

Digital Image Processing(CSE-4007)

LAB REPORT



VIT-AP
UNIVERSITY

Written by: VALIVETI MANIKANTA BHUVANESH

19BCD7088

L49+L50 slot

Under the guidance of Prof LAKHAN DEV SHARMA

LAB-1

Objective of Experiment: To perform Intensity Transformations on cameraman image and lena image

Tools Required: MATLAB

Theory: Intensity Transformations is a process of mapping each intensity value of an input image into the corresponding output intensity value through mathematical expression. These are applied on images for contrast manipulation or image thresholding. These are in the spatial domain. Intensity transformations are among the simplest of all image processing techniques.

Code:

```
%Intensity transformation
clear all;
clc
img = imread("cameramgrey.jpg");
subplot(5, 2, 1),
imshow(img);
title("Original cameraman image");
img2=imread("C:\Users\Manikanta Bhuvanesh\Desktop\6th sem\DIP\lab\Lab1\lena.jpg");
subplot(5, 2, 2),
imshow(img2);
title("Original lena image");
L = 2 ^ 8;
neg = (L - 1) - img;
subplot(5, 2, 3),
imshow(neg);
title("Negative cameraman Image")
neg2 = (L - 1) - img2;
subplot(5, 2, 4),
imshow(neg2);
title("Negative lena Image")

c=1;
b=im2double(img);
s1=(c*log(1+b));
subplot(5, 2, 5),
imshow(s1);
title("logarithmic cameraman Image c=1")

b2=im2double(img2);
s2=(c*log(1+b2));
subplot(5, 2, 6),
imshow(s2);
title("logarithmic lena Image c=1")

r=0.5;
p1=(c*(b.^r));
subplot(5,2,7),
```

```

imshow(p1);
title("Gamma cameraman Image r=0.5")

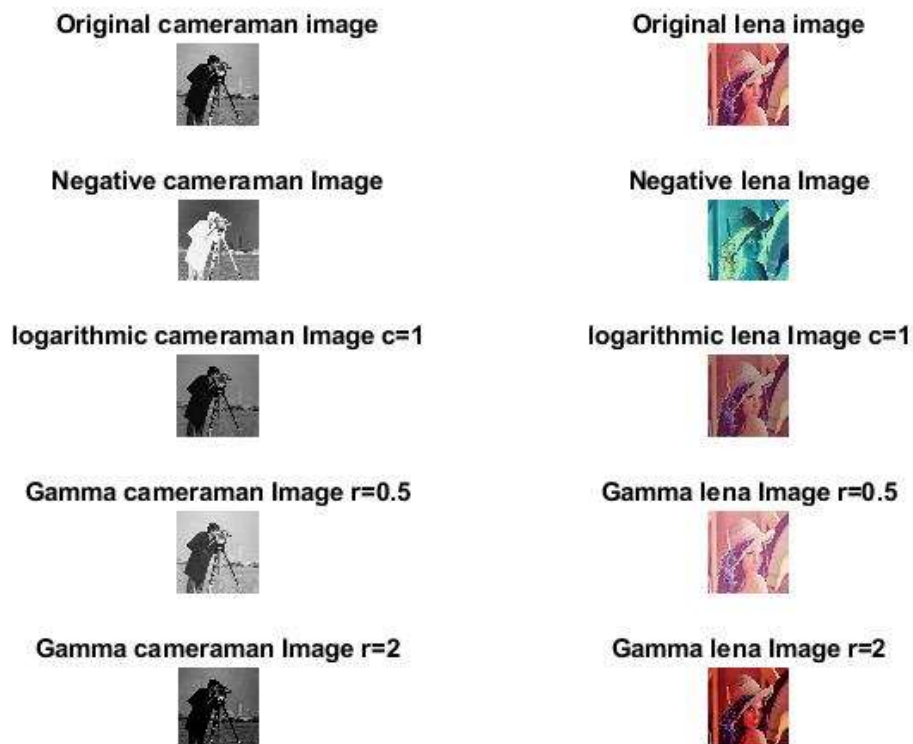
p2=(c*(b2.^r));
subplot(5,2,8),
imshow(p2);
title("Gamma lena Image r=0.5")

r=2;
p1=(c*(b.^r));
subplot(5,2,9),
imshow(p1);
title("Gamma cameraman Image r=2")

p2=(c*(b2.^r));
subplot(5,2,10),
imshow(p2);
title("Gamma lena Image r=2")

```

Results/Figure:



Conclusion: In this experiment we have seen logarithmic and gamma intensity transformation on cameraman image and lena image with different c and r values and comparison with original images

LAB-2

Objective of Experiment: To perform Spatial Filtering(Gaussian filter) and plot original Image, Noisy Image (Gaussian Noise), Filtered Image and Averaged Image

Tools Required: MATLAB

Theory:

- Spatial filtering is a process by which we can alter properties of an optical image by selectively removing certain spatial frequencies that make up an object. It is used directly on pixels of an image.
- A Gaussian Filter is a low pass filter. It replaces every element of the input signal with a weighted average of its neighborhood. This causes blurring regions of an image and is used for reducing noise (high frequency components).

Code:

```
clear all;
clc
I=imread('cameraman.tif');
subplot(2,2,1);
imshow(I);title('original image');
f=ones(5,5)/25;
g=[0.369 0.6065 0.369;0.6065 1 0.6065;0.3679 0.6065 0.3679]/4.8976;
J=imnoise(I,"gaussian",0,0.01);
subplot(2,2,2);
imshow(J);title('Noise image');
h=imfilter(J,g,'circular');
subplot(2,2,3);
imshow(h);title('Filtered image');
p=imfilter(I,g,'circular');
subplot(2,2,4);
imshow(p);title('Averaged image');
sgtitle('gaussian filtering');
```

