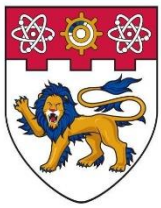


NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

AY2021 Semester 1

Computer Vision

Lab 2

Title:

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

Name: Raymond Toh

Matric Number: U1821599H

Tutor: Lu Shi Jian

Experiments Result

3.1 Edge Detection

a) Convert the image to grayscale. Display the image.

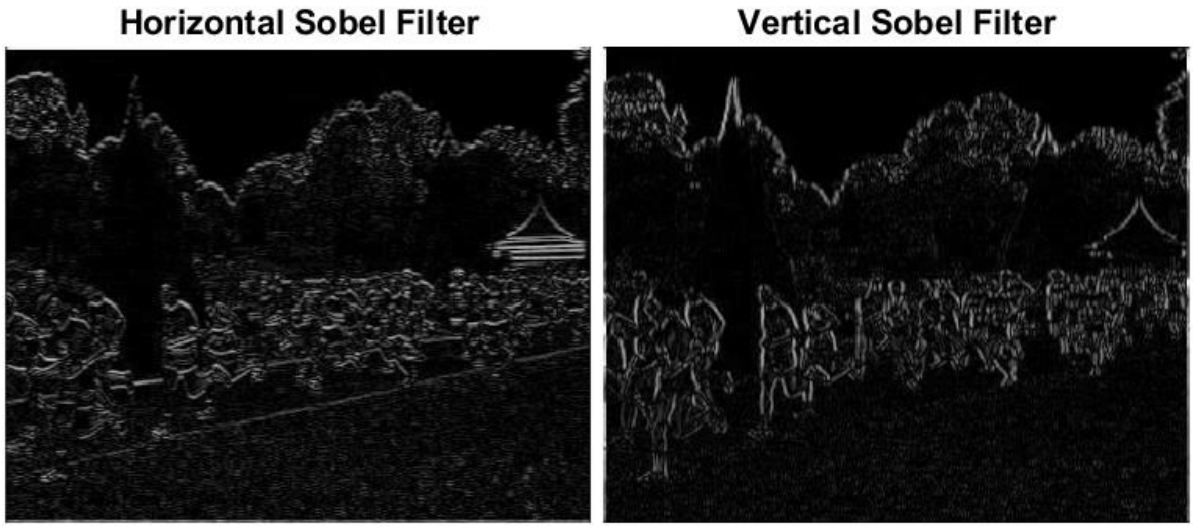
```
%Load img
Pc = imread('macritchie.jpg');

%Check for RGB or Gray_scale
whos Pc

%Convert to grayscale
Pc = rgb2gray(Pc);

%Show img
imshow(Pc, []);
```

b) Create 3x3 horizontal and vertical Sobel masks and filter the image using conv2. Display the edge-filtered images.

```
%Create hori and verti sobel mask
horiSobel = [-1 -2 -1;
              0  0  0;
              1  2  1];

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

%convolution img with sobel mask
img = conv2(double(Pc),double(horiSobel));
imghori = abs(img)/4;

imgverti = conv2(double(Pc),double(vertiSobel));
imgverti = abs(imgverti)/4;

%show result
figure;imshow(imghori, []);title('Horizontal Sobel Filter')
figure;imshow(imgverti, []);title('Vertical Sobel Filter')
```

What happens to edges which are not strictly vertical nor horizontal, i.e. diagonal?

Results:

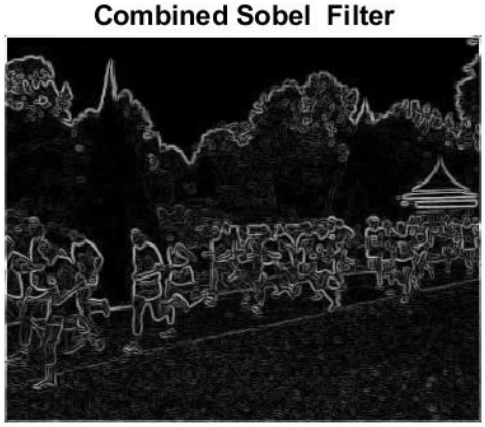


For diagonal edges, horizontal filter seem to detect more compared to vertical as we can see on the track line. However, if we want to detect diagonal edges with sobel operators, we can create the kernel [[0,1,2],[-1,0,1],[-2,-1,0]].

