Practical 2 Part 2 Edge Detection

February 9, 2021

Jaya Parmar, MATLAB filename part2_edgedetection.m

1 Introduction

Our aim is to apply Canny Edge Detection to extract horizontal edges of the staircase from the given image.

2 Method

We read in the staircase.jpg image first,

$$im = imread('stairs.jpg') \tag{1}$$

Then we convert the color image into a grayscale image as below

$$im = rgb2gray(im) \tag{2}$$

$$figure(1), imshow(im) \tag{3}$$

The gray image is shown in Figure 1 in results section.

We will start applying the Canny algorithm steps to the gray image now.

Image is first convolved with a Gaussian filter with 5X5 size and standard deviation 2

$$G = fspecial('gaussian', [55], 2) \tag{4}$$

$$imG = conv2(im, G, 'same') \tag{5}$$

Next, this images passed through two Sobel filters, one in horizontal direction 'h' and other in vertical directioon 'g'.

$$h = [-1\ -2\ -1\ ; 0\ 0\ 0; 1\ 2\ 1]; \tag{6}$$

$$g = [-1\ 0\ 1;\ -2\ 0\ 2;\ -1\ 0\ 1]; \tag{7}$$

and the masked results are stored in imGx and imGy variables

$$imGx = conv2(imG, h, 'same') \tag{8}$$

$$imGy = conv2(imG, g, 'same') \tag{9}$$

Then we find the magnitude and angle image using

$$grad_{im} = abs(imGx) + abs(imGy) \tag{10}$$

$$grad_{angle} = atan(imGy./imGx) \tag{11}$$

The angle image is in radians at the moment, we need to convert it to degrees by

$$grad_{deg} = floor(grad_{angle}/(pi/180)) \tag{12}$$

This yields angle between -90 to +90 degrees but we want them between 0 and +180 degrees. We convert the degrees with below three lines of code

$$min_{angle} = min(grad_{angle(:)}) \tag{13}$$

$$if min_{angle} < 0 \tag{14}$$

$$grad_{deg} = grad_{deg} + 180.0 \tag{15}$$

end

To start with quantization we need to first find the size of $grad_{deg}$ and create an empty matrix out_{angle} of that size,

$$[m, n] = size(grad_{deg}) \tag{16}$$

$$out_{angle} = zeros(m, n) \tag{17}$$

The angle image $grad_{deg}$ has edges in 4 directions, i.e. horizontal,vertical and two diagnol edges.
We will quantize out_{angle} to 0 for $grad_{deg}$ between o to 22.5, 157.5 to 202.5 or 337.5 to 360. This will extract only the horizontal edges.

$$for\, x = 2 : m - 1 \tag{18}$$

$$for\, y = 2 : n - 1 \tag{19}$$

if

$$(0 < grad_{deg(x,y)} \& grad_{deg(x,y)} <= 22.5) \tag{20}$$

||

$$(157.5 < grad_{deg(x,y)} \& grad_{deg(x,y)} <= 202.5) \tag{21}$$

||

$$(337.5 < grad_{deg(x,y)} \& grad_{deg(x,y)} <= 360) \tag{22}$$

then

$$out_{angle(x,y)} = 0 \tag{23}$$

Next we perform non-maximum suppression. To save the results, first we create an empty matrix out_{nonmax}

$$out_{nonmax} = zeros(m, n) \tag{24}$$

Again, to extract the horizontal edges we will compare the point to the point in the point in its north and south directions.
If the intensity of the edge point is greater than the north-south direction, this point is on horizontal edge.

$$for\, x = 2 : m - 1 \tag{25}$$

$$for\, y = 2 : n - 1 \tag{26}$$

if

$$grad_{im(x,y)} >= grad_{im(x,y-1)} \&\& grad_{im(x,y)} >= grad_{im(x,y+1)} \tag{27}$$

then

$$out_{nonmax} = grad_{im(x,y)} \tag{28}$$

See Figure 2 in the results section, generated with below command.

$$figure(2), imshow(out_{nonmax}); \tag{29}$$

Figure 2 contains information of all horizontal edge pixels. We need to clean out the image such that only the stair edges are seen. This will be done by hysteresis thresholding.

