NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

CZ4003: Computer Vision

Lab 1: Edge Detection + Hough Transform + Pixel Intensity SSD and 3D Stereo Vision

Lin HuangJiayin

U1721208D

Contents

1 Edge Detection

1.1 Display ‘macritchie.jpg’ in grayscale

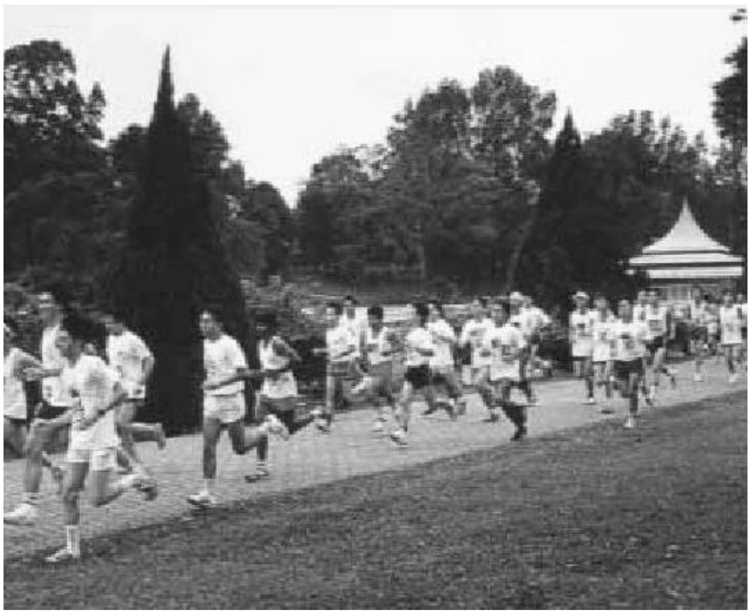
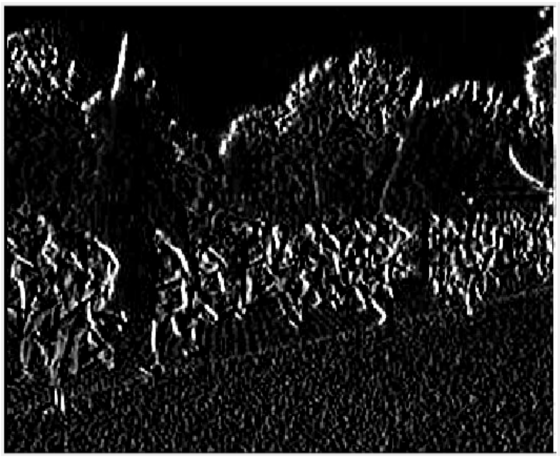


Figure 1. ‘macritchie.jpg’ in grey scale

1.2 Create a 3x3 Sobel masks and filter the image using conv2

```
H1=[-1 0 1; -2 0 2; -1 0 1];
H2 = [-1 -2 -1; 0 0 0; 1 2 1];
GyP = conv2(P,H1);
figure, imshow(uint8(GyP))
GxP = conv2(P,H2);
figure, imshow(uint8(GxP))
```

A horizontal filter is used to detect vertical edges , and a vertical filter is used to detect horizontal edges. Hence, H1 = horizontal filter, H2 = vertical filter.



