

**THE UNIVERSITY OF TEXAS AT ARLINGTON, TEXAS  
DEPARTMENT OF ELECTRICAL ENGINEERING**

**EE 5356**

**DIGITAL IMAGE PROCESSING**

**PROJECT # 10**

**by**

**SOUTRIK MAITI**

**1001569883**

**Presented to**

**Dr. K.R.RAO**

**April 12, 2019**

**Edge Detection**

***MATLAB Code:***

clc;

clear all;

close all;

%% Reading the image and converting to double

img = imread('lena512.bmp');

img = double(img);

%% vertical and horizontal kernels for Sobel operator

sobel\_h1 = [1,0,-1;

2,0,-2;

1,0,-1];

sobel\_h2 = [1,2,1;

0,0,0;

-1,-2,-1];

%% vertical and horizontal kernels for Prewitt operator

prewitt\_h1 = [-1,0,1;

-1,0,1;

-1,0,1];

prewitt\_h2 = [-1,-1,-1;

0,0,0;

1,1,1];

%% vertical and horizontal kernels for Robel operator

robel\_h1 = [1,0;

0,-1];

robel\_h2 = [0,1;

-1,0];

%% Kernel for laplacian of Gaussian

lap\_gauss = [1,1,1;

1,-8,1;

1,1,1];

%% Kernels for Canny's Edge Detection Algorithm

cy\_H=[2,4,5,4,2;

4,9,12,9,4;

5,12,15,12,5;

4,9,12,9,4;

2,4,5,4,2]/159;

cy\_H\_1=[-1,0,1;

-2,0,2;

-1,0,1];

cy\_H\_2=[1,2,1;

0,0,0;

-1,-2,-1];

%% Applying the Sobel Operator on the image

[sb\_1,sb\_2,sb\_3]=sobel\_op(img,sobel\_h1,sobel\_h2);

%% Displaying Original Image with results of sobel operator

figure(1);

subplot(2,2,1);

imshow(uint8(img));

title('Original Image');

subplot(2,2,2);

imshow(uint8(sb\_1));

title('sobel image before threshold');

subplot(2,2,3);

imshow(sb\_2);

title('sobel image after threshold');

subplot(2,2,4);

imshow(sb\_3);

title('Default sobel function');

saveas(gca,'sobel.jpg');

%% Applying the Sobel Operator on the image

[pw\_1,pw\_2,pw\_3]=prewitt\_op(img,prewitt\_h1,prewitt\_h2);

%% Displaying Original Image with results of Prewitt operator

figure(2);

subplot(2,2,1);

imshow(uint8(img));

title('Original Image');

subplot(2,2,2);

imshow(uint8(pw\_1));

title('Prewitt image before threshold');

subplot(2,2,3);

imshow(pw\_2);

title('Prewitt image after threshold');

subplot(2,2,4);

imshow(pw\_3);

title('Default Prewitt function');

saveas(gca,'prewitt.jpg');

%% Applying the Robel Operator on the image

[rob\_1,rob\_2,rob\_3]=robel\_op(img,robel\_h1,robel\_h2);

%% Displaying Original Image with results of Robel operator

figure(3);

subplot(2,2,1);

imshow(uint8(img));

title('Original Image');

subplot(2,2,2);

imshow(uint8(rob\_1));

title('Robel image before threshold');

subplot(2,2,3);

imshow(rob\_2);

title('Robel image after threshold');

subplot(2,2,4);

imshow(rob\_3);

title('Default Prewitt function');

saveas(gca,'robel.jpg');

%% Applying the Laplacian of Gaussian Operator on the image

[lg\_1,lg\_2,lg\_3]=gauss\_op(img,lap\_gauss);

%% Displaying Original Image with results of Laplacian of the Gaussian Operator

figure(4);

subplot(2,2,1);

imshow(uint8(img));

title('Original Image');

subplot(2,2,2);

imshow(uint8(lg\_1));

title('L(Gaussian)image before threshold');

subplot(2,2,3);

imshow(lg\_2);

title('L(Gaussian)image after threshold');

subplot(2,2,4);

imshow(lg\_3);

title('Default Laplacian of Gaussian function');

saveas(gca,'lap\_gauss.jpg');

%% Applying Canny Edge detection on image

[cy\_1,cy\_2,cy\_3]=cy\_op(cy\_H,cy\_H\_1,cy\_H\_2,img);