Hysteresis thresholding requires a low and high threshold. All pixels above high threshold will be set as 255 and all pixels below low threshold will be set as 0.

With trial and error we get the best results with below threshold values,

$$thresh_{low} = 0.08 * max(max(out_{nonmax})) \tag{30}$$

$$thresh_{high} = 0.2 * max(max(out_{nonmax})) \tag{31}$$

$$for\, x = 2 : m - 1 \tag{32}$$

$$for\, y = 2 : n - 1 \tag{33}$$

if

$$out_{nonmax(x,y)} < thresh_{low} \tag{34}$$

then

$$out_{nonmax} = 0 \tag{35}$$

elseif

$$out_{nonmax(x,y)} > thresh_{high} \tag{36}$$

then

$$out_{nonmax} = 255 \tag{37}$$

elseif

$$out_{nonmax(x,y+1)} > thresh_{high} || out_{nonmax(x,y-1)} > thresh_{high} \tag{38}$$

then

$$out_{nonmax} = 255 \tag{39}$$

else

$$out_{nonmax} = 0 \tag{40}$$

Finally, we convert the 'double' image into 'uint8' image since the grayscale image is 'uint8'.

$$final = uint8(out_{nonmax}) \tag{41}$$

$$figure(3), imshow(final) \tag{42}$$

Figure 3 shows the final image with horizontal edges of the staircase.

