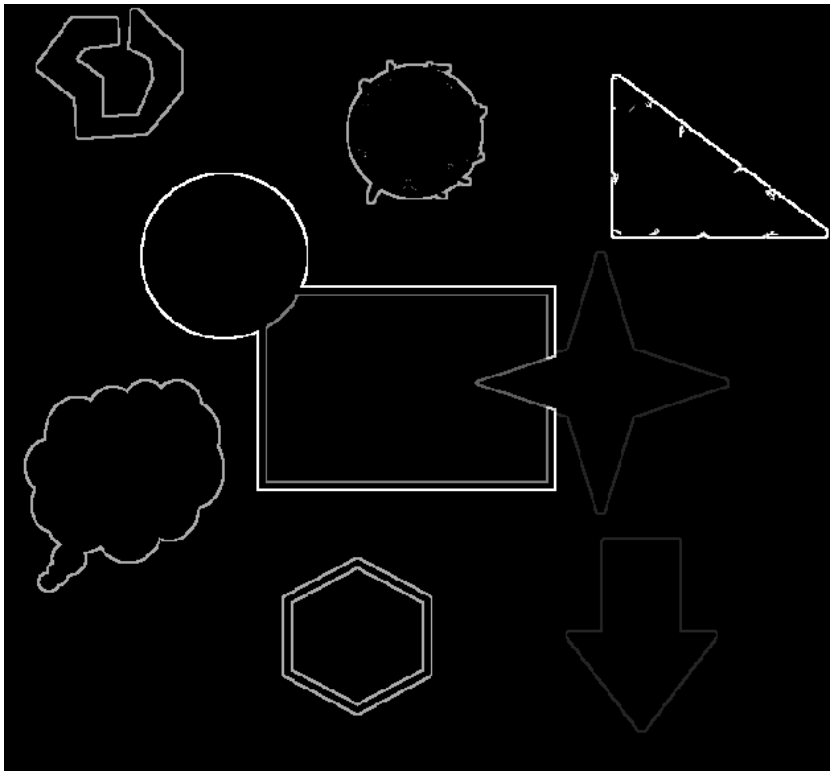
Closed Image



Inner Boundary Extracted Image



Outer Boundary Extracted Image



**Take away:**

We got a deep knowledge of erosion, dilation, opening and closing operations. This logic can be used to extract the boundaries of an image. Based on the user's requirements it may be the inner boundaries or the outer boundaries. This further helps in the processing of the images like shape recognition and much more.

**Course outcome:**

CO mapped – CO4

After performing this experiment, students will be able to apply proper morphological operations of image to extract shapes or features in an image.

**Practical Session No.:** 10

**Date:** 30/01/2024 - 26/03/2024

**Objective:** Mini-Project

**List of Papers:**

1. Y. Jusman, R. I. Tamarena, S. Puspita, E. Saleh and S. N. A. M. Kanafiah, "Analysis of Features Extraction Performance to Differentiate of Dental Caries Types Using Gray Level Co-occurrence Matrix Algorithm," 2020 10th IEEE International Conference on Control System, Computing and Engineering (ICCSCE), Penang, Malaysia, 2020, pp. 148-152, doi: 10.1109/ICCSCE50387.2020.9204937.

2. Fernando Roberti de Siqueira, William Robson Schwartz, Helio Pedrini, Multi-scale gray level co-occurrence matrices for texture description, Neurocomputing, Volume 120, 2013, Pages 336-345, ISSN 0925-2312, https://doi.org/10.1016/j.neucom.2012.09.042. (https://www.sciencedirect.com/science/article/pii/S0925231213003329)