c) Generate a combined edge image by squaring the horizontal and vertical edge images and adding the squared images. Suggest a reason why a squaring operation is carried out.

```
%squaring operation
imgedge = sqrt(imghori.^2+imgverti.^2);
figure;imshow(imgedge, []);title('Combined Sobel  Filter')
```

Results:

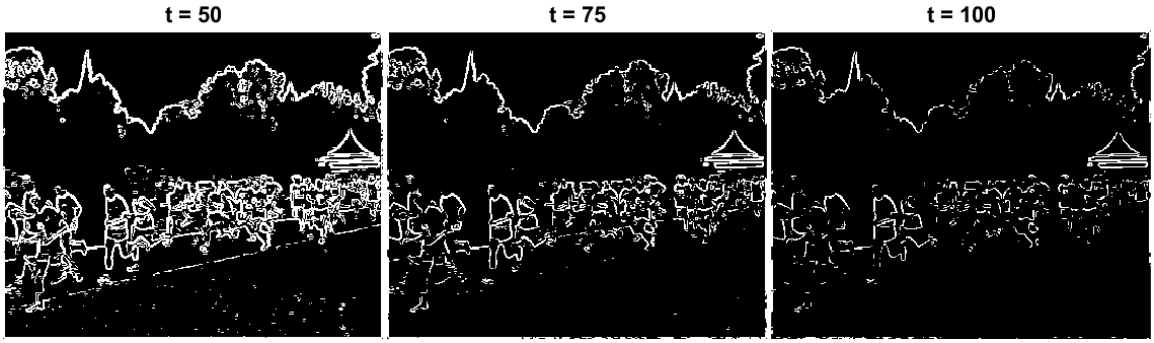


This squaring operation remove negative values and squareroot it, it will return us the magnitude of the gradient.

d) Try different threshold values and display the binary edge images. What are the advantages and disadvantages of using different thresholds?

```
%threshold for imgedge
threshold100 = imgedge>100;
threshold75 = imgedge>75;
threshold50 = imgedge>50;

%different refults for threshold
figure;imshow(threshold100, []);title('t = 100')
figure;imshow(threshold75, []);title('t = 75')
figure;imshow(threshold50, []);title('t = 50')
```

Results:



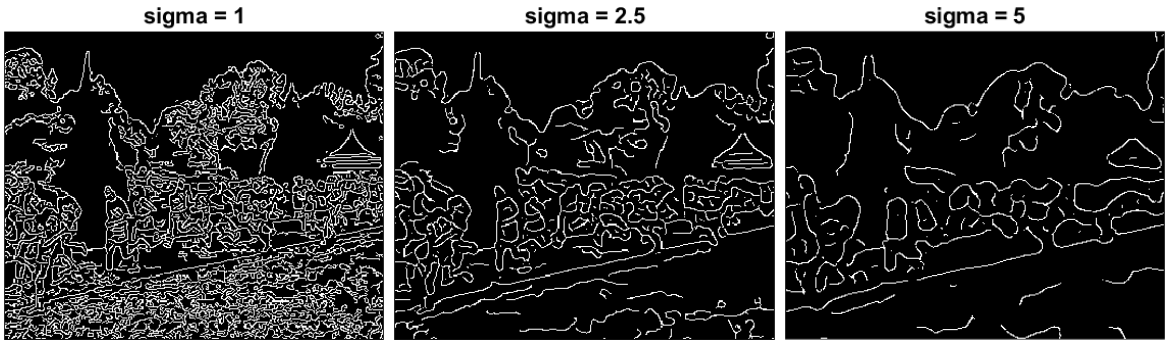
After applying threshold to the edge image, it will turn into a binary image. For example, by setting threshold = 50, whatever below or equal to 50 will be intensity 0 (black) and above 50 will be intensity 255 (white). Hence, the lower threshold we set, it will have stronger edges but in exchange of more noises as we can see from the image above.