%% Displaying original image and Canny image

figure(6);

subplot(2,2,1);

imshow(uint8(img));

title('Original Image');

subplot(2,2,2);

imshow(uint8(lg\_1));

title('Canny image before threshold');

subplot(2,2,3);

imshow(lg\_2);

title('Canny image after threshold');

subplot(2,2,4);

imshow(lg\_3);

title('Default Canny Algorithm');

saveas(gca,'canny.jpg');

%%%%%%%%%%%%%%%%%%%%%%%%%% Function Definitions %%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Function definition for Sobel Operator

function [ sb\_1,sb\_2,sb\_3 ] = sobel\_op(img,sobel\_h1,sobel\_h2)

ii\_c1 = conv2(img,sobel\_h1,'same');

ii\_c2 = conv2(img,sobel\_h2,'same');

sb\_1 = sqrt((ii\_c1).^2 + (ii\_c2).^2);

sb\_2 = (sb\_1 > 200);

sb\_3 = edge(img,'sobel',[]);

end

%% Function defined for Prewitt Operator

function [pw\_1,pw\_2,pw\_3]=prewitt\_op(img,prewitt\_H\_1,prewitt\_H\_2)

ii\_c1 = conv2(img,prewitt\_H\_1);

ii\_c2 = conv2(img,prewitt\_H\_2);

pw\_1 = sqrt(ii\_c1.^2 + ii\_c2.^2);

pw\_2 = (pw\_1 > 200);

pw\_3 = edge(img,'prewitt',[]);

end

%% Function defined for Robel Operator

function [rob\_1,rob\_2,rob\_3]=robel\_op(img,robel\_h1,robel\_h2)

ro\_bel\_1 = conv2(img,robel\_h1);

ro\_bel\_2 = conv2(img,robel\_h2);

rob\_1 = sqrt(ro\_bel\_1.^2 + ro\_bel\_2.^2);

rob\_2 = (rob\_1 > 60);

rob\_3 = edge(img,'roberts',[]);

end

%% Function defined for Laplacian of Gaussian Operator

function[lg\_1,lg\_2,lg\_3]=gauss\_op(img,lap\_gauss)

i\_filt = fspecial('gaussian');

i\_filt = imfilter(img,i\_filt);

lg\_1 = conv2(i\_filt,lap\_gauss);

lg\_2 = (lg\_1 > 50);

lg\_3 = edge(img,'log',[]);

end

%% Functions for doing the Canny Edge Detection Algorithm

function [cy\_1,cy\_2,cy\_3]=cy\_op(cy\_H,cy\_H\_1,cy\_H\_2,img)

ii\_c1 = conv2(img,cy\_H);

ii\_cx = conv2(ii\_c1,cy\_H\_1);

ii\_cy = conv2(ii\_c1,cy\_H\_2);

cy\_1 = abs(ii\_cx)+abs(ii\_cy);

phas\_1\_12 = abs(atan2(ii\_cy,ii\_cx)\*(180/pi));

dimen\_sion = size(img);

for ii = 1:dimen\_sion(1)

for jj = 1:dimen\_sion(2)

if (phas\_1\_12(ii,jj) <= 22.5 && phas\_1\_12(ii,jj) > 157.5)

phas\_1\_12(ii,jj) = 0;

elseif (phas\_1\_12(ii,jj) > 22.5 && phas\_1\_12(ii,jj) <= 67.5)

phas\_1\_12(ii,jj) = 45;

elseif (phas\_1\_12(ii,jj) > 67.5 && phas\_1\_12(ii,jj) <= 112.5)

phas\_1\_12(ii,jj) = 90;

elseif (phas\_1\_12(ii,jj) > 112.5 && phas\_1\_12(ii,jj) <= 157.5)

phas\_1\_12(ii,jj) = 135;

end

end

end

tp1\_12 = refr\_1(cy\_1,phas\_1\_12);

tp2 = (tp1\_12 > 50);