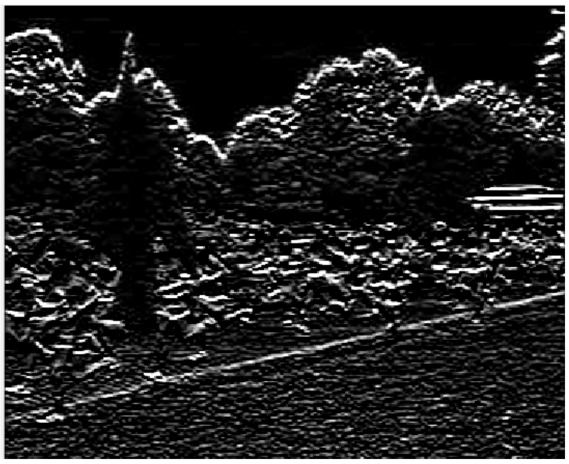


Figure 2. Image after horizontal filter

Figure 3. Image after vertical filter

Non-horizontal and non-vertical edges have been removed

1.3 Generate a combined edge image

```
GxyP = GxP.^2 + GyP.^2;
figure, imshow(uint8(GxyP))
```

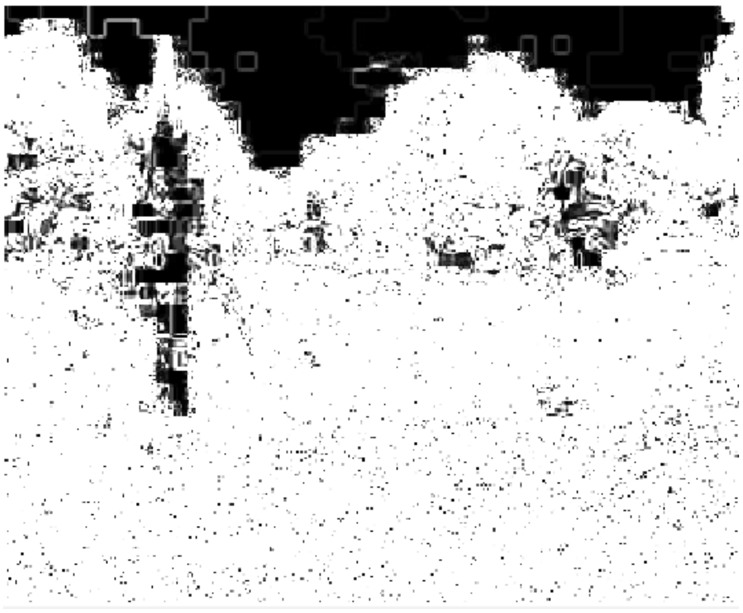


Figure 4. Combine edge image

A squaring operation is to obtain the magnitude of gradient since each image is a vector of the partial derivates in the x and y direction.

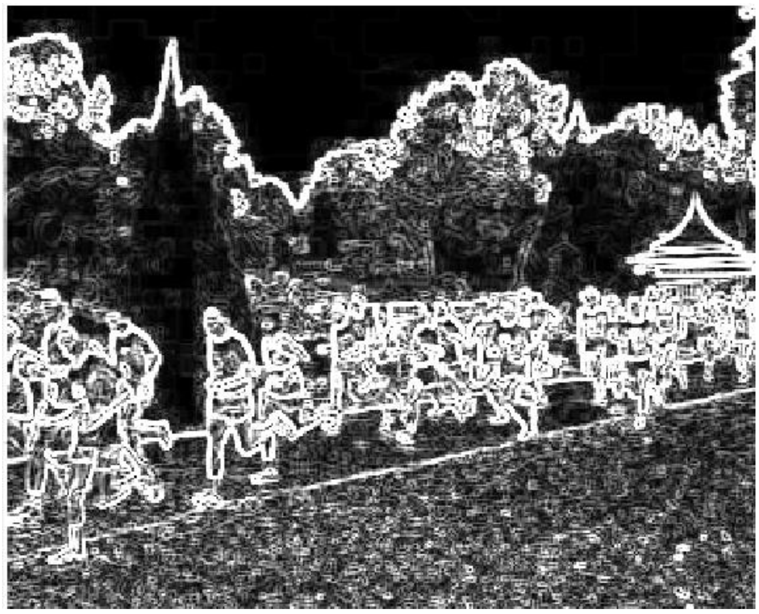


Figure 5. Absolute gradient magnitude

Following the lecture note's method of for absolute gradient magnitude, the above figure (5) can be obtained.

1.4 Threshold the edge image E at value t by Et = E >t

The combined edge image underwent the following thresholding: 10, 100, 1000, 10000, 100000 and 1000000

```
G10 = GxyP>10;
figure, imshow(G10)
% 2 0
G100 = GxyP>100;
figure, imshow(G100)
% 3 0
G1000 = GxyP>1000;
figure, imshow(G1000)
% 4 0
G10000 = GxyP>10000;
figure, imshow(G10000)
% 5 0
G100000 = GxyP>100000;
figure, imshow(G100000)
```