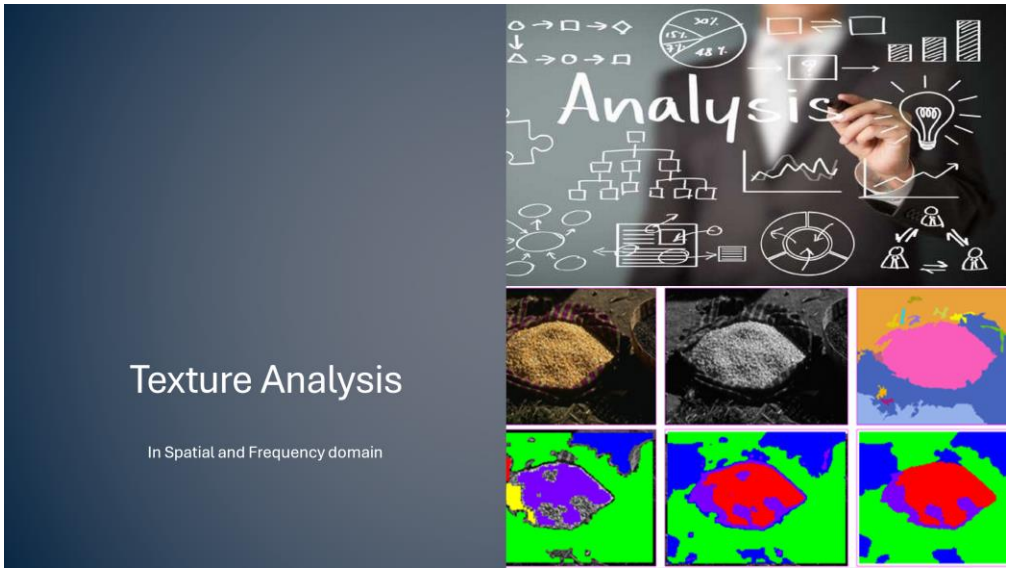
3. Aouat, S., Ait-hammi, I. & Hamouchene, I. A new approach for texture segmentation based on the Gray Level Co-occurrence Matrix. Multimed Tools Appl 80 https://doi.org/10.1016/j.ndteint.2004.03.004. https://www.sciencedirect.com/science/article/pii/S0963869504000258

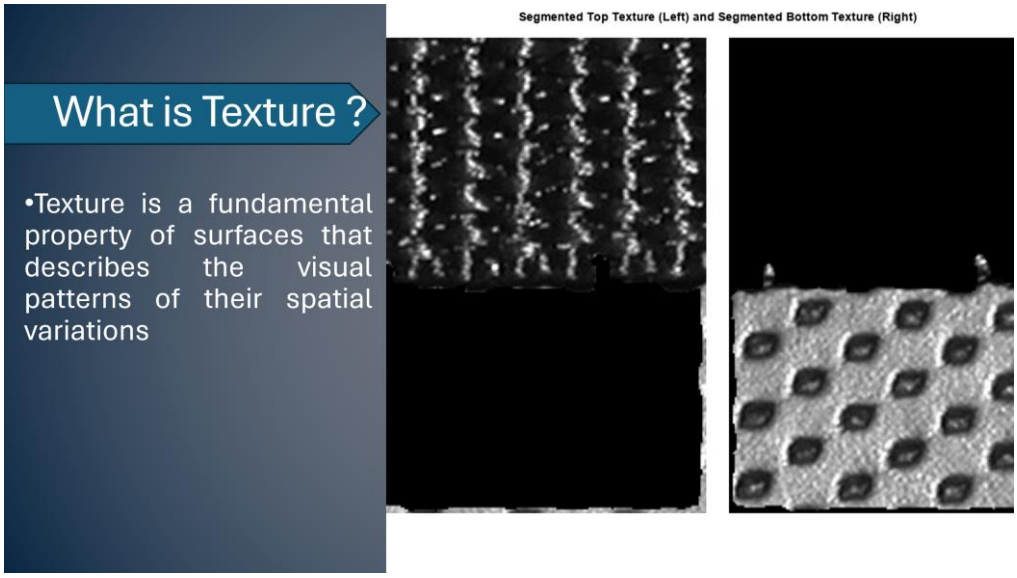
4. E.S. Gadelmawla, A vision system for surface roughness characterization using the gray level co-occurrence matrix, NDT & E International, Volume 37, Issue 7, 2004, Pages 577-588, ISSN0963-8695,https://doi.org/10.1016/j.ndteint.2004.03.004. (https://www.sciencedirect.com/science/article/pii/S0963869504000258)