[t1, t2] = find(tp1\_12 > 60);

cy\_2 = bwselect(tp2, t2, t1, 8);

cy\_3=edge(img,'canny');

end

function tmp = refr\_1(cy,cy\_agle)

dimen\_sion = size(cy);

tmp = zeros(dimen\_sion(1),dimen\_sion(2));

agle = [0:180].\*pi/180;

xx\_off = 1.5\*cos(agle);

yy\_off = 1.5\*sin(agle);

hh\_fract = xx\_off - floor(xx\_off);

vv\_fract = yy\_off - floor(yy\_off);

cy\_agle = fix(cy\_agle)+1;

for ii =3:(dimen\_sion(1) - 3)

for jj =3:(dimen\_sion(2) - 3)

o\_r = cy\_agle(ii,jj);

x\_1 = jj + xx\_off(o\_r);

y\_1 = ii - yy\_off(o\_r);

x\_2 = floor(x\_1);

x\_x = ceil(x\_1);

y\_2 = floor(y\_1);

y\_y = ceil(y\_1);

a\_1 = cy(y\_2,x\_2);

a\_2 = cy(y\_2,x\_x);

a\_3 = cy(y\_y,x\_2);

a\_4 = cy(y\_y,x\_x);

uper\_avg = a\_1 + hh\_fract(o\_r) \* (a\_2 - a\_1);

lwer\_avg = a\_3 + hh\_fract(o\_r) \* (a\_4 - a\_3);

v\_1 = uper\_avg + vv\_fract(o\_r) \* (lwer\_avg - uper\_avg);

if (cy(ii, jj) > v\_1)

x\_1 = jj - xx\_off(o\_r);

y\_1 = ii + yy\_off(o\_r);

x\_2 = floor(x\_1);

x\_x = ceil(x\_1);

y\_2 = floor(y\_1);

y\_y = ceil(y\_1);

a\_1 = cy(y\_2,x\_2);

a\_2 = cy(y\_2,x\_x);

a\_3 = cy(y\_y,x\_2);

a\_4 = cy(y\_y,x\_x);

uper\_avg = a\_1 + hh\_fract(o\_r) \* (a\_2 - a\_1);

lwer\_avg = a\_3 + hh\_fract(o\_r) \* (a\_4 - a\_3);

v2 = uper\_avg + vv\_fract(o\_r) \* (lwer\_avg - uper\_avg);

if (cy(ii,jj) > v2)

tmp(ii, jj) = cy(ii, jj);