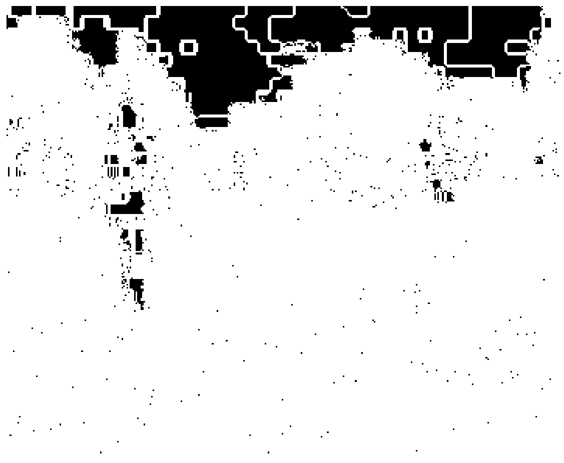


Figure 6. threshold of 10

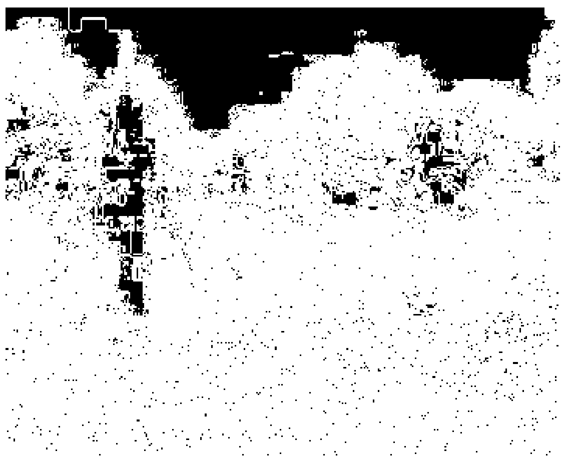


Figure 7. threshold of 100

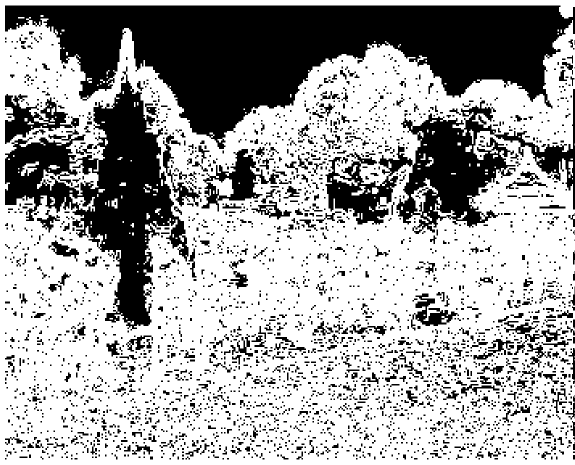


Figure 8. threshold of 1000

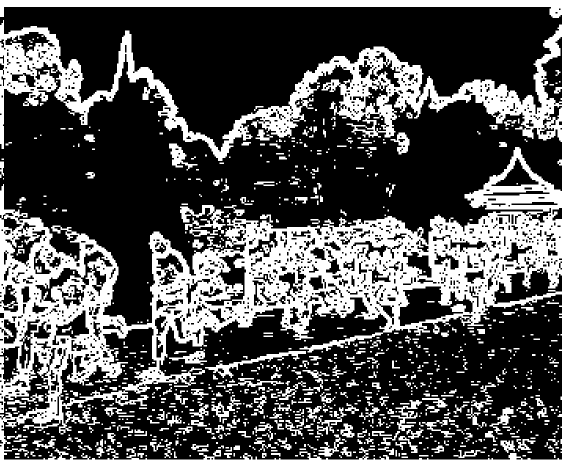


Figure 9. threshold of 10000

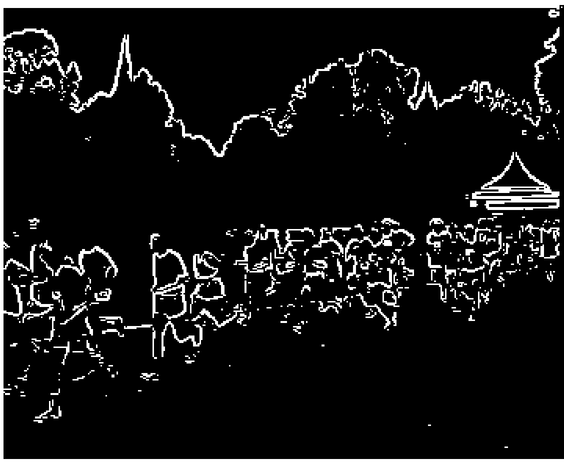


Figure 9. threshold of 100000 Figure 10. threshold of 1000000

Advantages of using different thresholds:

- Able to find an acceptable balance between edge and noise
    - Small threshold: large noise, unable to differentiate edges
    - Large threshold: loss of edges

Disadvantages of using different thresholds:

- “Dirty” method of doing edge detection
    - A lot of trial error is required to find a balance between noise and edge

1.5 Recompute the edge image using Canny edge detection algorithm

```
tl = 0.04;
th = 0.1;
sigma = 1.0;
Gedge = edge(P,'canny',[tl th], sigma);
figure, imshow(Gedge)
```

