EE5356\_ DIGITAL IMAGE PROCESSING

ASSIGNMENT # 10

NAME: PAVAI ARCHIMEDES

ST ID: 1001233996

# Edge detection:

Take a (256×256) or (512×512) 8-bit/pel image and perform the following edge detection operations:

1. Sobel Operator

2. Prewitt Operator

3. Robel operator

4. Laplacian of Gaussian

5. Canny’s Edge Detection

**Procedure:**

1) Read an image (any size up to 512x512).

2) Perform the edge detection using the techniques mentioned above.

3) Apply proper thresholding method and observe the difference in the image before and after applying thresholding.

4) Compare the final image obtained using default MATLAB Edge detection operator.

MATLAB PROGRAM:

MAIN FUNCTION:

clc;

clear all;

close all;

I=imread('C:\Users\PAVAI ARCHIMEDES\Desktop\girl512.bmp');

figure(1);

imshow(I);

title('Original image');

plll = [1,0,-1;2,0,-2;1,0,-1];

ppy = [1,2,1;0,0,0;-1,-2,-1];

sobel1(I,plll,ppy);

plll = [-1,0,1;-1,0,1;-1,0,1];

ppy = [-1,-1,-1;0,0,0;1,1,1];

prewitt1(I,plll,ppy);

plll = [1,0;0,-1];

ppy = [0,1;-1,0];

robel1(I,plll,ppy);

Gaussian = [1,1,1;1,-8,1;1,1,1];

gaussian1(I,Gaussian);

G=[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;

plll=[-1,0,1;-2,0,2;-1,0,1];

ppy=[1,2,1;0,0,0;-1,-2,-1];

canny1(G,plll,ppy,I);

function prewitt1(i,Gx,Gy)

i\_c1 = conv2(i,Gx);

i\_c2 = conv2(i,Gy);

prewitt\_filter = sqrt(i\_c1.^2 + i\_c2.^2);

threshold = (prewitt\_filter > 200);

prewitt\_default = edge(i,'prewitt',[]);

figure(4);

subplot(1,2,1);

imshow(uint8(prewitt\_filter));

title('prewitt before threshold');

subplot(1,2,2);

imshow(threshold);

title('prewitt after threshold');

figure(5);

imshow(prewitt\_default);

title('actual prewitt function');

end

function robel1(i,pll,ppy)

robel\_1 = conv2(i,pll);

robel\_2 = conv2(i,ppy);

robel\_filter = sqrt(robel\_1.^2 + robel\_2.^2);

threshold = (robel\_filter > 60);

robel\_default = edge(i,'roberts',[]);

figure(6);

subplot(1,2,1);

imshow(uint8(robel\_filter));

title('robel before threshold');

subplot(1,2,2);

imshow(threshold);

title('robel after threshold');

figure(7);

imshow(robel\_default);

title('actual robel function');

end

function sobel1(i,pll,ppy)

i\_c1 = conv2(i,pll);

i\_c2 = conv2(i,ppy);

sobel\_filter = sqrt((i\_c1).^2 + (i\_c2).^2);

threshold = (sobel\_filter > 200);

sobel\_default = edge(i,'sobel',[]);

figure(2);

subplot(1,2,1);

imshow(uint8(sobel\_filter));

title('sobel before threshold');

subplot(1,2,2);

imshow(threshold);

title('sobel after threshold');

figure(3);

imshow(sobel\_default);

title('actual sobel function');

end

function gaussian1(i,Gaussian)

k\_filtrr = fspecial('gaussian');

k\_filt = imfilter(i,k\_filtrr);

k\_c1kk = conv2(k\_filt,Gaussian);

threshold = (k\_c1kk > 50);

gaussian\_default = edge(i,'log',[]);

figure(8);

subplot(1,2,1);

imshow(uint8(k\_c1kk),[]);

title('gaus before threshold');

subplot(1,2,2);

imshow(threshold);

