

3D SCENE ANALYSIS ASSIGNMENT 3

Nivin Pradeep Kumar

S4035593

University of Groningen

Cendur Vishal Karthik Khanna

S3878732

University of Groningen

ABSTRACT

Perspective projection is a linear projection where three dimensional objects are projected on a picture plane. This has the effect that distant objects appear smaller than nearer objects. The lines which are parallel in nature (i.e., meet at the point at infinity) appear to intersect in the projected image. [1]

Index Terms— Central or perspective projection, Vanishing point, Direction of 3D parallel lines

1. INTRODUCTION

In this assignment we determine the orientation of a set of two or more parallel 3D lines from a perspective image of these lines. Furthermore assume that both the direction vector $\vec{\omega}$ of the parallel lines, and the camera constant f is unknown, but that the available information is extended by giving the perspective projection of a rectangle (or a number of rectangles with parallel sides).

2. EXERCISE 1

In this exercise we tried to obtain the camera constant f from the perspective projection of a rectangle, and from that, the direction vectors of the sides of the rectangle. We assume it to be a camera-centered coordinate system and consider u_∞, v_∞ for projection 1 and x_∞, y_∞ for projection 2 to be vanishing point.

$$\begin{pmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{pmatrix} = \frac{1}{\sqrt{u_\infty^2 + v_\infty^2 + f^2}} \begin{pmatrix} u_\infty \\ v_\infty \\ f \end{pmatrix} \quad (1)$$

$$\begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} = \frac{1}{\sqrt{x_\infty^2 + y_\infty^2 + f^2}} \begin{pmatrix} x_\infty \\ y_\infty \\ f \end{pmatrix} \quad (2)$$

Since the sides of the rectangular are perpendicular,

$$\vec{\omega} \cdot \vec{a} = 0 \quad (3)$$

Therefore the equation becomes,

$$\frac{1}{\sqrt{u_\infty^2 + v_\infty^2 + f^2} \sqrt{x_\infty^2 + y_\infty^2 + f^2}} [u_\infty x_\infty + v_\infty y_\infty + f^2] = 0 \quad (4)$$

$$f = \sqrt{-(u_\infty x_\infty + v_\infty y_\infty)} \quad (5)$$

3. EXERCISE 2

In this exercise we process a picture taken by camera containing a number of concentric rectangles with parallel sides using MATLAB and save two sets of parallel lines.

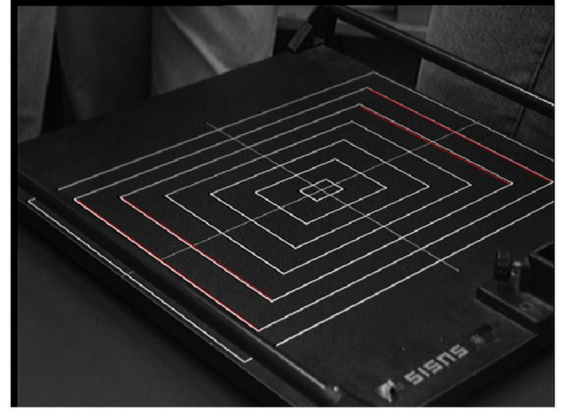


Fig. 1. First sets of parallel lines

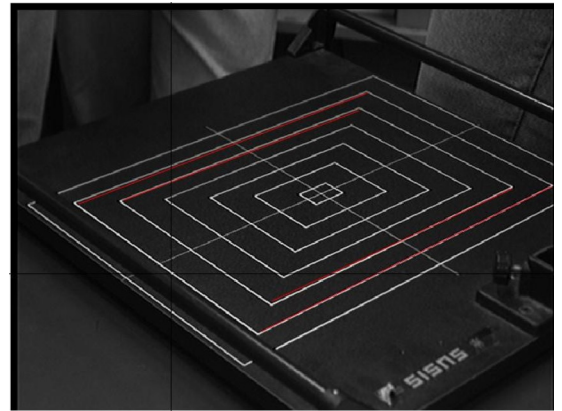


Fig. 2. Second set of parallel lines

The Figure 1 and 2 shows the two sets of parallel lines chosen and saved.

4. EXERCISE 3

In this exercise the matlab function in *par_line.m* determines the orientation of N parallel 3D lines in perspective image. The function reads the file like *parline.dat* which contains coordinates of the endpoints of the images of the parallel lines, this is obtained with the help of plugin called *savescanpoints*. The function after reading the data calls SVD method which returns the singular values of matrix in descending order. The singular values and singular vectors of a matrix is then stored in *u,s,v* respectively. Then we normalize and find the least square solution of the matrix system. We have passed the *parline.dat* which is obtained from the figure 1, which gives us the result as shown in figure 3. The snippet of the code is available in listing 1.