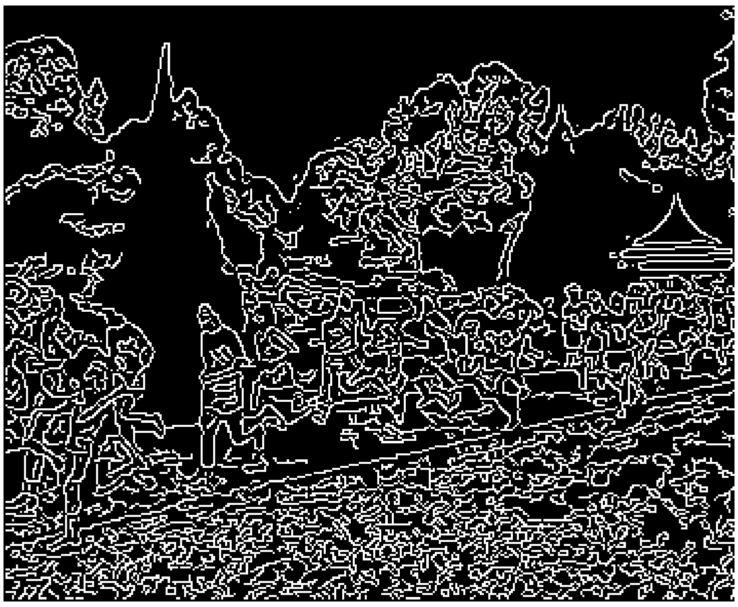


Figure 11. Canny edge detection algorithm

1.6 Different sigma values for Canny edge detection algorithm

```
Gedge = edge(P,'canny',[tl th], 2.0);
figure, imshow(Gedge)

Gedge = edge(P,'canny',[tl th], 3.0);
figure, imshow(Gedge)

Gedge = edge(P,'canny',[tl th], 4.0);
figure, imshow(Gedge)

Gedge = edge(P,'canny',[tl th], 5.0);
figure, imshow(Gedge)
```

