**JAYPEE INSTITUTE OF INFORMATION TECHNOLOGY**



**Computer Graphics & Image Processing**

**Project Report** On Image segmentation & various techniques

**Submitted To: Submitted By:**

Mr. Pawan k. Upadhyay Ayush Raghuwanshi

**ACKNOWLEDGMENT**

We would like to express our special thanks of gratitude to our Computer Graphics & Image Processingteacher Mr. Pawan k. Upadhyay who gave us the golden opportunity to do this wonderful project on the **“**On Image segmentation & various techniques**”**, which helped us in doing a lot of research work and improved our knowledge and information on this particular topic.

We would also like to express gratitude to our friends who helped us in completing this project. It undoubtedly helped develop our understanding, view, actual facts, history and development, changes and much more.

**1.Basic Filters:**

In this we have used various filters for Edge Detection. It is used for detection of discontinuity of an image.

* First derivative Operators:
  + Sobel Mask
  + Prewitt Mask
  + Robert Mask
* Second derivative Operators:
* LOG(Laplacian of a Gaussian) Mask(σ=3)
* Canny edge detector

The gradient based method such as prewitt has the most important drawback of being sensitive to noise. Canny edge detector is less sensitive to noise but more expensive than Robert, Sobel and Prewitt. However Canny edge detector performs better than all masks.

**Code:**

clc;

close all;

clear all;

% Read Colour Image and convert it to a grey level Image

% Display the original Image

mycolourimage = imread('image1.jpg');

myimage = rgb2gray(mycolourimage);

subplot(3,3,1);

imshow(myimage); title('Original Image');

% Apply Sobel Operator

% Display only the horizontal Edges

sobelhz = edge(myimage,'sobel','horizontal');

subplot(3,3,2);

imshow(sobelhz,[]); title('Sobel - Horizontal Edges');

% Apply Sobel Operator

% Display only the vertical Edges

sobelvrt = edge(myimage,'sobel','vertical');

subplot(3,3,3);

imshow(sobelhz,[]); title('Sobel - Vertical Edges');

% Apply Sobel Operator

% Display both horizontal and vertical Edges

sobelvrthz = edge(myimage,'sobel','both');

subplot(3,3,4);

imshow(sobelvrthz,[]); title('Sobel - All edges');

% Apply Roberts Operator

% Display both horizontal and vertical Edges

robertsedg = edge(myimage,'roberts');

subplot(3,3,5);

imshow(robertsedg,[]); title('Roberts - Edges');

% Apply Prewitt Operator

% Display both horizontal and vertical Edges

robertsedg = edge(myimage,'prewitt');

subplot(3,3,6);

imshow(robertsedg,[]); title('Prewitt - Edges');

% Apply Laplacian Filter

f=fspecial('laplacian');

lapedg = imfilter(myimage,f,'symmetric');

subplot(3,3,7);

imshow(lapedg,[]); title('Laplacian Filter');

% Apply LOG edge detection

% The sigma used is 3

f=fspecial('log',[15,15],3.0);

logedg1 = edge(myimage,'zerocross',[],f);

subplot(3,3,8);