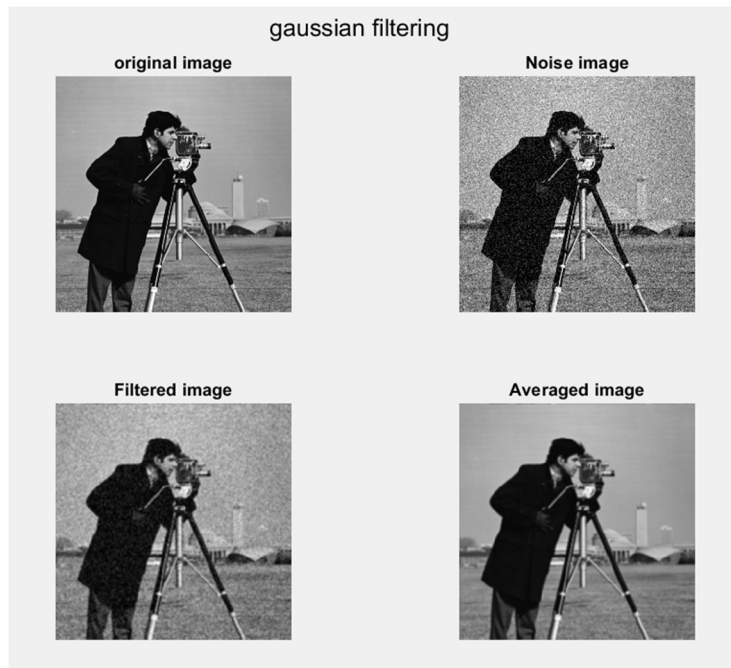
```
clear all;
clc
I=imread('cameraman.tif');
subplot(2,2,1);
imshow(I);title('original image');
f=ones(5,5)/25;
J=imnoise(I,"gaussian",0,0.01);
subplot(2,2,2);
imshow(J);title('Noise image');
h=imfilter(J,g,'circular');
subplot(2,2,3);
imshow(h);title('Filtered image');
p=imfilter(I,g,'circular');
subplot(2,2,4);
imshow(p);title('Averaged image');
sgtitle('5*5 spatial filtering');
```

```

clear all;
clc
I=imread('cameraman.tif');
subplot(2,2,1);
imshow(I);title('original image');
f=ones(3,3)/9;
J=imnoise(I,"gaussian",0,0.01);
subplot(2,2,2);
imshow(J);title('Noise image');
h=imfilter(J,g,'circular');
subplot(2,2,3);
imshow(h);title('Filtered image');
p=imfilter(I,g,'circular');
subplot(2,2,4);
imshow(p);title('Averaged image');
sgtitle('3*3 spatial filtering');

```

Results/ Figures:



5*5 spatial filtering

original image



Noise image



Filtered image



Averaged image



3*3 spatial filtering

original image



Noise image



Filtered image



Averaged image



Conclusion: In this experiment we have seen gaussian and spatial filtering on original cameraman image and noise image and comparison between original image ,noise image ,filtered image and averaged image using gaussian filter and spatial with 5*5 filter and 3*3 filter.

LAB-3

Objective of Experiment: To write a program to plot histogram (without using in-built function) of a given image and compare the results with the in-built function: histogram()

Tools Required: MATLAB

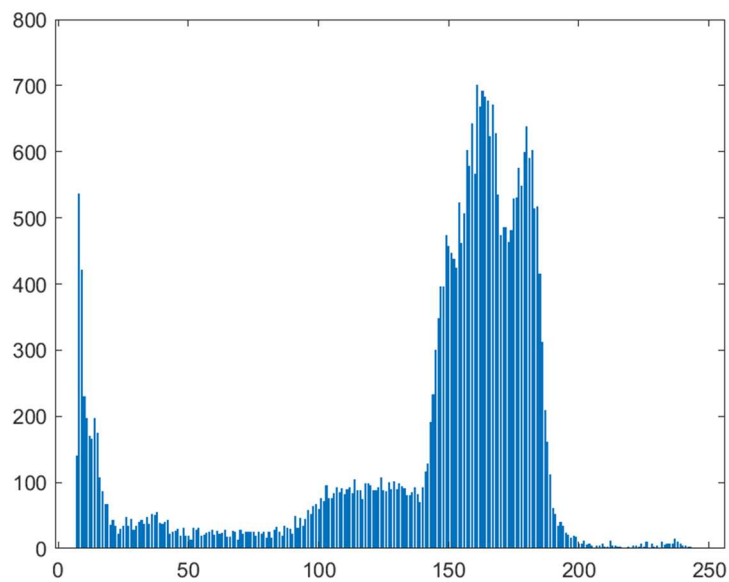
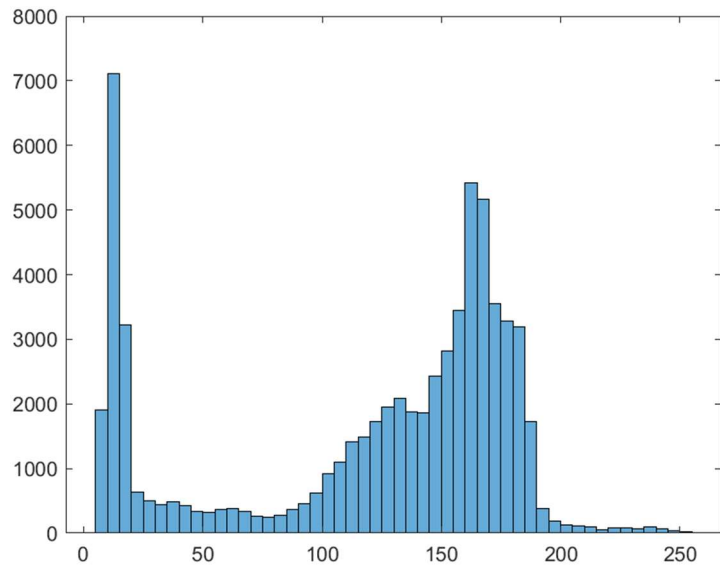
Theory: A histogram is a graphical representation that organises a group of data points into user-specified ranges. It is a popular graphing tool. It is used to summarize discrete or continuous data that are measured on an interval scale. It is often used to illustrate the major features of the distribution of the data in a convenient form.

Code:

```
clear; close all; clc;
I = imread('cameraman.tif');
y = zeros(256,1);
for i = 1:256
    for j = i + 1:256
        for x = 0:255
            if I(i,j) == x
                y(x, 1) = y(x, 1) + 1;
            end
        end
    end
end
x1 = 0:1:255;
bar(x1, y);
figure,title('Manual Histogram');

histogram(I)
figure,title("Built in Histogram");
```

Results/ Figures:



Conclusion: In this experiment we have seen how to plot histogram of image using inbuilt function and manual method.

LAB-4

Objective of Experiment: To write a program to perform Image sharpening using the Laplacian filters

Tools Required: MATLAB

Theory: Image sharpening encompasses any enhancement technique that highlights the edges and fine details of an image. Image sharpening is done by adding to the original image a signal proportional to a high-pass filtered version of the image.