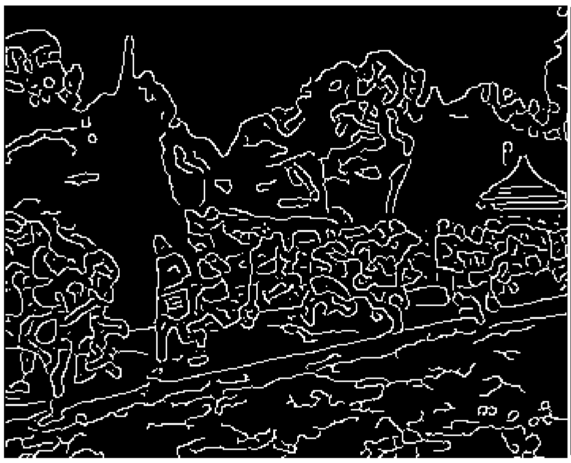


Figure 12. Sigma 2.0

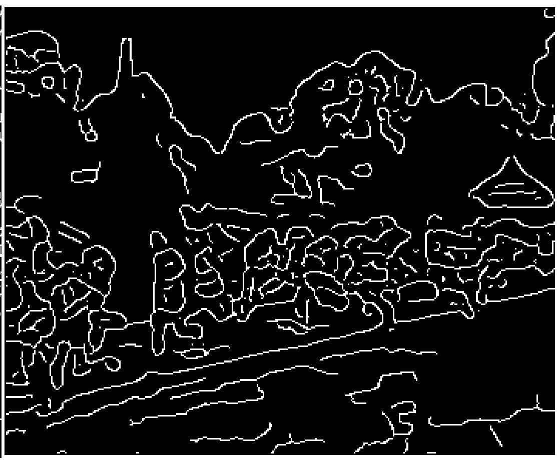


Figure 13. Sigma 3.0

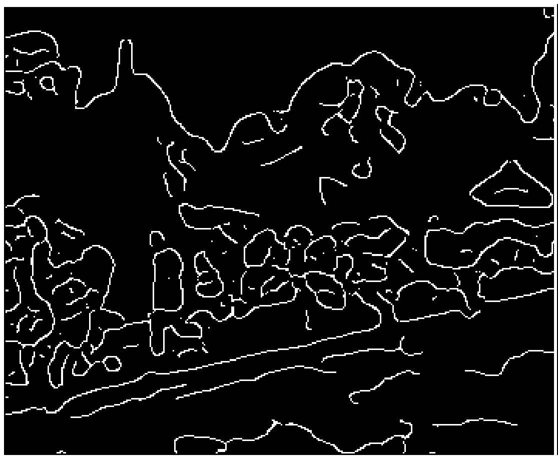


Figure 14. Sigma 4.0

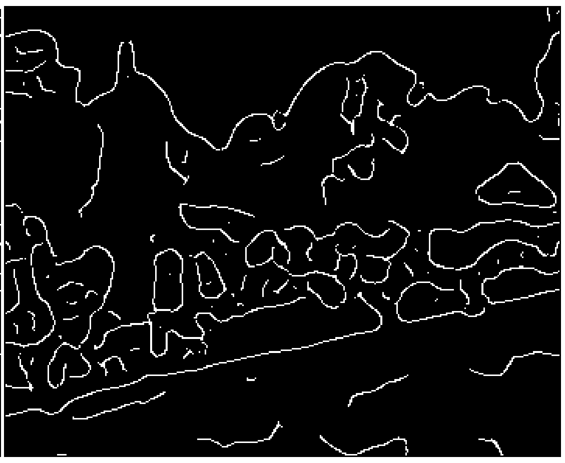


Figure 15. Sigma 5.0

A **lower** sigma is better for location accuracy of edgels .A **higher** sigma is better for noisy edgel removal. Hence, as sigma increases, details decreases.

1.6 Different tl values for Canny edge detection algorithm

```
Gedge = edge(P,'canny',[0.01 th], 1.0);
figure, imshow(Gedge)

Gedge = edge(P,'canny',[0.02 th], 1.0);
figure, imshow(Gedge)

Gedge = edge(P,'canny',[0.04 th], 1.0);
figure, imshow(Gedge)

Gedge = edge(P,'canny',[0.06 th], 1.0);
figure, imshow(Gedge)

Gedge = edge(P,'canny',[0.08 th], 1.0);
figure, imshow(Gedge)

Gedge = edge(P,'canny',[0.09 th], 1.0);
figure, imshow(Gedge)
```