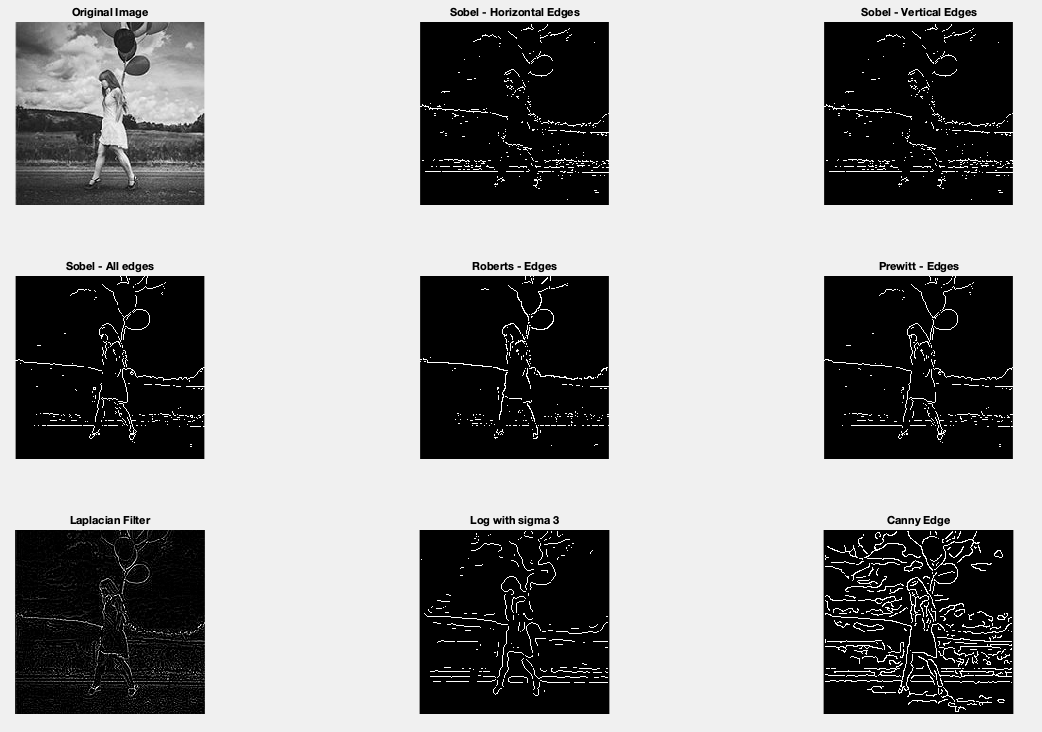
imshow(logedg1); title('Log with sigma 3');

% Apply Canny edge detection

cannyedg = edge(myimage,'canny');

subplot(3,3,9);

imshow(cannyedg,[]); title('Canny Edge');

**Output:** 

2. **Circular Hough Transform:**

It is a global processing and a specialization of Hough Transform. It is used to detect the circles in the input image. This transform is selective to circles and ignore elongated ellipses. The transform effectively searches for objects with a high degree of radial symmetry, with each degree of symmetry receiving one "vote" in the search space.

**Code:**

% Read the image that have circles

i=imread('image14.jpg');

% show image

imshow(i)

% select max & min threshold of circles we want to detect

Rmin = 10

Rmax = 50;

% Apply Hough circular transform

[centersDark1, radiiDark1] = imfindcircles(i, [Rmin Rmax],'ObjectPolarity','dark','Sensitivity',0.92);

% show the detected circles by Red color --

viscircles(centersDark1, radiiDark1,'LineStyle','--')

**Output: **

**3.Threshold Selection:** It is a local thresholding method in which we are thresholding an input image locally by passing some parameters. We take an image that is already edge operated.

**Code:**

image = imread('image2.jpeg');

mean\_image = imfilter(image, fspecial('average',[15,15]),'replica');

subtract = image - (mean\_image+20);

black\_white = im2bw(subtract,0);

subplot(1,2,1); imshow(black\_white); title('Threshold Image');

subplot(1,2,2); imshow(image); title('Original Image');

**Output:**

****

**4.Otsu Method:** It is a method of optimum global thresholding. Otsu gives best result when compared to threshold selection under local thresholding. This method minimize the weighted within class variance.