The Laplacian filter is an edge-sharpening filter, which sharpens the edges of the image. The input gray image is first subjected to a Laplacian filter, which acts as the pre-processing block and then Adaptive Histogram Equalization (AHE) is applied to the image obtained after pre-processing

Code:

```
clear all;
clc
I = imread('cameraman.tif');
subplot(1, 5, 1),
imshow(I);
title("Original cameraman image");
Lap=[0 1 0; 1 -4 1; 0 1 0];
[p1,output1]=Func134(I,Lap);
subplot(1, 5, 2),
imshow(output1);
title("Laplace filter sharpen (a)");
Lap1=[0 -1 0; -1 4 -1; 0 -1 0];
[p2,output2]=Func134(I,Lap1);
subplot(1, 5, 3),
imshow(output2);
title("Laplace filter sharpen (b)");
Lap2=[1 1 1;1 -8 1; 1 1 1];
[p3,output3]=Func134(I,Lap2);
subplot(1, 5, 4),
imshow(output3);
title("Laplace filter sharpen(c)");
Lap3=[-1 -1 -1;-1 8 -1; -1 -1 -1];
[p4,output4]=Func134(I,Lap3);
subplot(1, 5, 5),
imshow(output4);
title("Laplace filter sharpen (d)");
function [ p, image ] = Func134( img,Lap )
    I=zeros(size(img));
    A=padarray(img,[1,1]);
    A=double(A);
    for i=1:size(A,1)-2
        for j=1:size(A,2)-2
```

```

        I(i,j)=sum(sum(Lap.*A(i:i+2,j:j+2)));
    end
end
p=uint8(I);
image=img-p;
end

```

Results/ Figures:



Conclusion: In this experiment we have seen four different Laplace filters to sharpen the image and their comparison with original image

LAB-5

Objective of Experiment: To study noise of an image

- a) Add additive white gaussian noise to the given image and show the effect of mean and standard deviation. (use `imnoise()`)
- b) Add salt and pepper noise to the given image and show the effect of noise density. (use `imnoise()`)
- c) Add salt noise without using `imnoise()` function to a given Image.
- d) Add pepper noise without using `imnoise()` function to a given Image.

Tools Required: MATLAB

Theory: Image noise is random variation of brightness or color information in images, and is usually an aspect of electronic noise. There are three types of impulse noises. Salt Noise, Pepper Noise, Salt and Pepper Noise.

Salt Noise: Salt noise is added to an image by addition of random bright (with 255 pixel value) all over the image.

Pepper Noise: Salt noise is added to an image by addition of random dark (with 0 pixel value) all over the image.

Code:

```
clear all;
clc
I = imread('cameraman.tif');
subplot(3, 3, 2),
imshow(I);
title("Original cameraman image");
J = imnoise(I,'gaussian',0);
subplot(3, 3, 4),
imshow(J);
title("Gaussian mean 0 and variance 0.01");
J1 = imnoise(I,'gaussian',1);
subplot(3, 3, 5),
imshow(J1);
title("Gaussian mean 1 and variance 0.01");
J2 = imnoise(I,'gaussian',215);
subplot(3, 3, 6),
imshow(J2);
title("Gaussian mean 215 and variance 0.01");
J3 = imnoise(I,'gaussian',0,0.03);
subplot(3, 3, 7),
imshow(J3);
title("Gaussian mean 0 and variance 0.03");
J4 = imnoise(I,'gaussian',0,0.04);
subplot(3, 3, 8),
imshow(J4);
title("Gaussian mean 0 and variance 0.04");
```

```
J5 = imnoise(I,'gaussian',0,0.05);
subplot(3, 3, 9),
imshow(J5);
title("Gaussian mean 0 and variance 0.05");
```

```
clear all;
clc
I = imread('cameraman.tif');
subplot(2, 3, 2),
imshow(I);
title("Original cameraman image");
J = imnoise(I,'salt & pepper',0.02);
subplot(2, 3, 4),
imshow(J);
title("noise density 0.02");
J1 = imnoise(I,'salt & pepper',0.03);
subplot(2, 3, 5),
imshow(J1);
title("noise density 0.03");
J2 = imnoise(I,'salt & pepper',0.04);
subplot(2, 3, 6),
imshow(J2);
title("noise density 0.04");
```

```
clear all;
clc
I=imread('cameraman.tif');
subplot(2,3,2);
imshow(I);
title('Original Image');
b=20;
img_with_noise= im2double(I);
[row,col]=size(I);
x = randi([0,255],row,col);
img_with_noise(x <= b) = 0;
subplot(2,3,4);
imshow(img_with_noise);
title('pepper noise');
```

```
I=imread('cameraman.tif');
b=210;
img_with_noise= im2double(I);
[row,col]=size(I);
x = randi([0,255],row,col);
img_with_noise(x >= b) = 255;
subplot(2,3,6);
imshow(img_with_noise);
title('salt noise');
```

Results/ Figures:

Original cameraman image



Gaussian mean 0 and variance 0.01



Gaussian mean 1 and variance 0.01



Gaussian mean 215 and variance 0.01



Gaussian mean 0 and variance 0.03



Gaussian mean 0 and variance 0.04



Gaussian mean 0 and variance 0.05



Original cameraman image



noise density 0.02



noise density 0.03



noise density 0.04



Original Image



pepper noise



salt noise



Conclusion: In this experiment we have seen gaussian noise with constant mean and constant variance and comparison of all noisy images with original image. Added salt and pepper noise with different density and compared with original image. And we have seen how to add salt and pepper noise manually instead of using inbuilt function and comparison with original image.

LAB-6

Objective of Experiment: To add Gaussian Noise (mean=0.5, std= 0.01) and restore the image.

- a) Restore using Arithmetic Mean Filter
- b) Restore using Geometric Mean Filter

Tools Required: MATLAB

Theory: Gaussian noise has a uniform distribution throughout the signal. Gaussian Noise is a statistical noise having a probability density function equal to normal distribution, also known as Gaussian Distribution. Random Gaussian function is added to Image function to generate this noise. It is also called as electronic noise because it arises in amplifiers or detectors.

The pdf of a Gaussian random variable z is given by:

$$p(z) = \frac{1}{\sqrt{2\pi}\sigma} e^{-(z-\mu)^2/2\sigma^2},$$

where z represents (noise) gray value, μ is the mean, and σ is its standard deviation. The squared standard deviation σ^2 is usually referred to as variance.

Code:

```
clear all
clc
I=imread('cameraman.tif');
I1=I;
I2=I;
subplot(4, 1, 1),
imshow(I);
title("Original cameraman image");
v=0.01^2;
J=imnoise(I,'gaussian',0.5,v);
subplot(4, 1, 2),
imshow(J);
title("Original Noise image");

[row, col]=size(I);
for i = 2:row-1
    for j = 2:col - 1
        a = 0;
        p=0;
        for u = -1:1
            for v = -1:1
                a = a + J(i + u, j + v);
                p = p * J(i + u, j + v);
```

```

        end
    end
    I1(i, j) = a / 9;
    I2(i,j)=p^(1/9);
end
end
subplot(4, 1, 3),
imshow(I1);
title("Restored cameraman image with arthematic mean");
subplot(4, 1, 4),
imshow(I2);
title("Restored cameraman image with geometic mean");

```

Results/ Figures:

Original cameraman image



Original Noise image



Restored cameraman image with arthematic mean



Restored cameraman image with geometic mean



Conclusion: In this experiment we have seen restoring of gaussian noise image using arithmetic mean filter and geometric mean filter and compared with original image of cameraman.