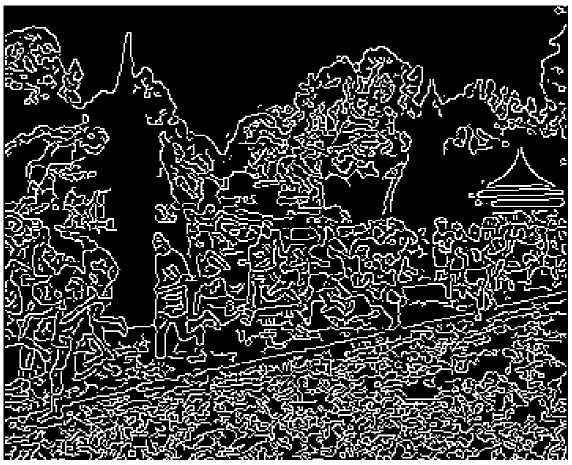


Figure 16. tl = 0.01

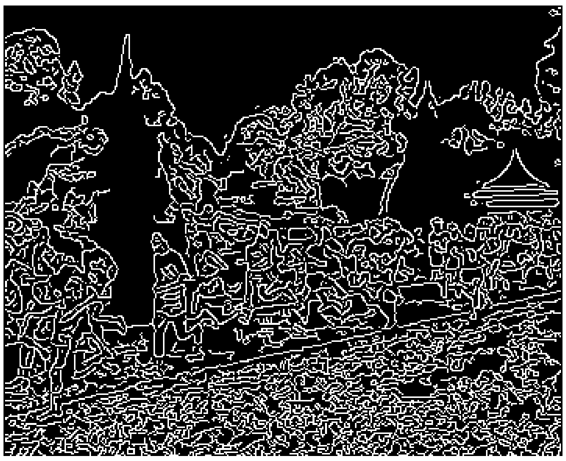


Figure 17. tl = 0.02

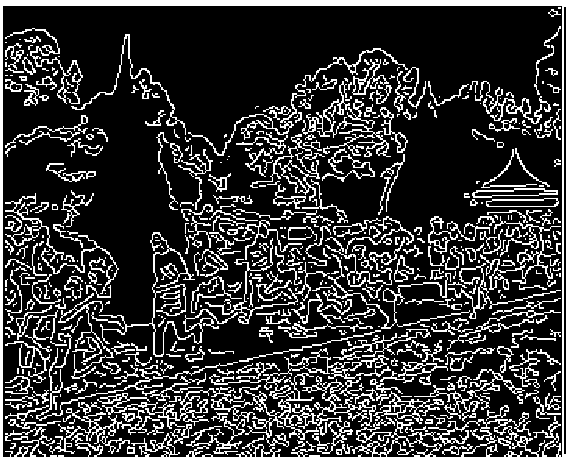


Figure 18. tl = 0.04

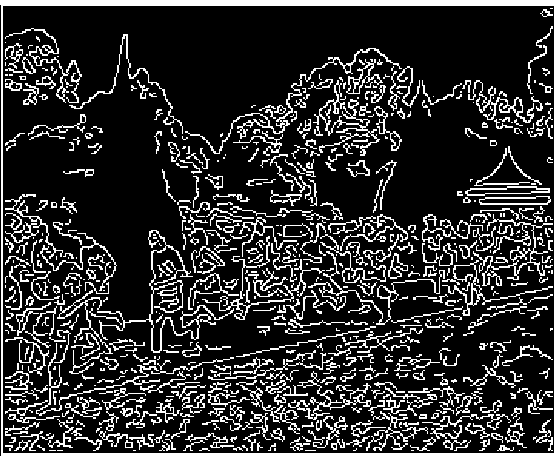


Figure 19. tl = 0.06

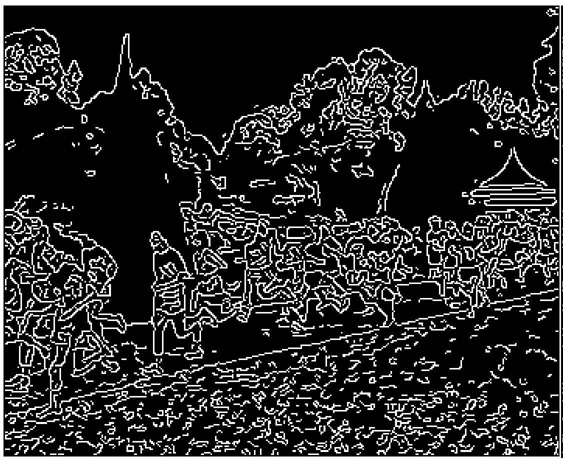


Figure 20. tl 0.8

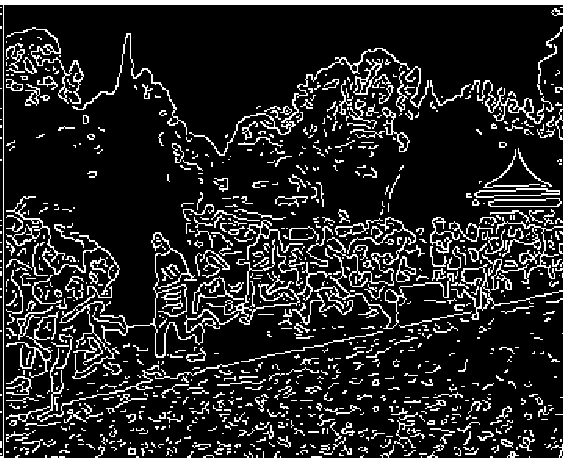


Figure 21. tl 0.9

A **lower** tl is better for location accuracy of edgels .A **higher** tl is better for noisy edgel removal. Hence, as sigma increases, details decreases. The decrease in details works the same as for sigma, however, the rate of decreases is lower for tl. In addition, the value of tl cannot be larger than th.

For hysteresis thresholding, there is a lower threshold (tl) and higher threshold (th). The loss of details is minimum as a significant edgels has a value higher than th. When tl increases, there are areas (patches) that removed together, and it fits the neighbour pixels condition.

2 Line Finding using Hough Transform

2.1 Radon transform

```
help radon
[H,xp] = radon(Gedge);
figure, imshow(uint8(H))
```

```
 radon Radon transform.
    The radon function computes the Radon transform, which is the
    projection of the image intensity along a radial line oriented at a
    specific angle.

    R = radon(I,THETA) returns the Radon transform of the intensity image I
    for the angle THETA degrees. If THETA is a scalar, the result R is a
    column vector containing the Radon transform for THETA degrees. If
    THETA is a vector, then R is a matrix in which each column is the Radon
    transform for one of the angles in THETA. If you omit THETA, it
    defaults to 0:179.

    [R,Xp] = radon(...) returns a vector Xp containing the radial
    coordinates corresponding to each row of R.
```

The radon transform will map each pixel into equivalent sinusoidal function, which is the equivalent to Hough transformation. However, this is since the current application is of discrete value (greyscale). When the image is not of binary format, radon transform will no longer act the same as Hough transform.