e) Recompute the edge image using the more advanced Canny edge detection algorithm with tl = 0.04, th = 0.1, σ = 1.0

i. Try different values of sigma ranging from 1.0 to 5.0 and determine the effect on the edge images. What do you see and can you give an explanation for why this occurs? Discuss how different sigma are suitable for noisy edgel removal, and location accuracy of edgels.

```
%Use Canny edge detection with tl=0.04, th=0.1, sigma=1.0, 2.5, 5
canny1 = edge(Pc, 'canny', [0.04 0.1], 1.0);
canny2 = edge(Pc, 'canny', [0.04 0.1], 2.5);
canny3 = edge(Pc, 'canny', [0.04 0.1], 5.0);

%show results
figure;imshow(canny1, []);title('sigma = 1')
figure;imshow(canny2, []);title('sigma = 2.5')
figure;imshow(canny3, []);title('sigma = 5')
```

Results:

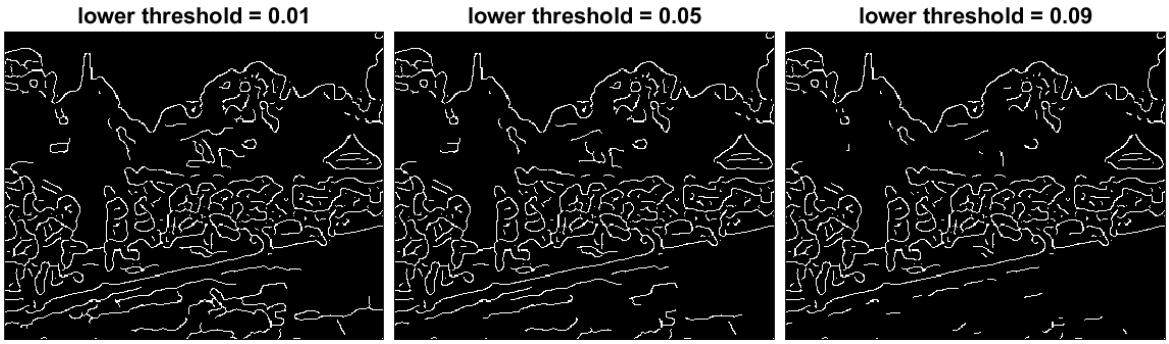


In the first stage of canny edge detection algorithms, it uses gaussian filter. The sigma used in gaussian filter directly affect the results of canny algorithms. It represent the level of details that are simplified. The lower the sigma, it causes less blurring, and allow detection of small and sharps lines (location accuracyof edgels). However, the higher in sigma, it detect larger and smoother edges which also result in less unwanted noises (noisy edgel removal).

ii. Try raising and lowering the value of tl. What does this do? How does this relate to your knowledge of the Canny algorithm?

```
%Change values of TL for canny detection with fix value sigma = 3
canny4 = edge(Pc, 'canny', [0.01 0.1], 3.0);
canny5 = edge(Pc, 'canny', [0.05 0.1], 3.0);
canny6 = edge(Pc, 'canny', [0.09 0.1], 3.0);
```

Results:



In the last stage of canny edge detection algorithms, it uses hystersis thresholding, which is able to set a low threshold tl and a high threshold th. Edge pixel lower than low threshold will be discarded and edge pixel higher than high threshold will be considered as edges if it also met its condition. Hence, in above image, we are able to see lower threshold = 0.01 show more edge pixels as compared to lower threshold = 0.09.

3.2 Line Finding using Hough Transform

a) Reuse the edge image computed via the Canny algorithm with sigma=1.0.

```
canny = edge(Pc, 'canny', [0.04 0.1], 1.0);
figure;imshow(canny, []);
```

Results:

b) The Radon transform, which for binary images is equivalent to the Hough transform. Explain why the transforms are equivalent in this case. When are they different?

