EE 417 COMPUTER VISION

Lab #3: Edge Detection with First and Second Order Derivative Filters

Lab Date: 27 October 2021

Osman Burak Çevik

[osmanburak@sabanciuniv.edu](mailto:osmanburak@sabanciuniv.edu)

26399

Introduction

Lab #3 focuses on implementing edge detection algorithms based on Sobel, Prewitt and Laplacion of Gaussian operators.

Prewitt Operator

Prewitt filtering is a discrete 2D derivative operation which can be applied with the following kernels

Calendar

Description automatically generated

To detect edges in a grayscale image, the gradient image obtained by the horizontal and the vertical Prewitt operators is binarized by a threshold. The gradient image is calculated as

Text, whiteboard

Description automatically generated

Prewitt operator is a gradient-based operator. It computes the gradient approximation of image intensity function for image edge detection. At the pixels of an image, the Prewitt operator produces either the normal to a vector or the corresponding gradient vector. It uses two 3x3 kernels or masks shown above which are convolved with the input image to calculate approximations of the derivatives; one for horizontal changes, one for vertical.

Results for lab3prewitt.m when the input is Object\_contours.jpg and threshold is 50:

A picture containing shape

Description automatically generated

Results for lab3prewitt.m when the input is Object\_contours.jpg and threshold is 100:

A picture containing diagram

Description automatically generated

Results for lab3prewitt.m when the input is Object\_contours.jpg and threshold is 250:

A picture containing shape

Description automatically generated

Results for lab3prewitt.m when the input is mantaRay.jpg and threshold is 20:

Timeline

Description automatically generated with medium confidence

Results for lab3prewitt.m when the input is mantaRay.jpg and threshold is 120:

A picture containing graphical user interface

Description automatically generated

Results for lab3prewitt.m when the input is mantaRay.jpg and threshold is 220:

A picture containing graphical user interface

Description automatically generated

Prewitt Operator Conclusions

Prewitt filter approximates partial derivatives of the given image with the given kernels(convolution). The partial derivatives are discrete versions of gaussian convolutions of rows and columns. The local convolutions calculate partial derivatives(X-Y axis). Increasing the threshold value makes prewitt edges disappear. This can be easily observed from the mantaRay.jpg image results with different threshold values. Prewitt operator has great performance on detecting vertical & horizontal edges and orientation of an image. However the magnitutde of the coefficient is fixed and diagonal points are not preserved all the time.

Sobel Operator

Sobel filtering is a discrete 2D derivative operation which can be applied with the following kernels

Calendar

Description automatically generated with medium confidence

To detect edges in a grayscale image by employing Sobel operators, the same procedure is applied as Prewitt based edge detection except the kernels are changed.

Results for lab3sobel.m when the input is Object\_contours.jpg and threshold is 100:

A picture containing shape

Description automatically generated

Results for lab3sobel.m when the input is Object\_contours.jpg and threshold is 200:

A picture containing shape

Description automatically generated

Results for lab3sobel.m when the input is peppers.png and threshold is 100:

A picture containing text, gallery, screenshot

Description automatically generated

Results for lab3sobel.m when the input is peppers.png and threshold is 200:

Timeline

Description automatically generated

Sobel Operator Conclusions

Sobel filter approximates partial derivatives of the given image with the given kernels(convolution). The partial derivatives are discrete versions of gaussian convolutions of rows and columns. The local convolutions calculate partial derivatives(X-Y axis). Increasing threshold makes Sobel edges disappear.

Laplacian of Gaussian Smoothed Image

As Laplace operator may detect edges as well as noise, it is des,rable to smooth the image first by a Gaussian filter. Applying the Laplacian for a Gauss-filtered image can be done in one step of convolution with the following kernel

Calendar

Description automatically generated with low confidence

Different from the Sobel and Prewitt based edge detection algorithms, zero crossing points represent the edge pixels.