LAB-7

Objective of Experiment: To add Salt noise to image, restore using Contraharmonic Mean Filter and to show effect of 3 different possible values of Q for salt noise. (by taking Necessary Assumption).

Tools Required: MATLAB

Theory: With a contraharmonic mean filter, the color value of each pixel is replaced with the contraharmonic mean of color values of the pixels in a surrounding region. It reduces or virtually eliminates the effects of salt-and-pepper noise.

The contraharmonic mean filter is given by the expression:

$$\hat{f}(m,n) = \frac{\sum_{s,t} g(s,t)^{Q+1}}{\sum_{s,t} g(s,t)^Q}$$

where Q is called order of the filter.

- This yields the arithmetic mean filter for Q=0 and the harmonic mean filter for Q=-1. For positive values of Q, it reduces pepper noise and for negative values of Q, it reduces salt noise. It cannot do both simultaneously

Code:

```
clear all
clc
I=imread('cameraman.tif');
subplot(5, 1, 1),
imshow(I);
title("Original cameraman image");
J = imnoise(I,'salt & pepper');
subplot(5, 1, 2),
imshow(J);
title("Original Noise image");

% Restore using contraharmonic
I1=im2double(I);
masksize=3;
Q=-1;
sumn=[];
sumd=[];
pixln=0;
pixld=0;
[row,col]=size(I1);
for i=1:row;
    for j=1:col;
        for m=-masksize:masksize;
```

```

        for n=-masksize:masksize;
            if (i+m>0 && i+m<row && j+n>0 && j+n<col && masksize+m>0 &&
masksize+m<row && masksize+n>0 && masksize+n<col)
                pixl1=(I1(i+m,j+n)).^(Q+1);
                pixl2= (I1(i+m,j+n)).^Q;
                pixln=pixln+pixl1;
                pixld=pixld+pixl2;
            end
        end
        end
        res(i,j)=(pixln/pixld);
        pixln=0;
        pixld=0;
    end
end
subplot(5,1,3),
imshow(res);
title("Filter with Q=-1")

I1=im2double(I);
masksize=3;
Q=0;
sumn=[];
sumd=[];
pixln=0;
pixld=0;
[row,col]=size(I1);
for i=1:row;
    for j=1:col;
        for m=-masksize:masksize;
            for n=-masksize:masksize;
                if (i+m>0 && i+m<row && j+n>0 && j+n<col && masksize+m>0 &&
masksize+m<row && masksize+n>0 && masksize+n<col)
                    pixl1=(I1(i+m,j+n)).^(Q+1);
                    pixl2= (I1(i+m,j+n)).^Q;
                    pixln=pixln+pixl1;
                    pixld=pixld+pixl2;
                end
            end
        end
        res(i,j)=(pixln/pixld);
        pixln=0;
        pixld=0;
    end
end
subplot(5,1,4),
imshow(res);
title("Filter with Q=0")

I1=im2double(I);
masksize=3;
Q=1;
sumn=[];
sumd=[];
pixln=0;
pixld=0;
[row,col]=size(I1);
for i=1:row;

```

```

    for j=1:col;
        for m=-masksize:masksize;
            for n=-masksize:masksize;
                if (i+m>0 && i+m<row && j+n>0 && j+n<col && masksize+m>0 &&
masksize+m<row && masksize+n>0 && masksize+n<col)
                    pixl1=(I1(i+m,j+n)).^(Q+1);
                    pixl2= (I1(i+m,j+n)).^Q;
                    pixln=pixln+pixl1;
                    pixld=pixld+pixl2;
                end
            end
        end
        res(i,j)=(pixln/pixld);
        pixln=0;
        pixld=0;
    end
end
subplot(5,1,5),
imshow(res);

title("Filter with Q=1")

```

Results/ Figures:

Original cameraman image



Original Noise image



Filter with Q=-1



Filter with Q=0



Filter with Q=1



Conclusion: In this experiment we have seen restoring of salt and pepper noise image with contrahormanic filter with different Q values of -1,0,1. And comparison between all filtered images with original image.

LAB-8

Objective of Experiment: To perform edge detection of cameraman image by comparing 'Sobel' | 'Prewitt' | 'Roberts' | 'log' | 'zerocross' | 'Canny' | 'approxcanny' Methods

Tools Required: MATLAB

Theory: Edge detection is an image processing technique for finding the boundaries of objects within images. It works by detecting discontinuities in brightness. Edge detection is used for image segmentation and data extraction in areas such as image processing, computer vision, and machine vision.

Code:

```
clear all
clc
I=imread('cameraman.tif');
subplot(8, 1, 1),
imshow(I);
title("Original cameraman image");
BW1 = edge(I,'Sobel');
subplot(8, 1, 2),
imshow(BW1);
title("Sobel");
BW2 = edge(I,'canny');
subplot(8, 1, 3),
imshow(BW2);
title("Canny");
BW3 = edge(I,'prewitt');
subplot(8, 1, 4),
imshow(BW3);
title("Prewitt");
BW4 = edge(I,'roberts');
subplot(8, 1, 5),
imshow(BW4);
title("Roberts");
BW5 = edge(I,'log');
subplot(8, 1, 6),
imshow(BW5);
title("Log");
BW6 = edge(I,'zerocross');
subplot(8, 1, 7),
imshow(BW6);
title("Zerocross");
BW7 = edge(I,'approxcanny');
subplot(8, 1, 8),
imshow(BW7);
title("Approxcanny");
sgtitle('Edge detection using different methods');
```

Results/ Figures:

Edge detection using different methods

Original cameraman image



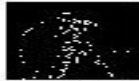
Sobel



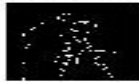
Canny



Prewitt



Roberts



Log



Zerocross



Approxcanny



Conclusion: In this experiment we have seen different edge detection methods Sobel, Prewitt, Roberts, log, zerocross, Canny and approxcanny. And compared all the methods with original image.