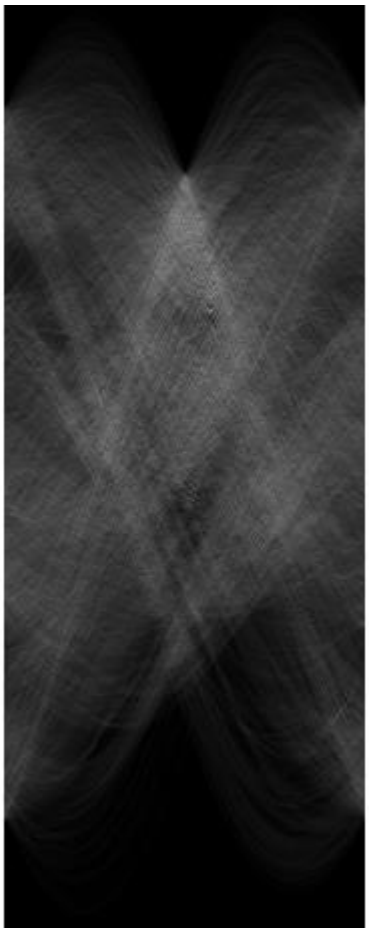


Figure 22. H

2.2 Location of maximum pixel intensity

```
[radius, theta]= find(H ==max(H(:)))
```

```
radius = 157
theta = 104
```

2.3 Derive the equations to convert the line representation to the equation in image coordinates

```
  Name         Size               Bytes   Class     Attributes

  P          290x358              103820  uint8
```

```
radius = xp(radius)
[A B] = pol2cart(theta*pi/180, radius);
B = -B
C = A*(A+179) + B*(B+145)

% 3.2 e
xl = 0;
whos P
xr = 357;
```

2.4 Superimpose the estimated line

```
yl = (C-A *xl)/ B;
yr = (C-A * xr)/B;

imshow(P)
line([xl xr], [yl yr]);
```

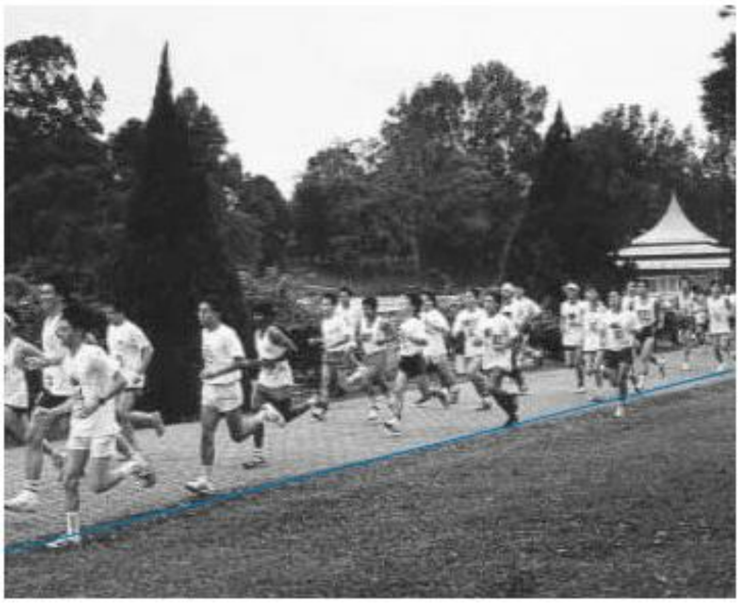


Figure 23. Superimpose of estimated line

The line is not a perfect fit towards the edge. On the left side of the image, the line is below the edge. On the right side of the image, the line is above the edge.

Source of Errors:

- The edge might not be linear. Which means that a linear function will not be able to match it accurately.
- Using radon might have resulted in a change in accuracy

Improvements:

- Increasing the significant values

3. 3D Stereo

3.1 Disparity map algorithm

```
function disparity = algo(left,right,x,y)
h_x = floor(x/2);
h_y = floor(y/2);

[x1,y1]=size(left);
%disparity = zeros(x1,y1);
disparity = ones(x1-(x-1),y1-(y-1));

for i = 1+h_x:x1-h_x
    for j = 1+h_y:y1-h_y
        T = rot90(left(i - h_x : i+h_x, j-h_y: j+h_y),2);
        s1 = conv2(right(i - h_x: i +h_x,:).^2,ones(x,y), 'same');
        s2 = sum(sum(T.^2));
        s3 = 2*conv2(right(i-h_x: i+h_x,:),T,'same');
        S = s1 + s2 - s3;
        min_S = min(S(h_y+1,:));
        xr = find(S(h_y+1,:) == min_S,1);
        disparity(i,j ) =j-xr;
    end
end
```

3.2 Convert to grey scale

```
Lc = imread('corridorl.jpg');
L = im2double(rgb2gray(Lc));
figure, imshow(L);

Rc = imread('corridorr.jpg');
R = im2double(rgb2gray(Rc));
figure, imshow(R);
```