title('gaus after threshold');

figure(9);

imshow(gaussian\_default);

title('actual gaussian function');

end

function canny1(G,Gx,Gy,I)

kk\_c1 = conv2(I,G);

ikk\_cx = conv2(kk\_c1,Gx);

kk\_cy = conv2(kk\_c1,Gy);

mag = abs(ikk\_cx)+abs(kk\_cy);

phase\_1 = abs(atan2(kk\_cy,ikk\_cx)\*(180/pi));

dim = size(I);

for i = 1:dim(1)

for j = 1:dim(2)

if (phase\_1(i,j) <= 22.5 && phase\_1(i,j) > 157.5)

phase\_1(i,j) = 0;

elseif (phase\_1(i,j) > 22.5 && phase\_1(i,j) <= 67.5)

phase\_1(i,j) = 45;

elseif (phase\_1(i,j) > 67.5 && phase\_1(i,j) <= 112.5)

phase\_1(i,j) = 90;

elseif (phase\_1(i,j) > 112.5 && phase\_1(i,j) <= 157.5)

phase\_1(i,j) = 135;

end

end

end

t1 = ref(mag,phase\_1);

t2 = (t1 > 50);

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

T = bwselect(t2, t2, t1, 8);

canny\_default=edge(I,'canny');

figure(10);

subplot(1,2,1);

imshow(uint8(mag));

title('canny before threshold');

subplot(1,2,2);

imshow(uint8(T));

title('canny after threshold');

figure(11);

imshow(canny\_default);

title('actual canny function');

end

function t = ref(canny,canny\_angle)

dim = size(canny);

t = zeros(dim(1),dim(2));

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

x\_yy = 1.5\*cos(an);

y\_off = 1.5\*sin(an);

h\_fra = x\_yy - floor(x\_yy);

v\_fra = y\_off - floor(y\_off);

canny\_angle = fix(canny\_angle)+1;

for i =3:(dim(1) - 3)

for j =3:(dim(2) - 3)

or = canny\_angle(i,j);

x1 = j + x\_yy(or);

y1 = i - y\_off(or);

x2 = floor(x1);

xx = ceil(x1);

y2 = floor(y1);

yy = ceil(y1);

a1 = canny(y2,x2);

a2 = canny(y2,xx);

a3 = canny(yy,x2);

a4 = canny(yy,xx);

upper\_avg = a1 + h\_fra(or) \* (a2 - a1);

lower\_avg = a3 + h\_fra(or) \* (a4 - a3);

v1 = upper\_avg + v\_fra(or) \* (lower\_avg - upper\_avg);

if (canny(i, j) > v1)

x1 = j - x\_yy(or);

y1 = i + y\_off(or);

x2 = floor(x1);

xx = ceil(x1);

y2 = floor(y1);

yy = ceil(y1);

a1 = canny(y2,x2);

a2 = canny(y2,xx);

a3 = canny(yy,x2);

a4 = canny(yy,xx);

upper\_avg = a1 + h\_fra(or) \* (a2 - a1);

lower\_avg = a3 + h\_fra(or) \* (a4 - a3);

v2 = upper\_avg + v\_fra(or) \* (lower\_avg - upper\_avg);

if (canny(i,j) > v2)

t(i, j) = canny(i, j);