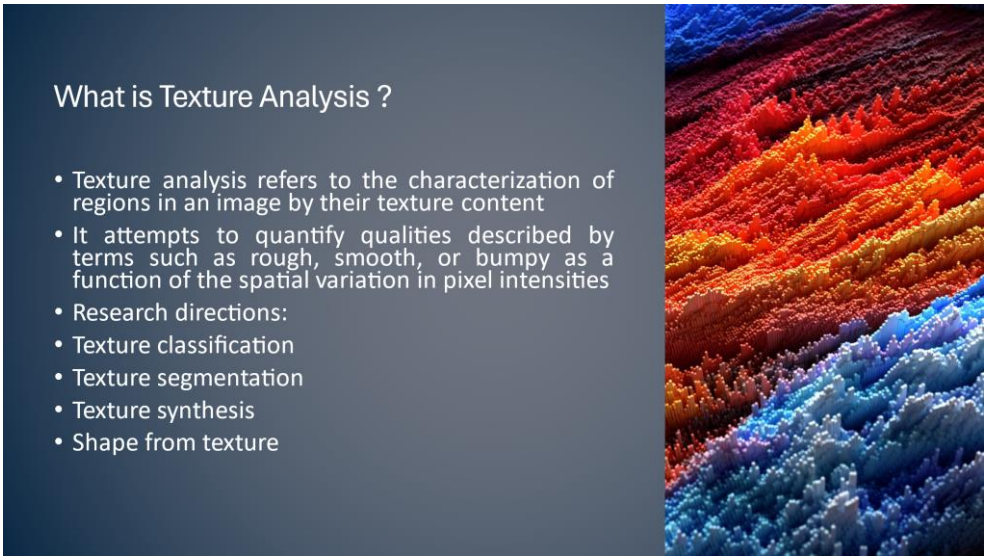
**White Paper:**

Drive Link: https://drive.google.com/drive/folders/1k0l-GwE7ozHTg7YiZnNsqI4JIr9LnbMI?usp=sharing

**Presentation:**

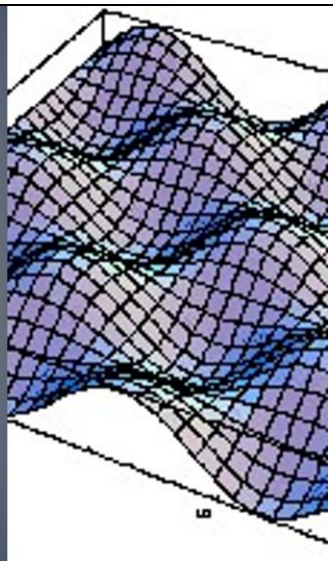


## Methods of Texture Analysis:

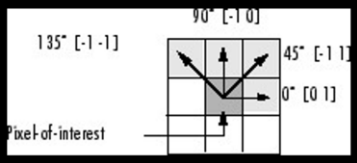
- **Statistical methods:** These methods represent textures using statistical features, such as the grey level co-occurrence matrix (GLCM)
- **Spectral methods:** These methods represent textures using their frequency domain representation



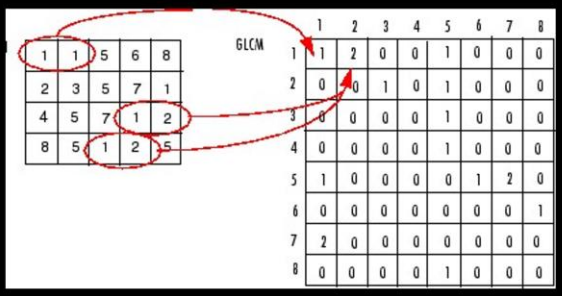
## Gray Level Co-occurrence Matrix (GLCM):

- It is a method used to describe the spatial distribution of gray levels in an image
- It calculates the frequency with which pairs of pixels with specific gray levels occur at a specified distance and orientation from each other
- Using different values of 'D' (the distance between the pixels being compared) and the normalized GLCM, we can calculate the following parameters:
- Contrast
- Correlation
- Energy
- Homogeneity

| Angle | Offset |
|---|---|
| 0 | [0 D] |
| 45 | [-D D] |
| 90 | [-D 0] |
| 135 | [-D -D] |




## GLCM Example:



## Results of Spatial Domain Analysis:

Reference Grayscale Image

Standard deviation filtered image

Calculated Parameters:

| Parameters | Offset 1 | Offset 2 | Offset 3 | Offset 4 |
|---|---|---|---|---|
| Contrast | 0.5232 | 0.8075 | 0.5038 | 0.7712 |
| Correlation | 0.8735 | 0.8048 | 0.8783 | 0.8136 |
| Energy | 0.1035 | 0.0848 | 0.1015 | 0.0861 |
| Homogeneity | 0.8288 | 0.7744 | 0.8238 | 0.7787 |