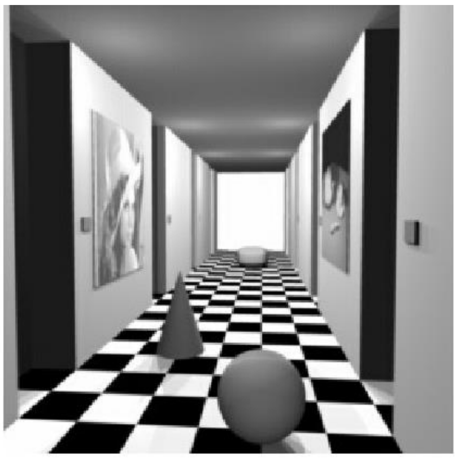


Figure 24. ‘corridorl.jpg’

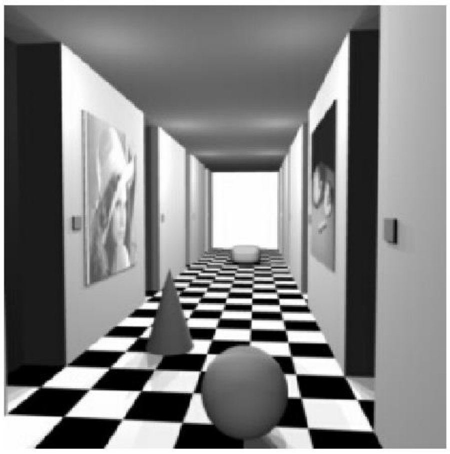


Figure25. ‘corridorr.jpg’

3.3 imshow(-D,[-15 15])

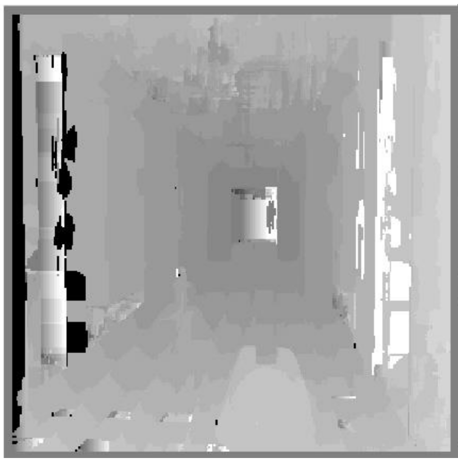


Figure 26. Disparity map D

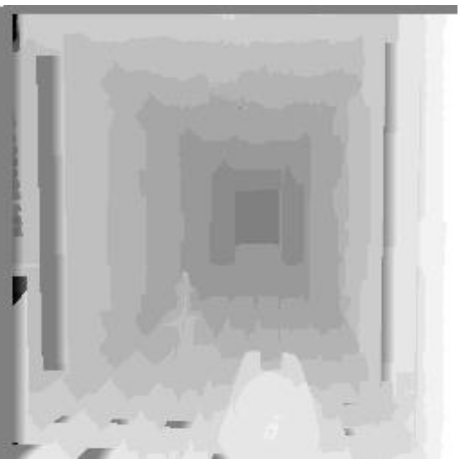


Figure 27. ‘corridor_disp.jpg’

The image obtained is similar to “corridor_disp.jpg”. The nearer points are brighter, and the further points are darker. The color variation winthin the image is consistent too.

However, the image has a significant amount of noise, specially on the top region, and the centre of image, where it should be a flat dark color.

3.4 Rerun algorithm on ‘triclops-i2l.jpg’ and ‘triclops-i2r.jpg’

Figure 28. Disparity map on triclops

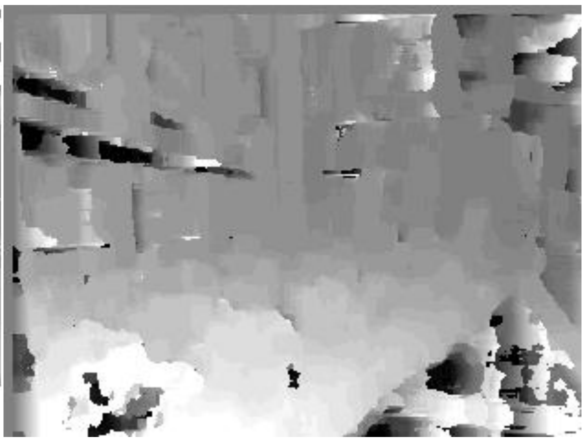


Figure 29. ‘tricolops-id.jpg’

The algorithm works better on triclops than on the corridor, as the obtained image is closer to the given image.

The algorithm works better on images of less contrast

Appendix

files

- “q1&2.m” : Question 1 and 2
- “algo.m”: Question 3.1
- “q3.m”: Question 3