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);
```

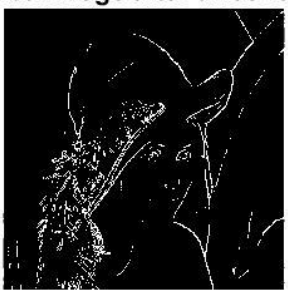
```
            y_1 = ii - yy_off(o_r);
            x_2 = floor(x_1);
            x_x = ceil(x_1);
            y_2 = floor(y_1);
            y_y = ceil(y_1);
            a_1 = cy(y_2,x_2);
            a_2 = cy(y_2,x_x);
            a_3 = cy(y_y,x_2);
            a_4 = cy(y_y,x_x);
            uper_avg = 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:

Original Image

sobel image before threshold

sobel image after threshold



Default sobel function

Original Image

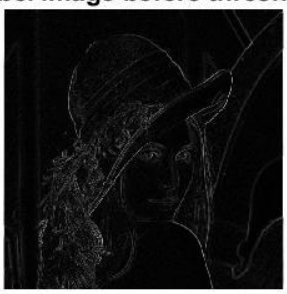
Prewitt image before threshold

Prewitt image after threshold

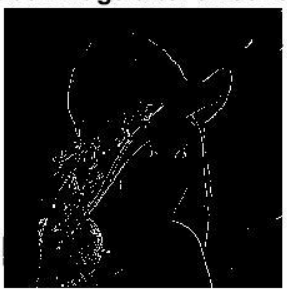
Default Prewitt function

Original Image

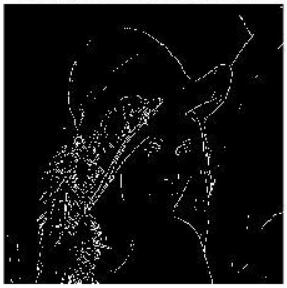
Robel image before threshold



Robel image after threshold



Default Prewitt function



Original Image

L(Gaussian)image before threshold

L(Gaussian)image after threshold

Default Laplacian of Gaussian function

Original Image

Canny image before threshold

Canny image after threshold

Default Canny Algorithm

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 2^{nd} 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.