Results for lab3log.m when the input is Object\_contours.jpg:

Graphical user interface

Description automatically generated

Results for lab3log.m when the input is mantaRay.jpg:

Graphical user interface

Description automatically generated

Results for lab3log.m when the input is peppers.png:

Graphical user interface

Description automatically generated

Laplacian of Gaussian Conclusions

LoG or Mexican Hat function is basically applying laplacian to the gaussian filtered image. Instead of the first derivative like Sobel and Prewitt, LoG is a second derivative function. The edges are detected by zero crossings.

I’ve used three different images therefore the graident values are different on plots(1st:

-30 to 30, 2nd: -150 to 150, 3rd: -50 to 50).

Derivative filters are very sensitive to noise. This is why LoG first applies the Gaussian filter to smoothen the image, in other words reduce the noise.

When Gaussian’s scale parameter is increased, larger scale edges can be detected. When decreased, finer features can be detected. However while eliminating noise by smoothening the image, fine details and image edges are lost. Therefore when applying the LoG, one should select what she/he is looking for!

Discussion on 1st Derivative and 2nd Derivative Edge Detection Algorithms

1st derivative filters such as Sobel and Prewitt are looking into the graph of the intensity values of the image. If the slope of the graph reaches a peak the algorithm declares that pixel as edges. However when there is noise, the algorithm detects that noise as edge as well(unexistent peak will occur). 2nd derivative filters such as LoG first smoothens the image and reduces the noise, then investigates the zero crossings. 2nd derivative graph of the intensity values of the image crosses zero when there is an edge. Therefore 2nd order filters are more advantegous because the error rate is lower thanks to smoothening.

According to Lindeberg blobs are detected as local extrema in space and scale, within the LoG scale-space volume. Therefore the LoG filter is also used for blob detection. LoG is also used in image coding(Laplacian Pyramid).

Note to TAs: Some Log Filtered images are hard to see but if you increase the brightness you can see the objects etc in the processed image and **the zero-crossings finding script is already in my lab3log.m file.**

APPENDIX

%% lAB3 edge detection

clear all; close all; clc;

%% sobel

clear all; close all; clc;

img=imread('peppers.png');

[row,col,ch]=size(img);

if (ch==3)

img=rgb2gray(img);

end

[sobel\_X, sobel\_Y, sob\_Grad, sob\_Edge]=lab3sobel(img,200);

figure

subplot(2,3,1);

imshow(img);

title('Original');

subplot(2,3,2);

imshow(sobel\_X);

title('Sobel X Filtered');

subplot(2,3,3);

imshow(sobel\_Y);

title('Sobel Y Filtered');

subplot(2,3,5);

imshow(sob\_Grad);

title('Sobel Gradient');

subplot(2,3,6);

imshow(sob\_Edge);

title('Sobel Edges');

%% prewitt

clear all; close all; clc;

im=imread('mantaRay.jpg');

[row,col,chNum]=size(im);

if (chNum==3)

im=rgb2gray(im);

end

[prewX, prewY, prewGrad, prewEdge]=lab3prewitt(im,220);

figure

subplot(2,3,1);

imshow(im);

title('Original Image');

subplot(2,3,2);

imshow(prewX);

title('Prewitt X Filtered Image');

subplot(2,3,3);

imshow(prewY);

title('Prewitt Y Filtered Image');

subplot(2,3,5);

imshow(prewGrad);

title('Prewitt Gradient');

subplot(2,3,6);

imshow(prewEdge);

title('Prewitt Edges');

%% Log

clear all; close all; clc;

img=imread('peppers.png');

Th1=10;

Th2=210;

[E]=lab3log(img,Th1,Th2);

main.m

function [x\_img,y\_img,sob\_Grad,sob\_Edge] = lab3sobel(img,thold)

[row,col,ch]=size(img);

if (ch==3)

img=rgb2gray(img);

end

img=double(img);

xfilter=[-1 0 1;-2 0 2;-1 0 1];

yfilter=[-1 -2 -1;0 0 0;1 2 1];

x\_img=conv2(img, xfilter);

y\_img=conv2(img, yfilter);

sob\_Grad=sqrt(x\_img.^2+y\_img.^2);

sob\_Edge=zeros(size(img));

for i=1:1:row

for j=1:1:col

if sob\_Grad(i,j)<=thold

sob\_Edge(i,j)=0;

else

sob\_Edge(i,j)=255;

end

end

end

x\_img=uint8(x\_img);

y\_img=uint8(y\_img);

sob\_Grad=uint8(sob\_Grad);

sob\_Edge=uint8(sob\_Edge);

end

lab3sobel.m

function [x\_img,y\_img,prewGrad,prewEdge] = lab3prewitt(img,thold)