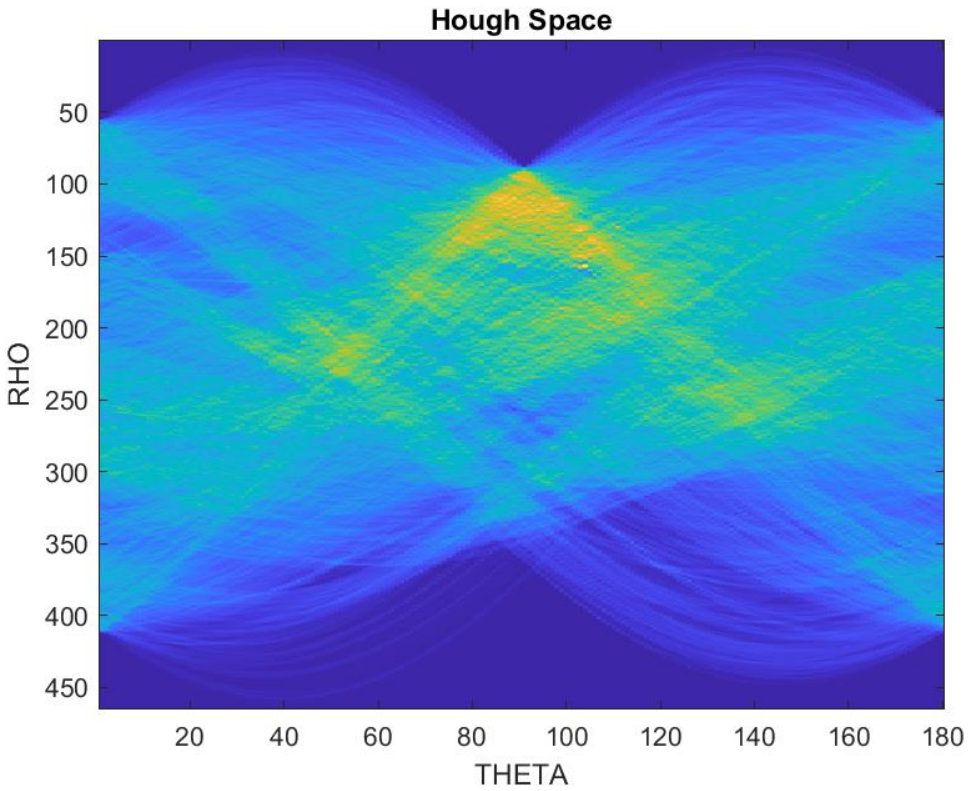
As mention earlier in this report, binary image only consist of vectors 1s and 0s. Hough transform usually perform after normal or hysteresis thresholding which will output an binary image. In Radon transform, if image was not a binary image, intensity will be represented from 0 to 255, which will be considered during transformation.

Additional info: Professor, I found hough transform function in MATLAB here.
https://www.mathworks.com/help/images/ref/hough.html

Display H as an image

```
[Hough, xp] = radon(canny);
%show hough space result
figure;imagesc(uint8(Hough));title('Hough Space');
xlabel('THETA');
ylabel('RHO');
```

Results:



c) **Find the location of the maximum pixel intensity in the Hough image in the form of [theta, radius]**

```
%Get max intensity
maxpeak = max(Hough, [], 'all');
[rho, theta] = find(Hough >= maxpeak);
```

Results:

```
rho = 157
theta = 104
```

d) **Derive the equations to convert the [theta, radius] line representation to the normal line equation form Ax + By = C in image coordinates. Find C.**

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

Results:

```
A = 1.838606e+01
B = 7.374248e+01
C = 1.975976e+04
```

e) **Based on the equation of the line Ax+By = C that you obtained, compute yl and yr values for corresponding xl = 0 and xr = width of image - 1.**

Pc Size: 290x358

```
xl = 0;
yl = (C-A*xl)/B;
xr = 358-1;
yr = (C-A*xr)/B;
```

Results:

```
yl = 2.679563e+02
yr = 1.789463e+02
```

f) Display the original 'macritchie.jpg' image. Superimpose your estimated line. Does the line match up with the edge of the running path? What are, if any, sources of errors? Can you suggest ways of improving the estimation?

```
figure;imshow(Pc, []);
line([xl xr], [yl yr], 'Color', 'blue');
```

Results:

Yes. The line have a good estimated matched up with the edge of running path. We can see at the end of the track, the line was off by abit. One way to improve the estimation was to improve the quality of the image. As edge detector will be able to capture more distinct edges which will then improve the quality of the estimation.

3.3 3D Stereo

Write the disparity map algorithm as a MATLAB function script which takes two arguments of left and right images, and 2 arguments specifying the template dimensions. It should return the disparity map.