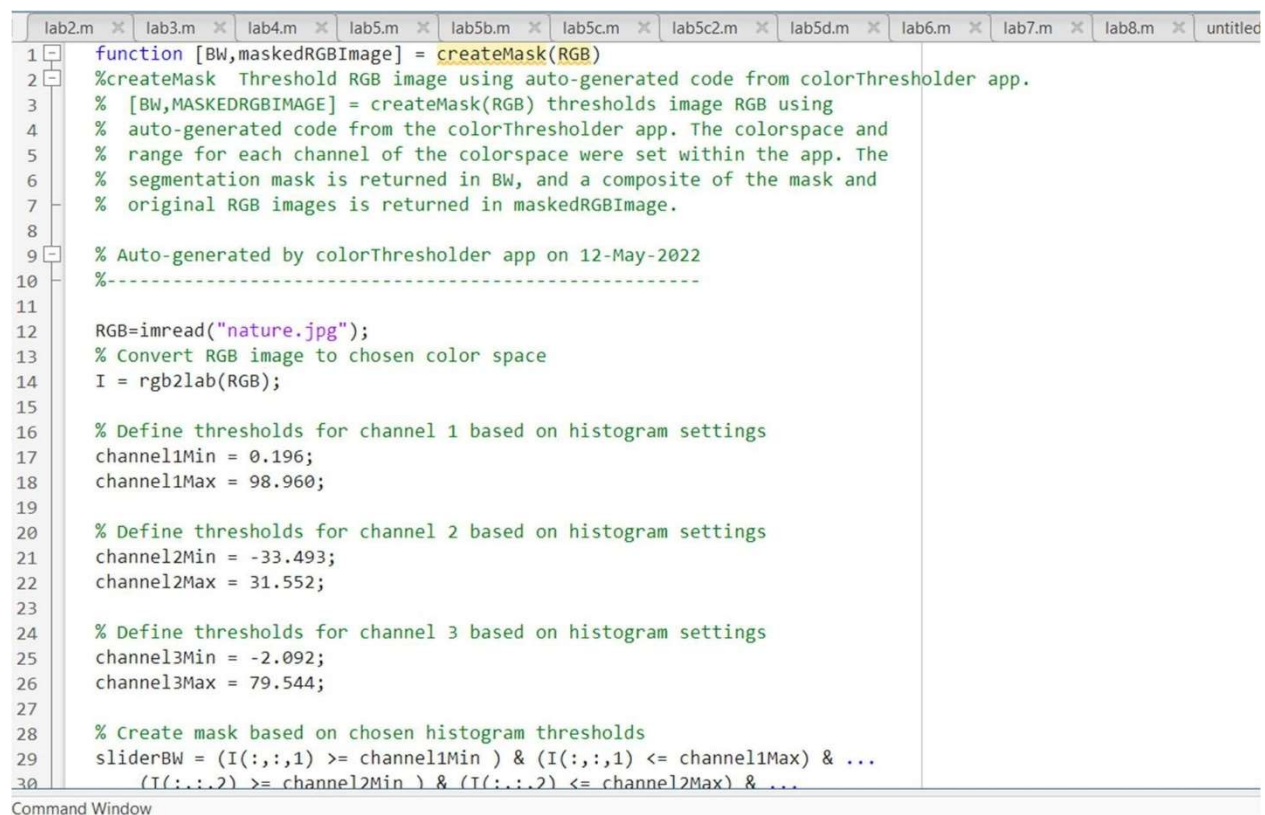
LAB-9

Objective of Experiment: To perform Image Segmentation using thresholding. To Segment Image and Create Mask Using Color Thresholder App. Study Color Thresholder App

Tools Required: MATLAB,COLOR THRESHOLDER APP

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. Thresholding is a type of image segmentation, where we change the pixels of an image to make the image easier to analyze. In thresholding, we convert an image from colour or grayscale into a binary image, i.e., one that is simply black and white. we use thresholding as a way to select areas of interest of an image, while ignoring the parts we are not concerned with.

Code:



```
lab2.m x lab3.m x lab4.m x lab5.m x lab5b.m x lab5c.m x lab5c2.m x lab5d.m x lab6.m x lab7.m x lab8.m x untitled
1 function [BW,maskedRGBImage] = createMask(RGB)
2 %createMask Threshold RGB image using auto-generated code from colorThresholder app.
3 % [BW,MASKEDRGBIMAGE] = createMask(RGB) thresholds image RGB using
4 % auto-generated code from the colorThresholder app. The colorspace and
5 % range for each channel of the colorspace were set within the app. The
6 % segmentation mask is returned in BW, and a composite of the mask and
7 % original RGB images is returned in maskedRGBImage.
8
9 % Auto-generated by colorThresholder app on 12-May-2022
10 %-----
11
12 RGB=imread("nature.jpg");
13 % Convert RGB image to chosen color space
14 I = rgb2lab(RGB);
15
16 % Define thresholds for channel 1 based on histogram settings
17 channel1Min = 0.196;
18 channel1Max = 98.960;
19
20 % Define thresholds for channel 2 based on histogram settings
21 channel2Min = -33.493;
22 channel2Max = 31.552;
23
24 % Define thresholds for channel 3 based on histogram settings
25 channel3Min = -2.092;
26 channel3Max = 79.544;
27
28 % Create mask based on chosen histogram thresholds
29 sliderBW = (I(:, :, 1) >= channel1Min ) & (I(:, :, 1) <= channel1Max) & ...
30 (I(:, :, 2) >= channel2Min ) & (I(:, :, 2) <= channel2Max) & ...
```

Command Window

```

31 (I(:,:,3) >= channelismin) & (I(:,:,3) <= channelismax);
32
33 % Create mask based on selected regions of interest on point cloud projection
34 I = double(I);
35 [m,n,~] = size(I);
36 polyBW = false([m,n]);
37 I = reshape(I,[m*n 3]);
38 temp = I(:,1);
39 I(:,1) = I(:,2);
40 I(:,2) = I(:,3);
41 I(:,3) = temp;
42 clear temp
43
44 % Project 3D data into 2D projected view from current camera view point within app
45 J = rotateColorSpace(I);
46
47 % Apply polygons drawn on point cloud in app
48 polyBW = applyPolygons(J,polyBW);
49
50 % Combine both masks
51 BW = sliderBW & polyBW;
52
53 % Invert mask
54 BW = ~BW;
55
56 % Initialize output masked image based on input image.
57 maskedRGBImage = RGB;
58
59 % Set background pixels where BW is false to zero.
60 maskedRGBImage(repmat(~BW,[1 1 3])) = 0;

```

```

62 end
63
64 function J = rotateColorSpace(I)
65
66 % Translate the data to the mean of the current image within app
67 shiftVec = [7.467597 43.019465 54.130602];
68 I = I - shiftVec;
69 I = [I ones(size(I,1),1)]';
70
71 % Apply transformation matrix
72 tMat = [0.003618 -0.010458 -0.000000 0.307123;
73         0.008955 0.003159 0.004643 -0.801875;
74         0.005200 0.001835 -0.007997 8.772824;
75         0.000000 0.000000 0.000000 1.000000];
76
77 J = (tMat*I)';
78 end
79
80 function polyBW = applyPolygons(J,polyBW)
81
82 % Define each manually generated ROI
83 hPoints(1).data = [0.067122 -0.530150;
84                   0.592253 -0.598556;
85                   0.614817 -1.343426;
86                   0.071225 -0.568153];
87
88 % Iteratively apply each ROI
89 for ii = 1:length(hPoints)
90     if size(hPoints(ii).data,1) > 2
91         J = (tMat*I)';
92         end
93     function polyBW = applyPolygons(J,polyBW)
94
95     % Define each manually generated ROI
96     hPoints(1).data = [0.067122 -0.530150;
97                       0.592253 -0.598556;
98                       0.614817 -1.343426;
99                       0.071225 -0.568153];
100
101     % Iteratively apply each ROI
102     for ii = 1:length(hPoints)
103         if size(hPoints(ii).data,1) > 2
104             in = inpolygon(J(:,1),J(:,2),hPoints(ii).data(:,1),hPoints(ii).data(:,2));
105             in = reshape(in,size(polyBW));
106             polyBW = polyBW | in;
107         end
108     end
109 end
110 end