**Code:**

close all;

I1=imread('image2.jpeg');

%I1=rgb2gray(I);

imshow(I1);

figure, imhist(I1);

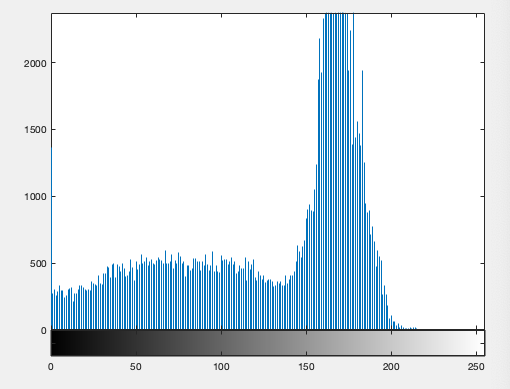
T2 = graythresh(I1);

it2= im2bw(I1,T2);

figure,imshow(it2);

**Output:**

****

****

**5. Morphological watershed:** In this the local minima of gray levels gives the catchment basins and the local maxima defines the watershed lines. A solution is to limit the number of regional minima. In output image it is easy to detect the markers.

**Code:**

close all;

I= imread('image10.png');

%I rgb2gray(RGB);

I1 = imtophat(I, strel('disk', 10));

figure, imshow(I1);

I2 = imadjust(I1);

figure,imshow(I2);

level = graythresh(I2);

BW = im2bw(I2,level);

figure,imshow(BW);

C=~BW;

figure,imshow(C);

D = ~bwdist(C);

D(C) = -Inf;

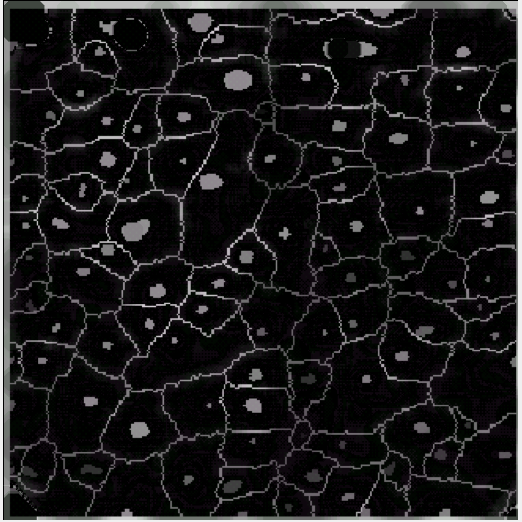
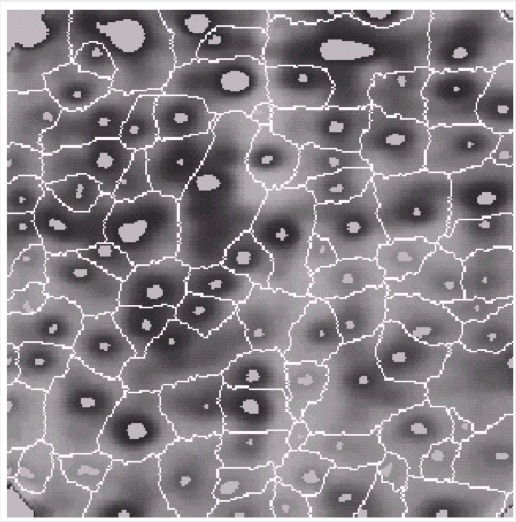
L = watershed(D);

Wi=label2rgb(L,'hot','w');

figure,imshow(Wi);

im=I;

**Output:**

****

**6.K-means Clustering:** It is an algorithm which is used to segment the interest area from the background. Partition the data points into K clusters randomly. Find the centroids of each cluster.

It operates on 2D array where pixel in row and RGB in column. We take mean values for each class(K=3).

**Code:**

% Load in an input image

im = imread('image12.png');

% We also cast to a double array, because K-means requires it in matlab

imflat = double(reshape(im, size(im,1) \* size(im,2), 3));

K = 3

[kIDs, kC] = kmeans(imflat, K, 'Display', 'iter', 'MaxIter', 150, 'Start', 'sample');

colormap = kC / 256; % Scale 0-1, since this is what matlab wants

% Reshape kIDs back into the original image shape

imout = reshape(uint8(kIDs), size(im,1), size(im,2));

imwrite(imout - 1, colormap, 'image6.jpg');

**Output:**

****