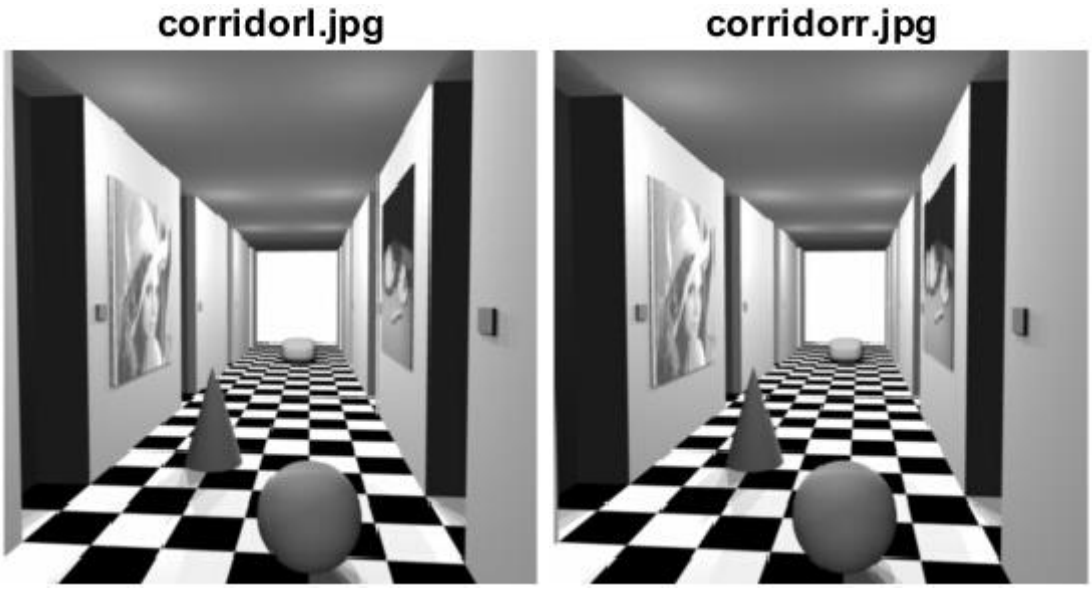
a) **Write the disparity map algorithm as a MATLAB function script which takes two arguments of left and right images, and 2 arguments specifying the template dimensions. It should return the disparity map. Try and minimize the use of for loops, instead relying on the vector / matrix processing functions.**

```
function map = dmap(imgleft, imgright, tempheight, tempwidth)
[height, width] = size(imgleft);

map = ones(height - tempheight + 1, width - tempwidth + 1);
imgstart = round(tempwidth / 2);
th = floor(tempheight / 2);
tw = floor(tempwidth / 2);
for row = imgstart:height - imgstart - 1
    for col = imgstart:width - imgstart - 1

        T = imgleft(row-th:row+tw,col-th:col+tw);
        left = col-14;
        right = col;
        if left < imgstart
            left = imgstart;
        end
        ssd_min = Inf;
        xr_min = left;
        for xr = left:right
            I = imgright(row-th:row+tw,xr-th:xr+tw);
            I_flipped = rot90(I, 2);
            ssd_1 = ifft2(fft2(I) .* fft2(I_flipped));
            ssd_1 = ssd_1(tempheight, tempwidth);
            ssd_2 = ifft2(fft2(T) .* fft2(I_flipped));
            ssd_2 = ssd_2(tempheight, tempwidth) * 2;
            ssd = ssd_1 - ssd_2;
            if ssd < ssd_min
                ssd_min = ssd;
                xr_min = xr;
            end
        end
        map(row, col) = col - xr_min;
    end
end
```

b) Download the synthetic stereo pair images of 'corridorl.jpg' and 'corridorr.jpg', converting both to grayscale.