```



```

Editor - D:\6th sem\Digital Image Processing\lab\bhuvcreateMask.m
lab2.m x lab3.m x lab4.m x lab5.m x lab5b.m x lab5c.m x lab5c2.m x lab5d.m x lab6.m x lab7.m x lab8.m x untitled8.m x vm.m x vysh.m x bhuv.m x bhuvcreateMask.m
1 function [BW,maskedRGBImage] = createMask(RGB)
2 %createMask Threshold RGB image using auto-generated code from colorThresholder app.
3 % [BW,MASKEDRGBIMAGE] = createMask(RGB) thresholds image RGB using
4 % auto-generated code from the colorThresholder app. The colorspace and
5 % range for each channel of the colorspace were set within the app. The
6 % segmentation mask is returned in BW, and a composite of the mask and
7 % original RGB images is returned in maskedRGBImage.
8
9 % Auto-generated by colorThresholder app on 12-May-2022
10 %-----
11
12 RGB=imread("nature.jpg");
13 % Convert RGB image to chosen color space
14 I = rgb2lab(RGB);
15
16 % Define thresholds for channel 1 based on histogram settings
17 channel1Min = 0.196;
18 channel1Max = 98.960;
19
Command Window
0 0 0 0 1 0 0
1 0 0 0 0 0 0
1 0 0 1 0 0 0
1 0 0 1 1 0 0
1 0 0 1 0 0 0
0 0 0 1 0 0 0
1 0 0 0 0 0 0
1 0 0 0 0 0 0
0 0 0 1 0 0 0
0 0 0 1 0 0 0
1 0 0 1 0 0 1

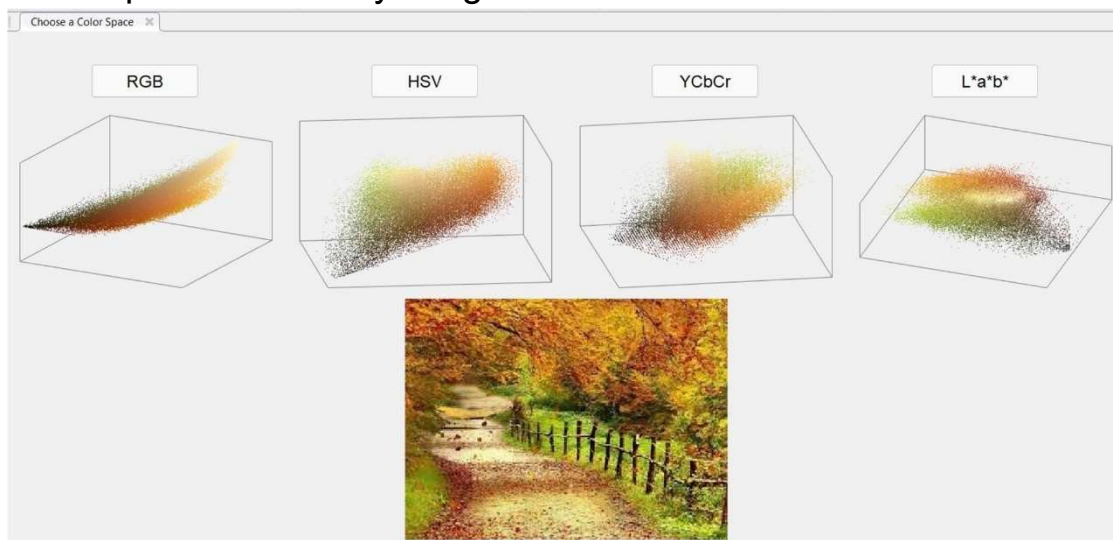
```

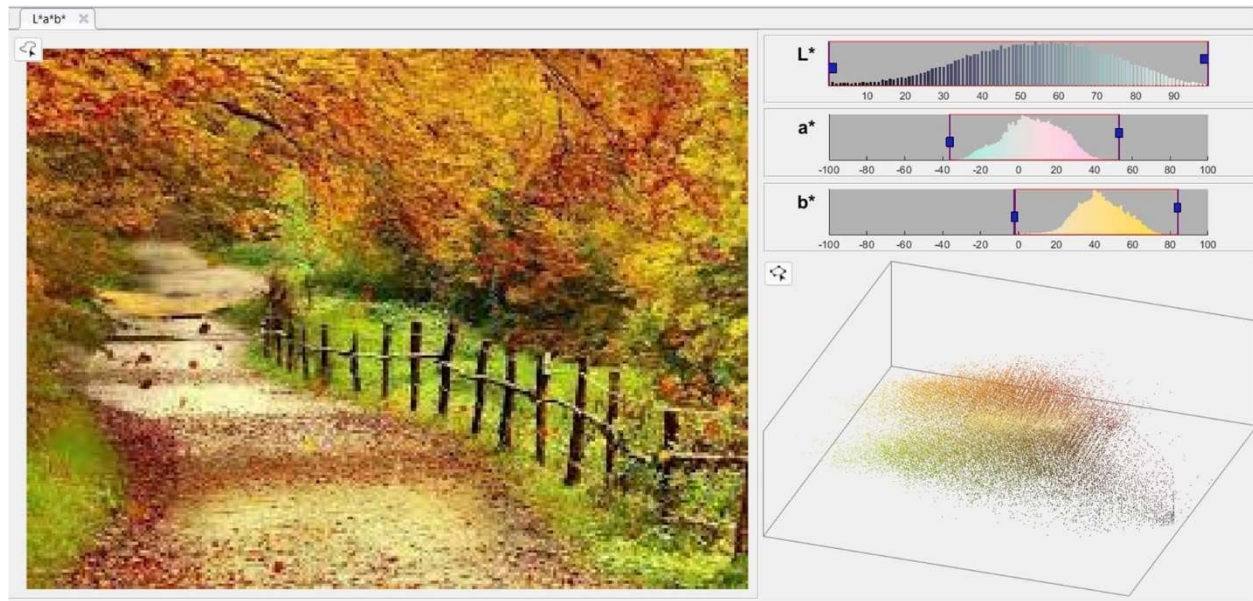
Results/ Figures:

Original image:

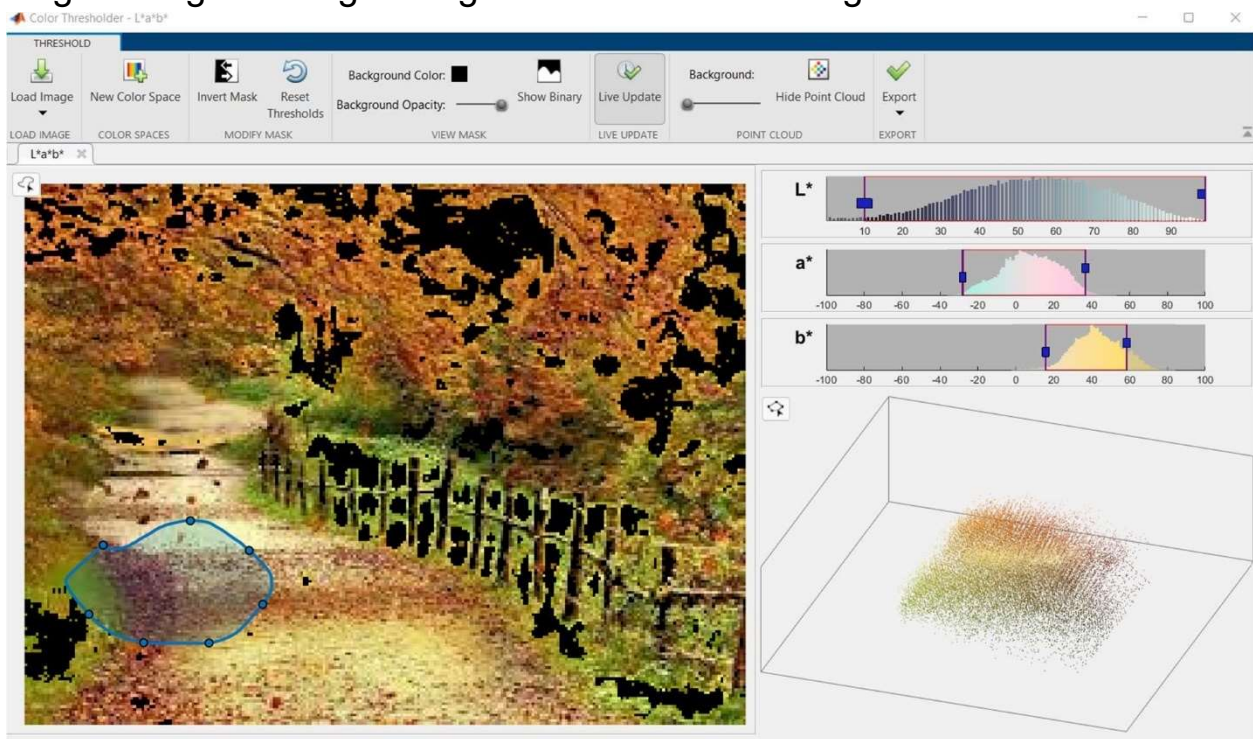


Color spaces shown by image:

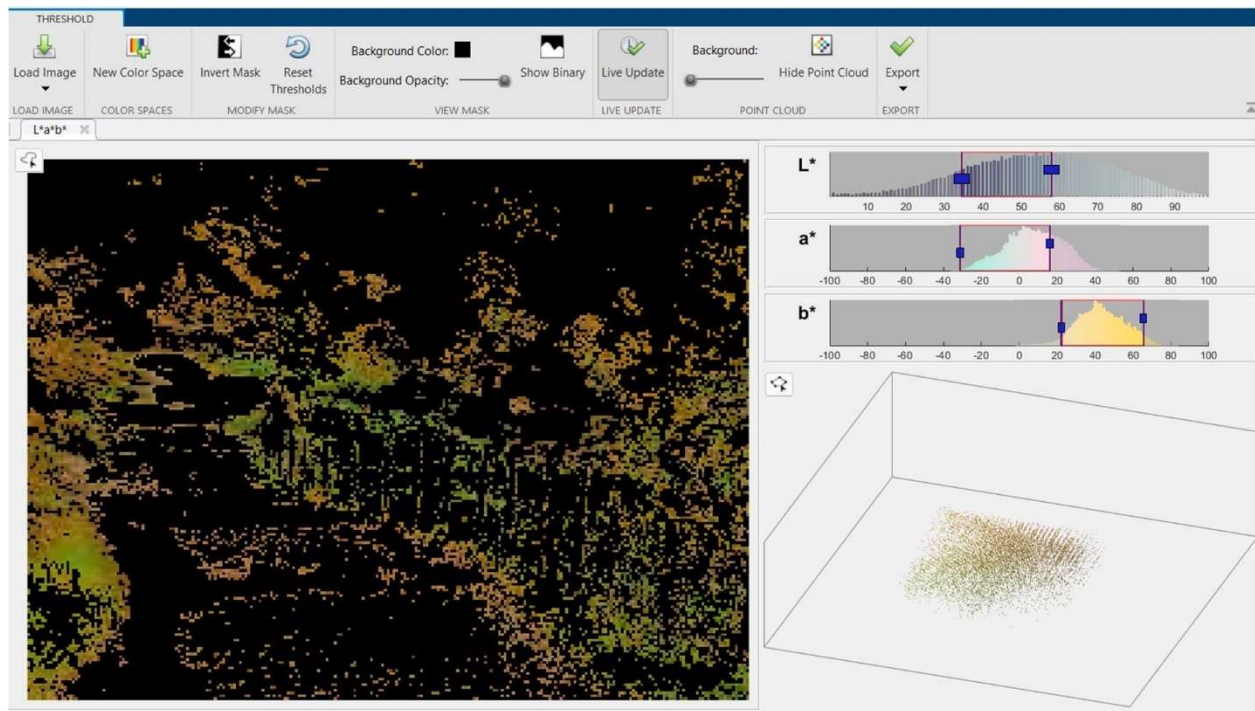




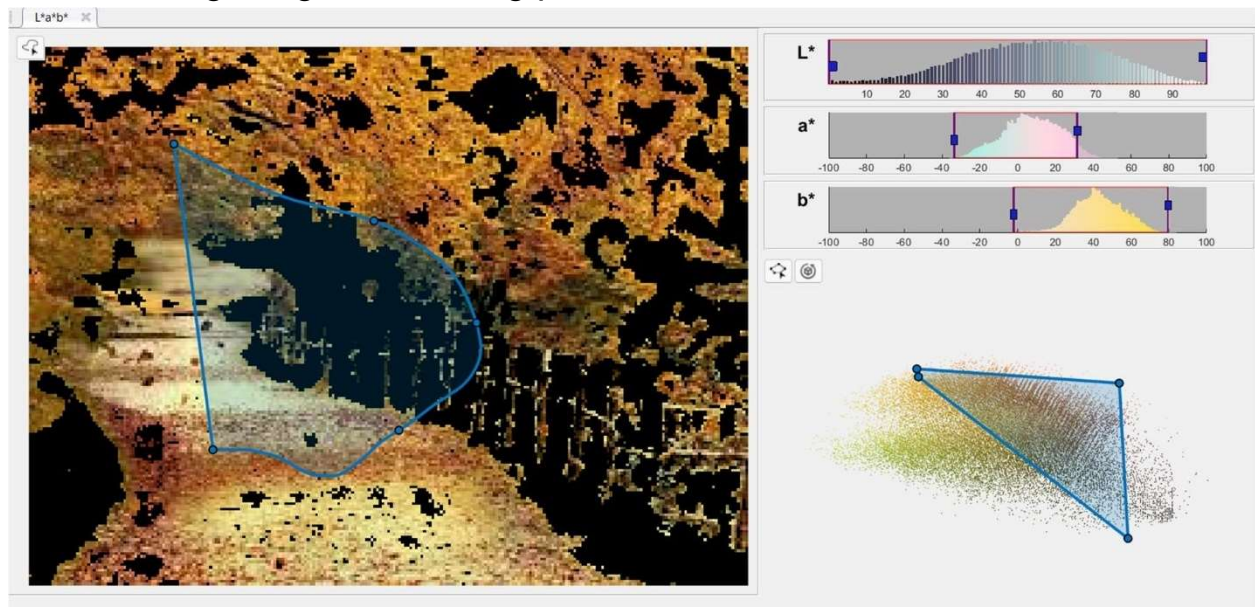
Segmenting the image using automatic thresholding:



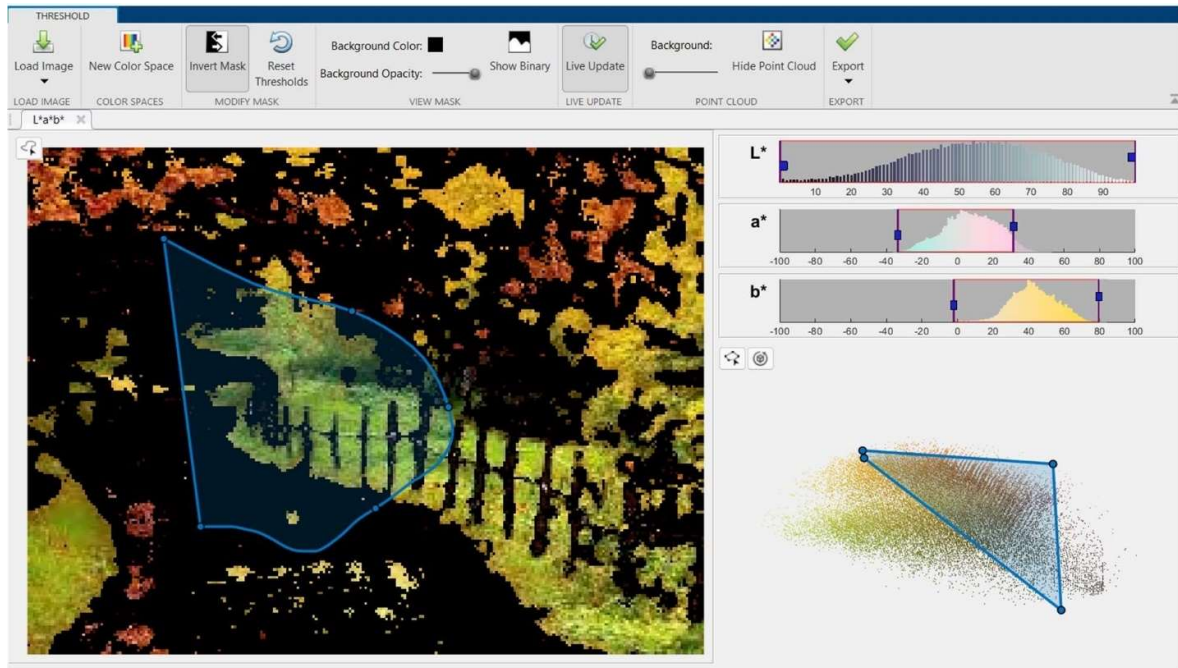
Refining automatic thresholding using color controls:



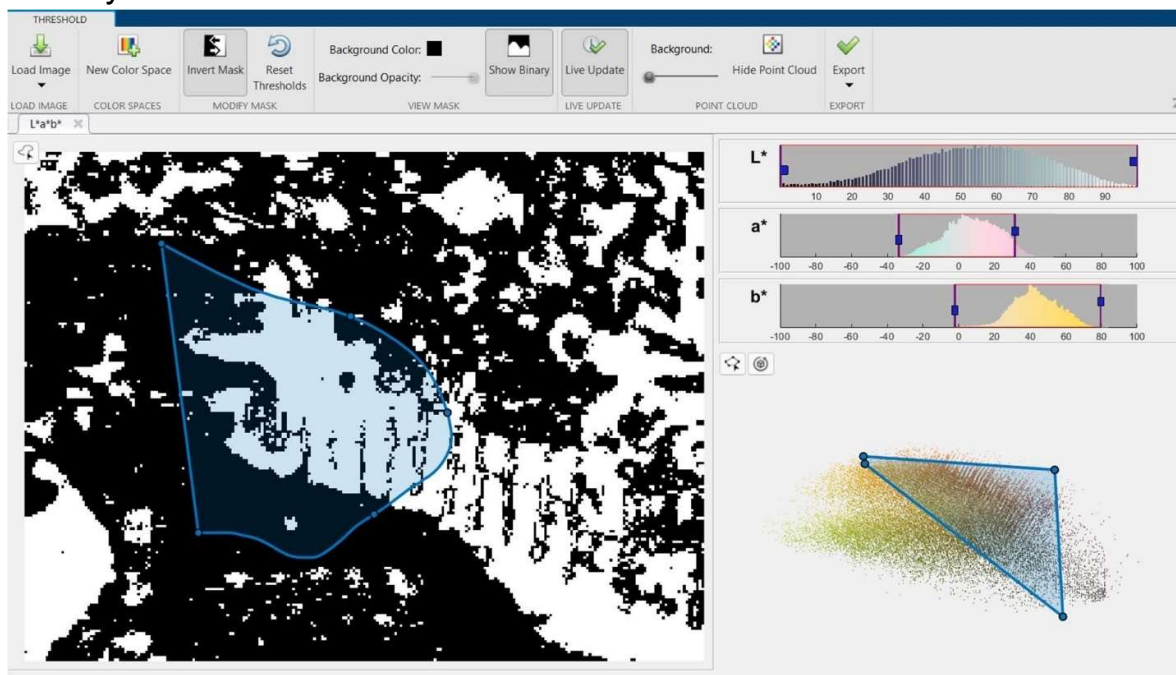
Thresholding image color using point cloud:



Invert mask:



Binary mask:



Conclusion: In this experiment we have performed Image Segmentation using thresholding. And Created Mask of different types Using Color Thresholder App.