end

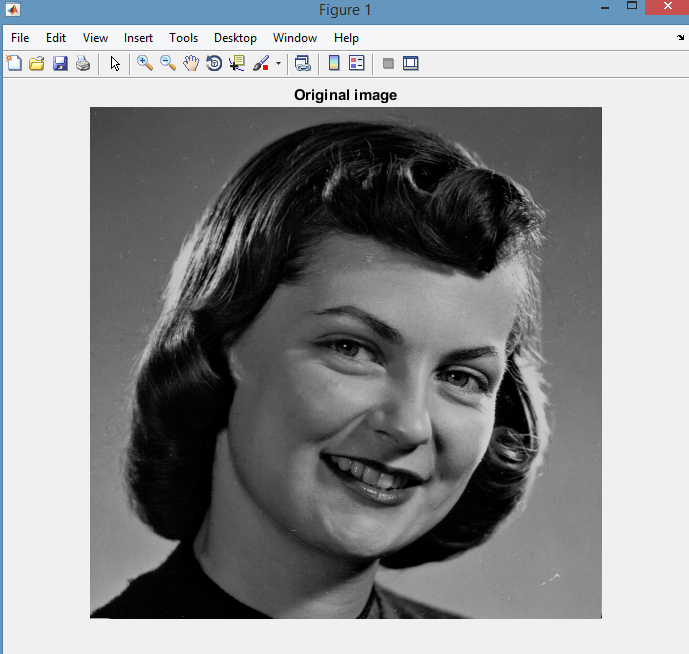
end

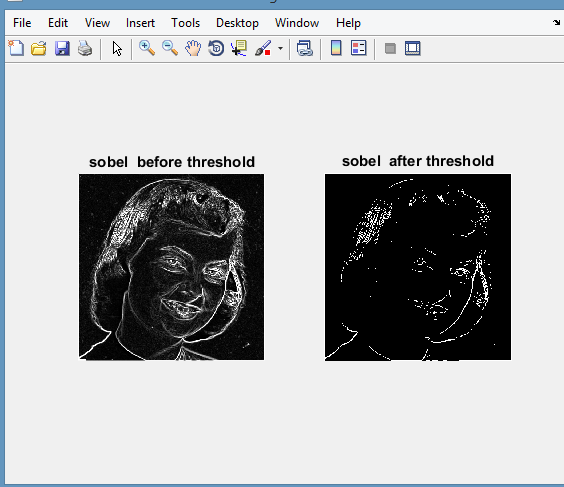
end

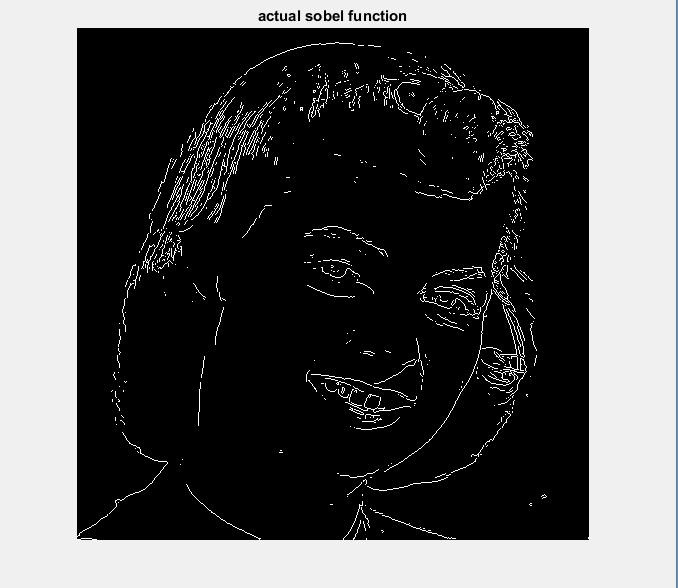
end

end

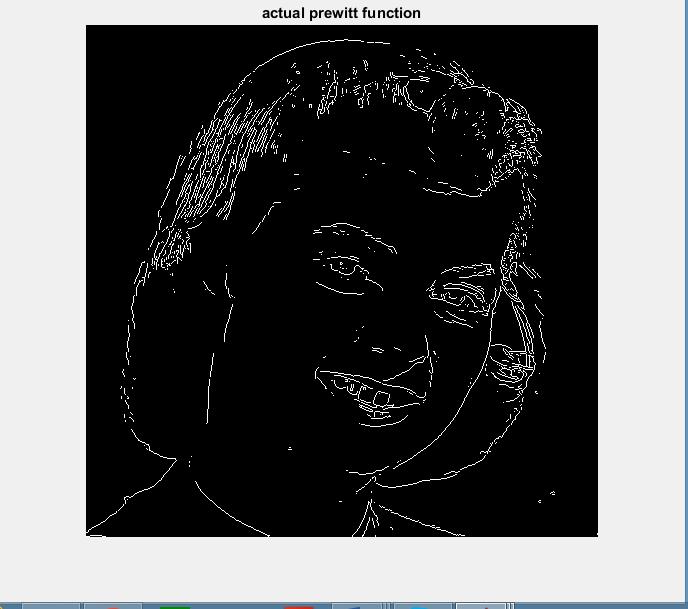
RESULTS:

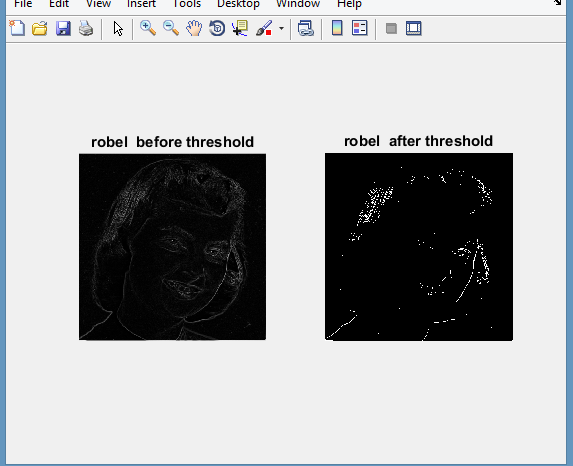


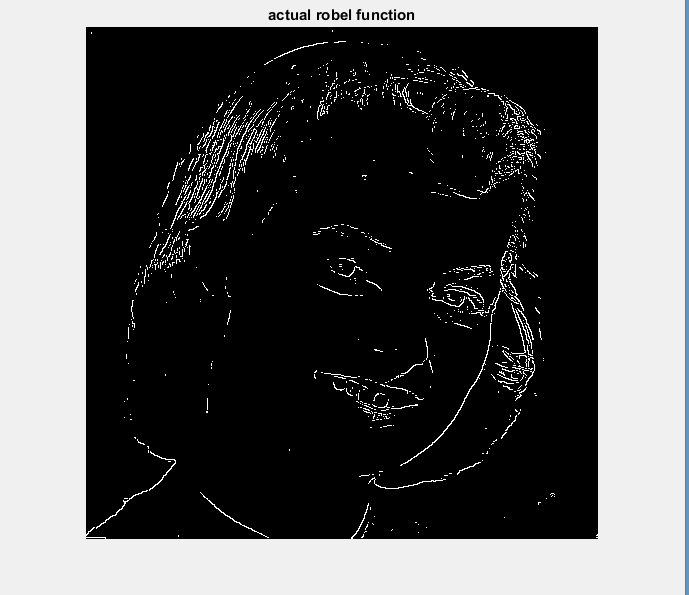






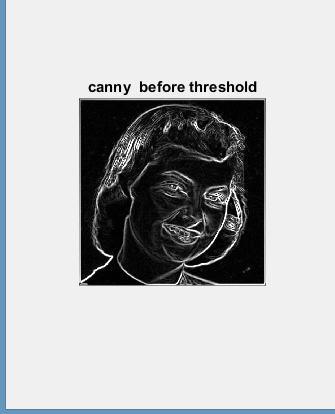


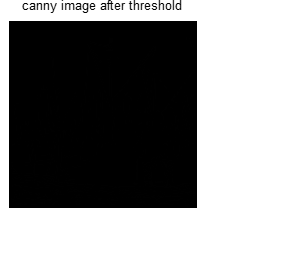














CONCLUSION:

* proper thresholding method is applied and the difference in the image before and after applying thresholding is observed.
* The final image obtained using default MATLAB Edge detection operator.