end

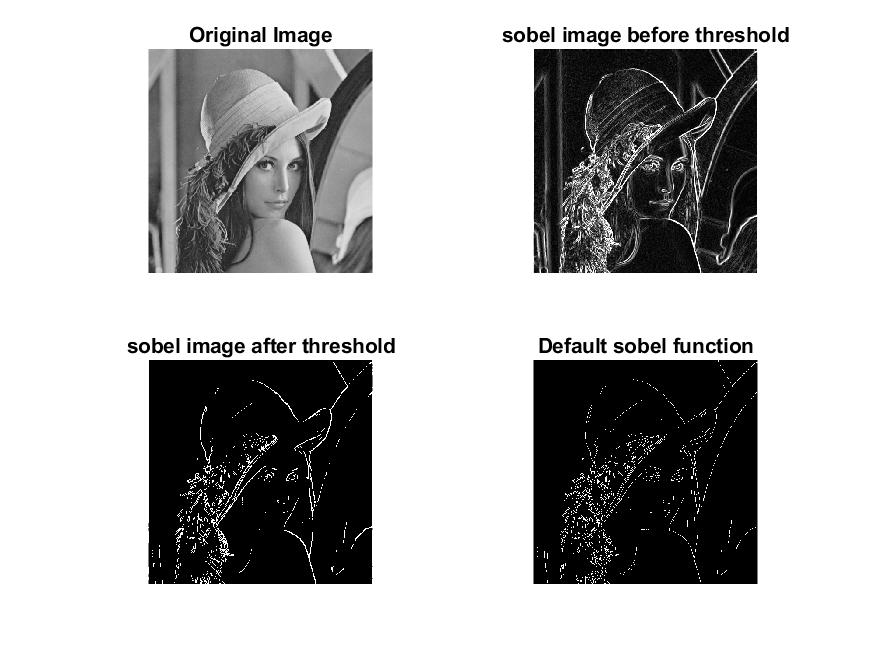
end

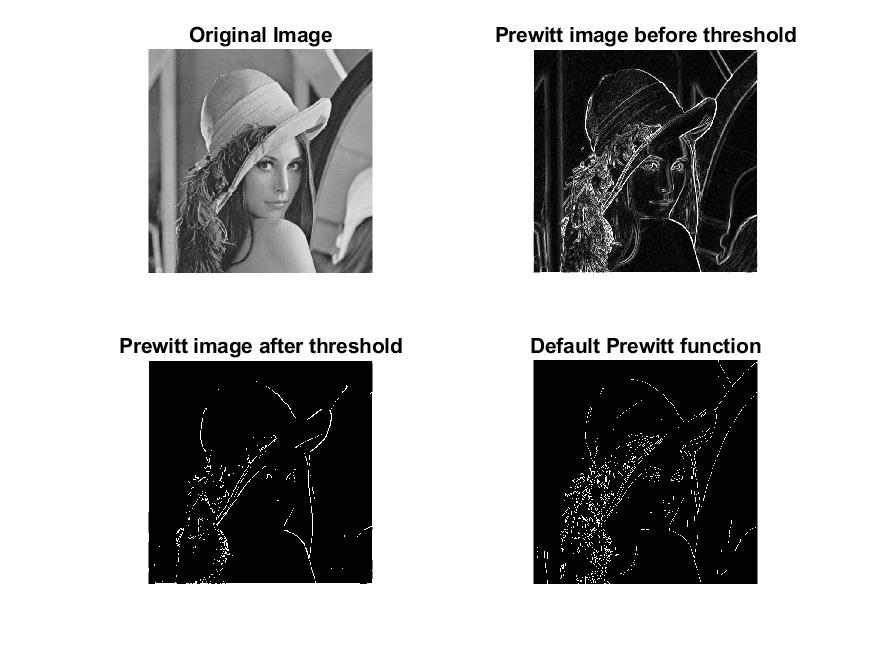
end

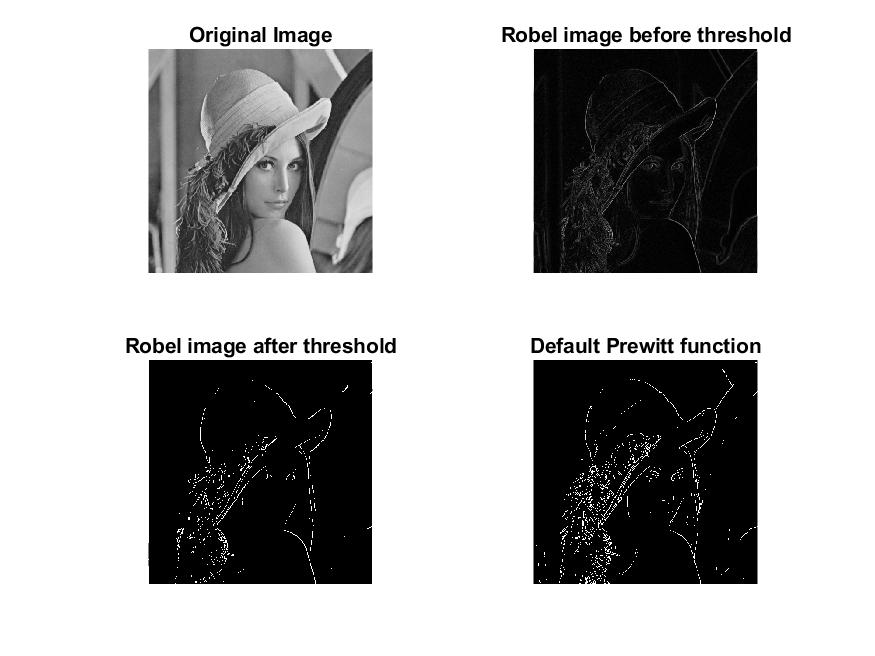
end

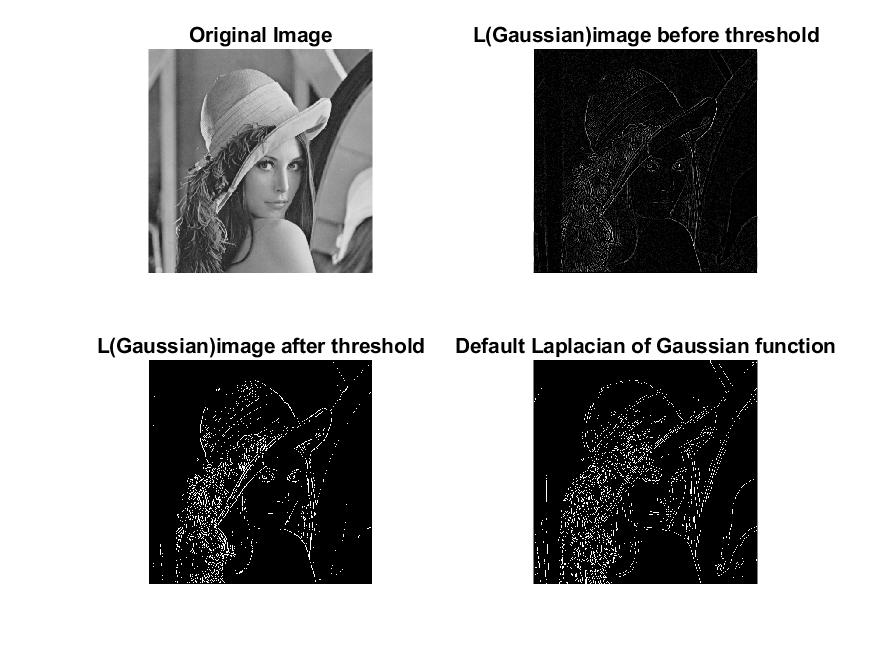
end

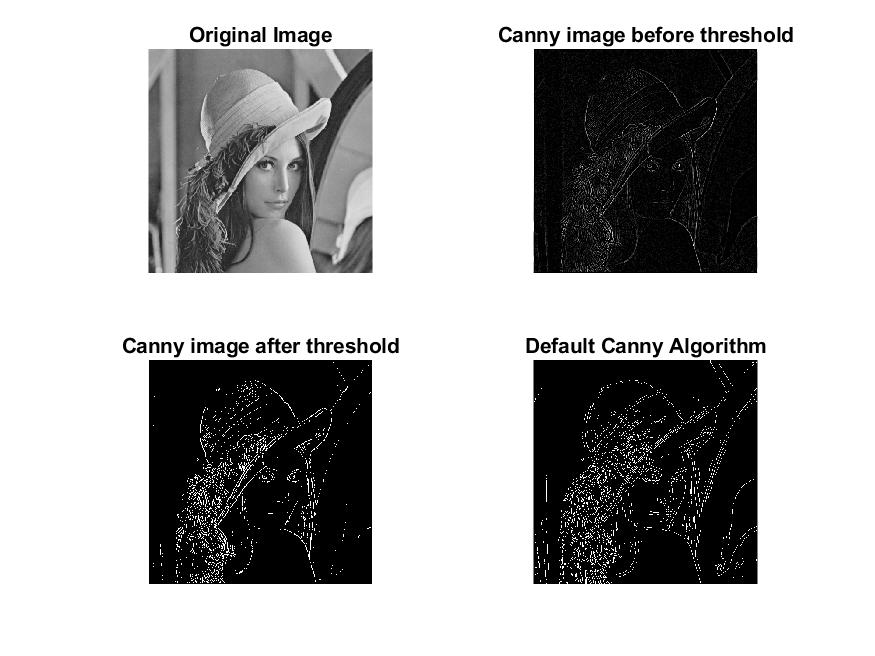
***Results:***

******





******

******

***Conclusion:***

* Edge detection is an image processing technique for finding the boundaries of objects within images. It works by detecting discontinuities in brightness.
* Sobel Operator is essentially a spatial gradient measurement of an image. It highlights regions of higher spatial frequencies which are the edges in the image.
* Prewitt Operator calculates the approximate derivative using two 3 X 3 kernels, one corresponding to horizontal changes and the other to the vertical changes.
* Roberts Cross operator estimates 2-D spatial gradient measurement of the image, higher frequencies corresponding to edges. It is mostly used in grayscale images.
* The Laplacian of Gaussian method is 2-D isotropic measure of 2nd spatial derivative which highlights the regions of rapid intensity change thereby, detecting the edges.
* Canny Edge Detection first smoothens the image, thereby eliminating noise & then finds image gradient to highlight the high spatial derivatives. Canny’s edge detection is based on three criteria namely, low error rate, localization of edge points and response of single point.