Fig. 3. Output of exercise 3

```
1 function par_line(file1)
2     data = readfile(file1);
3
4     n = 3;
5
6     f = 1.0;           %Temporary value
7
8     m = size(data,1);
9     if m<n; fprintf('M < 3 / error\n'); return; end
10
11     a = zeros(m,n);
12     a(:,1) = data(:,4) - data(:,2);
13     a(:,2) = -(data(:,3) - data(:,1));
14     a(:,3) = data(:,2) .* -a(:,2) - data(:,1) .* a
        (:,1);
15
16     [U,S,V] = svd(a); % call matlab SVD routine
17     v_min = V(:,n); % s and v are already sorted
        from largest to smallest
18     if all(v_min < 0); v_min = -v_min; end % ?
19
20     wvec = [v_min(1)/f v_min(2)/f v_min(3)];
21     wvec = wvec / norm(wvec,2);
22     fprintf('Least squares solution vector %d :
        (%f %f %f)\n',i,wvec );
23
24
25
26
27 function data=readfile(file)
28     f = fopen(file,'r');
29     for i=1:4; fgets(f); end
30     all = fscanf(f,'%f %f %f %f '); m = length(all)
        /4;
31     data= reshape(all,4,m)';
32     fclose(f);
```

Listing 1. Snippet of par_line

5. EXERCISE 4

In this exercise we modify the *par_line.m* to a matlab function called *rectangle.m* which reads two files mainly *par_lines.data* and *par_lines1.data*. The function returns the camera constant *f*, the direction vector of the sides of the rectangles and finally the normal of the planar patch containing the rectangles. In the function we first calculate SVD of the two data in the files. Then we go through the sorted list of vector, since the list of vectors is in descending(Matlab function provides that) we can easily calculate the direction of vector of the side of the rectangle, We calculate the normal of the planar patch by doing the cross product between the set of parallel lines. The code snippet is given in listing 2.

```
1 function rectangle(file1,file2)
2     data = readfile(file1);
3     data1 = readfile(file2);
4
5     n = 3;
6     f = 1.0;
7     %for file1
8     m = size(data,1);
9     if m<n; fprintf('M < 3 / error\n'); return; end
10    a = zeros(m,n);
11    a(:,1) = data(:,4) - data(:,2);
12    a(:,2) = -(data(:,3) - data(:,1));
13    a(:,3) = data(:,2) .* -a(:,2) - data(:,1) .* a
        (:,1);
14
15    [U,S,V] = svd(a); % call matlab SVD routine
16    v_min = V(:,n); % s and v are already sorted
        from largest to smallest
17    if all(v_min < 0); v_min = -v_min; end % ?
18    wvec = [v_min(1)/f v_min(2)/f v_min(3)];
19    wvec = wvec / norm(wvec,2);
20    %for file2
21    a = zeros(m,n);
22    a(:,1) = data1(:,4) - data1(:,2);
23    a(:,2) = -(data1(:,3) - data1(:,1));
24    a(:,3) = data1(:,2) .* -a(:,2) - data1(:,1) .*
        a(:,1);
25
26    [U,S,V] = svd(a); % call matlab SVD routine
27    v_min = V(:,n); % s and v are already sorted
        from largest to smallest
28    if all(v_min < 0); v_min = -v_min; end % ?
29    rvec = [v_min(1)/f v_min(2)/f v_min(3)];
30    rvec = rvec / norm(rvec,2);
31    fprintf('the direction vectors of the side1: (%f
        %f %f)\n',wvec );
32
33    fprintf('the direction vectors of the side2: (%f
        %f %f)\n',rvec );
34
35    fprintf('normal of the planar patch: (%f %f %f)\n',
        cross(wvec,rvec));
36
37
38
39 function data=readfile(file)
40     f = fopen(file,'r');
41     for i=1:4; fgets(f); end
42     all = fscanf(f,'%f %f %f %f '); m = length(all)
        /4;
43     data= reshape(all,4,m)';
44     fclose(f);
```

Listing 2. Snippet of Rectangle.m

The output of which is given in figure 4. We were not able to do the camera constant f but now how the methodology works which we have explained below. First we calculate the equation of the line with a two point formula as shown. Using this formula we have to find the equation for 2 lines.

$$(y - y_1) = \frac{(y_2 - y_1)}{(x_2 - x_1)}(x - x_1) \quad (6)$$

Let m be,

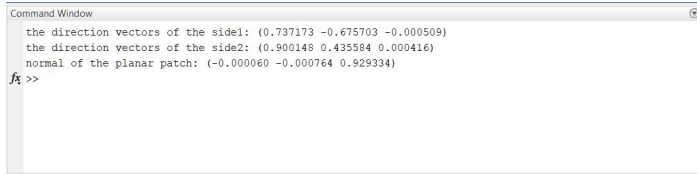
$$m = \frac{(y_2 - y_1)}{(x_2 - x_1)} \quad (7)$$

Then the two equation becomes,

$$y = M_1x - M_1x_1 + y_1 \quad (8)$$

$$y = M_2x - M_2x_3 + y_3 \quad (9)$$

When we solve for x and y in this we get the camera constant f .



```

Command Window
the direction vectors of the side1: (0.737173 -0.675703 -0.000509)
the direction vectors of the side2: (0.900148 0.435584 0.000416)
normal of the planar patch: (-0.000060 -0.000764 0.929334)
fx >>

```

Fig. 4. Output of exercise 4

6. REFERENCES

- [1] R. Szeliski, *Computer Vision Algorithm and Application*. Springer.