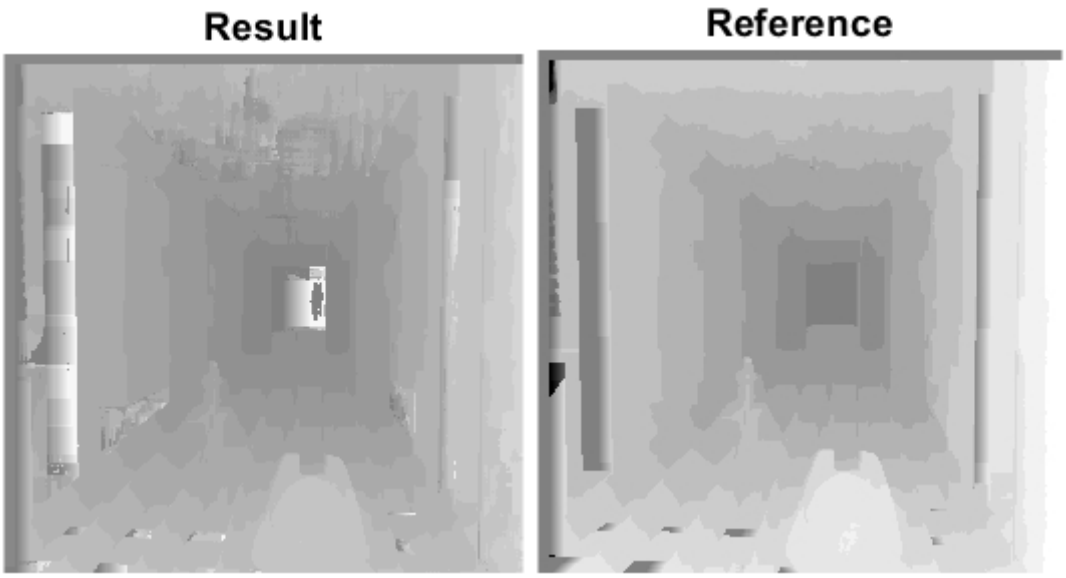
```
left = imread('corridorl.jpg');
left = rgb2gray(left);
right = imread('corridorr.jpg');
right = rgb2gray(right);

figure;imshow(left, []);title('corridorl.jpg')
figure;imshow(right, []);title('corridorr.jpg')
```

Results:



c) Run your algorithm on the two images to obtain a disparity map D, and see the results.

```
D = dmap(left, right, 11, 11);
ref = imread('corridor_disp.jpg');
figure;imshow(D, [-15 15]);title('Result')
figure;imshow(ref, []);title('Reference')
```