3 Results

Figure 1

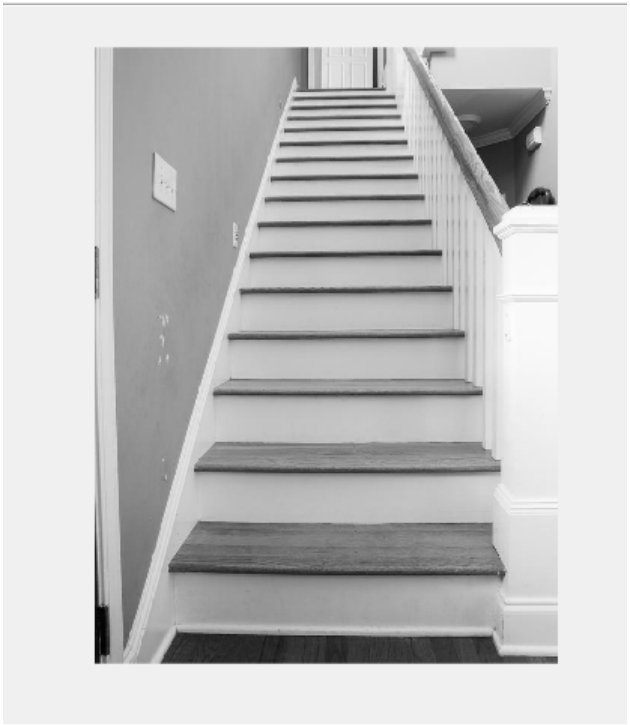


Figure 2

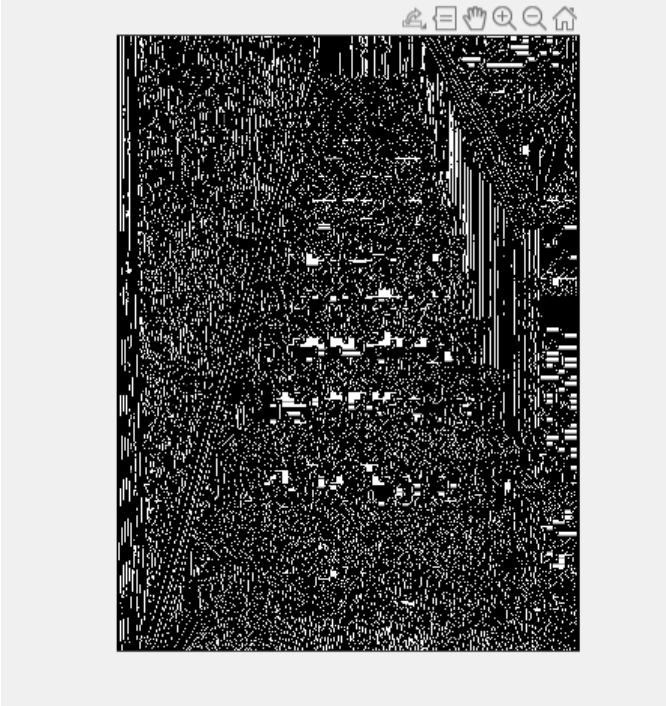
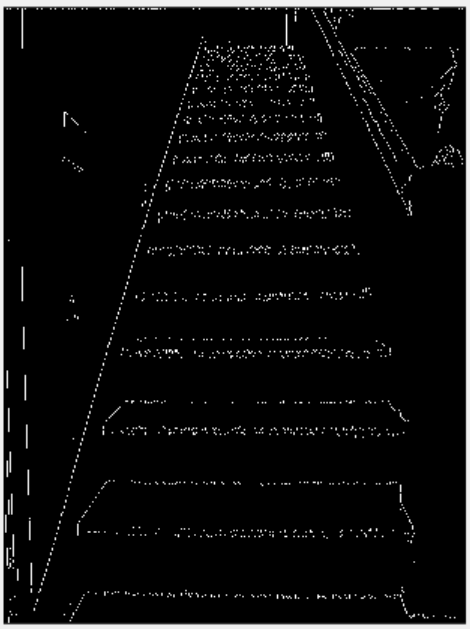


Figure 3



4 Discussion

The final result shows some of the edge pixels as non-edge pixels. The results can be improvised by either changing the threshold values or using further enhancement algorithms. It could also happen that the original color image has noise in one or more of its channels leading to the results we achieved.

5 Appendix

```
set(0, 'defaulttextinterpreter','Latex');

im=imread('stairs.jpg');
im=rgb2gray(im);
figure(1),imshow(im);

G=fspecial('gaussian',[5 5],2);
imG = conv2(im,G,'same'); %Low pass filtering

g = [-1 0 1;-2 0 2;-1 0 1];
h = [-1 -2 -1;0 0 0;1 2 1];
```

```
imGx = conv2(imG,h,'same');    %Edge detection
imGy = conv2(imG,g,'same');

grad_im = abs(imGx)+abs(imGy);   %Magnitude image
grad_angle = atan(imGy./imGx);   %Angle image (radian)

grad_deg = floor(grad_angle/(pi/180));  %Angle image(
   degrees)

min_angle = min(grad_angle(:));   %from -90-->+90 to
   0-->180
if min_angle<0
    grad_deg=grad_deg+180.0;
end

[m,n] = size(grad_deg);
out_angle = zeros(m,n);

for x=2:m-1   %Quantization (convert 0-->360 to 0-->3)
    for y=2:n-1
        if(0<grad_deg(x,y)&grad_deg(x,y)<=22.5)
           |(157.5<grad_deg(x,y)&grad_deg(x,y)<=202.5)
           |(337.5<grad_deg(x,y)&grad_deg(x,y)<=360)
            out_angle(x,y)=0;
        end
    end
end

out_non_max=zeros(m,n); %Non-maximum suppression (
   Horizontal edge is out_angle=0)
for x = 2:m-1
for y = 2:n-1
    if(grad_im(x,y)>=grad_im(x,y+1)&&grad_im(x,y)>=
       grad_im(x,y-1))
       out_non_max(x,y)=grad_im(x,y);
    end
end
end
figure(2), imshow(out_non_max);

thresh_low=0.08*max(max(out_non_max));
thresh_high=0.2*max(max(out_non_max));
for x = 2:m-1        %Threshold hysteresis operation
    for y = 2:n-1
        if(out_non_max(x,y)<thresh_low)
```

```
            out_non_max(x,y)=0;
        elseif(out_non_max(x,y)>thresh_high)
            out_non_max(x,y)=255;
        elseif(out_non_max(x,y+1)>thresh_high||
           out_non_max(x,y-1)>thresh_high)
            out_non_max(x,y)=255;
        else
            out_non_max(x,y)=0;
        end
    end
end

final = uint8(out_non_max);
figure(3),imshow(final)
```