[row,col,chNum]=size(img);

if (chNum==3)

img=rgb2gray(img);

end

img=double(img);

xfilter=[-1 0 1;-1 0 1;-1 0 1];

yfilter=[-1 -1 -1;0 0 0;1 1 1];

x\_img=conv2(img, xfilter);

y\_img=conv2(img, yfilter);

prewGrad=sqrt(x\_img.^2+y\_img.^2);

prewEdge=zeros(size(img));

for i=1:1:row

for j=1:1:col

if prewGrad(i,j)<=thold

prewEdge(i,j)=0;

else

prewEdge(i,j)=255;

end

end

end

x\_img=uint8(x\_img);

y\_img=uint8(y\_img);

prewGrad=uint8(prewGrad);

prewEdge=uint8(prewEdge);

end

lab3prewitt.m

function[E] = lab3log(img,Th1,Th2)

[row,col,chNum]=size(img);

if (chNum==3)

img=rgb2gray(img);

end

img=double(img);

J=lab2gaussfilt(img);

kernel=[0 1 0; 1 -4 1; 0 1 0];

E=zeros(size(img));

G = zeros(size(img));

G=conv2(J,kernel,'same');

k=1

% for i = k+1:1:row-k-1

% for j = k+1:1:col-k-1

% Window = J(i-k:i+k,j-k:j+k);

% value = sum(dot(Window,kernel));

% G(i,j) = value;

% end

% end

for i=k+1:1:row-k-1

for j=k+1:1:col-k-1

if (abs(G(i,j))>=Th1)

if ( (G(i,j)\*G(i+1,j)<0) || (G(i,j)\*G(i-1,j))<0) || (G(i,j)\*G(i,j+1)<0) || (G(i,j)\*G(i,j-1)<0 )

if( (abs(G(i+1,j)-G(i,j))>=Th2) || (abs(G(i-1,j)-G(i,j))>=Th2) || (abs(G(i,j+1)-G(i,j))>=Th2) || (abs(G(i,j-1)-G(i,j))>=Th2))

E(i,j)=255;

end

end

else

if ( (G(i+1,j)\*G(i-1,j)<0) || (G(i,j+1)\*G(i,j-1)<0) )

if( (abs(G(i+1,j)-G(i-1,j))/2>=Th2) || (abs(G(i,j+1)-G(i,j-1))/2>=Th2) )

E(i,j)=255;

end

end

end

end

end

for(x=1:1:100)

gp(x) = G(80,x) ;

end

img=uint8(img);

G=uint8(G);

E=uint8(E);

figure

subplot(1,3,1);

imshow(img);

title('Original Image');

subplot(1,3,2);

imshow(G);

title('LoG Filtered Image');

subplot(1,3,3);

plot(gp);

title('Gradient Profile');

end

lab3log.m

function [Filtered\_img] = lab2gaussfilt(img)

[row,col,ch]=size(img);

if (ch==3)

img = rgb2gray(img);

end

Gaussian\_matrix = (1/273)\*[ 1 4 7 4 1 ;

4 16 26 16 4 ;

7 26 41 26 7 ;

4 16 26 16 4 ;

1 4 7 4 1 ];

Filtered\_img = zeros(size(img));

img = double(img);

k = 2;

for i = k+1:1:row-k-1

for j = k+1:1:col-k-1

Window = img(i-k:i+k,j-k:j+k);

value = sum(sum(Window.\*Gaussian\_matrix));

Filtered\_img(i,j) = value;

end

end

%method 2

% Filtered\_img = conv2(img,Gaussian\_matrix,'full');

%

img = uint8(img);

Filtered\_img = uint8(Filtered\_img);

end

lab2gaussfilt.m