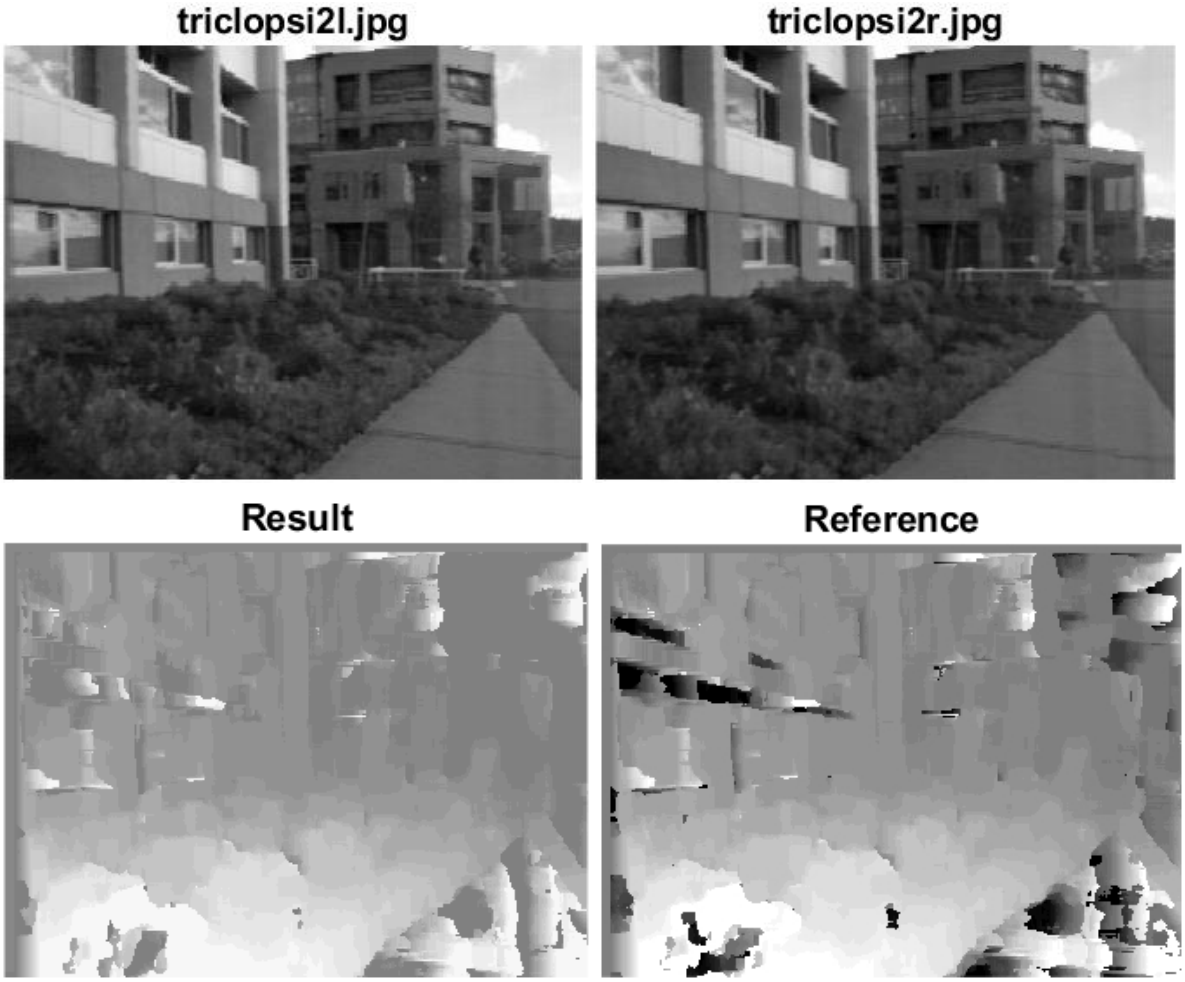


From above image, we can see that our results is much similar to the reference image. However, the wall at the end of original image are different due to the same intensity of neighbouring pixel which also causes our result to be high intensity as compared to reference image.

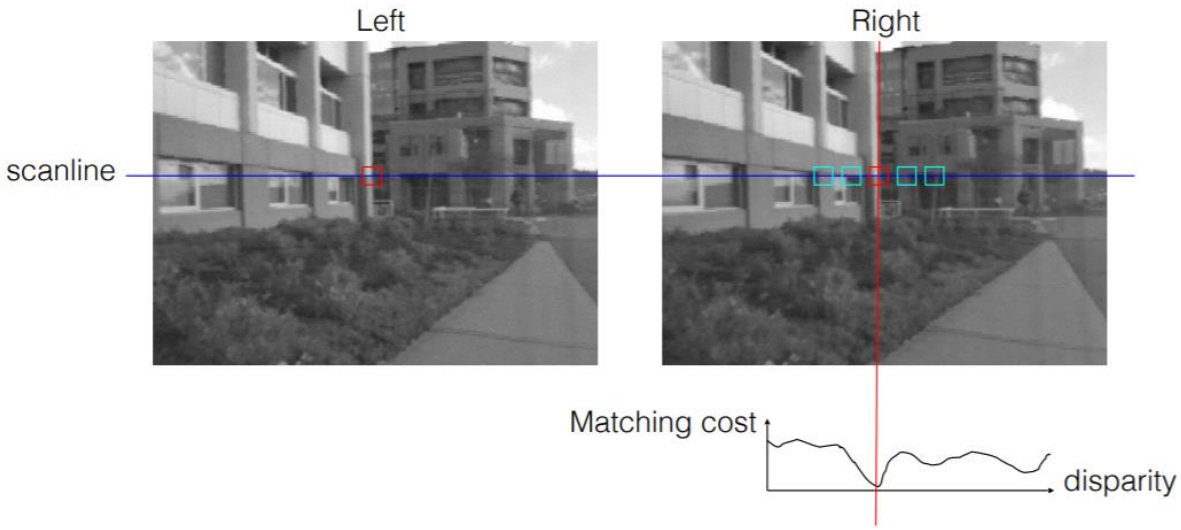
d) Rerun your algorithm on the real images of 'triclops-i2l.jpg' and triclops-i2r.jpg'. Again you may refer to 'triclops-id.jpg' for expected quality. How does the image structure of the stereo images affect the accuracy of the estimated disparities?

```
left2 = imread('triclopsi2l.jpg');
left2 = rgb2gray(left2);
right2 = imread('triclopsi2r.jpg');
right2 = rgb2gray(right2);
figure;imshow(left2, []);title('triclopsi2l.jpg')
figure;imshow(right2, []);title('triclopsi2r.jpg')
```

Results:



The result of not as appealing as previous examples as the matching cost was not a smooth curve.



To improve this, we can use a smaller window size to gain more details of the image. However, the downside of having smaller window size is it will introduce noises.