University of Girona

Spain

Visual Perception

Lab 1 - Calibration of a simulated camera

Submitted by :
Mr. Nayee Muddin Khan
DOUSAI

1 Introduction

The visual perception lab 1 is focused on calibrating the camera by a transformation matrix by using both hall method and Faugeras-Tuscani method. The projection points has to be computed by generating random points in a given range and then to calibrate accordingly with the given two methods. We have didcided the work into three different parts and the parts are explained respectively in the below sections

2 Part 1

2.1 Step 1 and 2

At first we have to define intrinsic (I) and extrinsic (E) transformation matrices. The matrices are defined as shown below:

$$I = \begin{pmatrix} \alpha_u & 0 & u_0 & 0 \\ 0 & \alpha_v & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}; E = \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

The camera was to be calibrated in $EulerXYX1$. The intrinsic and extrinsic parameters are defined as below:

au=557.0943; av=712.9824; u0=326.3819; v0=298.6679;
f=80 mm.;
Tx=100 mm.; Ty=0 mm.; Tz=1500 mm.;
Phix=0.8∗**pi**/2; Phiy=−1.8∗**pi**/2; Phix1=**pi**/5; Euler XYX1
Image **size**: 640x480

To calculate the extrinsic parameter matrix we have to compute the total rotation in $XYX1$ direction and it is represented as above E matrix. The code used for the computing the matrices and the results are as shown below:

```
I = [au, 0, u0,0;
      0, av, v0, 0;
       0, 0, 1,0];


rotx = [1,0,0;
        0, cos(phix), -sin(phix);
        0, sin(phix),  cos(phix)];

roty = [cos(phix), 0, sin(phix);
        0, 1, 0;
        -sin(phix), 0, cos(phix)];

rotx1 = [cos(phix),-sin(phix),0;
         sin(phix), cos(phix),0;
         0,0,1];

R = rotx*roty*rotx1;
E = [R(1,:),Tx; R(2,:),Ty; R(3,:),Tz; 0, 0,0, 1];
```

```
I =

  557.0943         0  326.3819         0
         0  712.9824  298.6679         0
         0         0    1.0000         0

>> E

E =

   1.0e+03 *

    0.0001   -0.0003    0.0010    0.1000
    0.0006   -0.0008   -0.0003         0
    0.0008    0.0006    0.0001    1.5000
         0         0         0    0.0010
```

Figure 1: Left: (a)Rotation Code; Right: (b)The result matrices

2.2 Steps 3, 4 and 5

For the next following steps we take random six points in the range of $[-480, 480; -480, 480; -480, 480]$. The random points are genered by using rand function from matlab, as it will be easy to generate more random points in the following steps to get 10 and 50 points.The points has to be made homogeneous matrix by adding identity column for the transformation matrix.The randommatrix has to be multiplied with intrinsic (I) and extrinsic (E) transformation matrices, the equation is given as :

$$pmatrix = A * randmatrix; where A = I * E$$

The above pmatrix contains the projected points with three cordinates for the random six points which are normalised into 2D space and the resultant pmatrix is written and ploted as below:

```
pmatrix =

   1.0e+05 *

    4.1022    6.3080    7.2184    4.2297    5.9612    8.1065
    5.2256    0.6953    5.6041    0.3605    2.3386    7.0819
    0.0141    0.0126    0.0199    0.0137    0.0170    0.0161
```

Figure 2: pmatrix result

```
randmatrix = [randi([-480,480],[6,1]),randi([-480,480],[6,1]),randi([-480,480], [6,1])];
randmatrix = randmatrix';
randmatrix(4,:) = 1
figure
scatter (randmatrix(1,:), randmatrix(2,:),randmatrix(3,:));
%step 4
pmatrix = zeros (2,6);
pmatrix = I*E*randmatrix;
scatter (pmatrix(1,:), pmatrix(2,:));
%step 5
% normalize the projected points
projNorm(1,:) = pmatrix(1,:)./pmatrix(3,:);
projNorm(2,:) = pmatrix(2,:)./pmatrix(3,:);

%figure, scatter3(randmatrix(1,:), randmatrix(2,:), randmatrix(3,:), 'bo'), title('3D points');
%figure, scatter(projNorm(1,:), projNorm(2,:), 'bo', 'MarkerFaceColor', 'b'), title('projected 2D points');
```

Figure 3: Matlab code

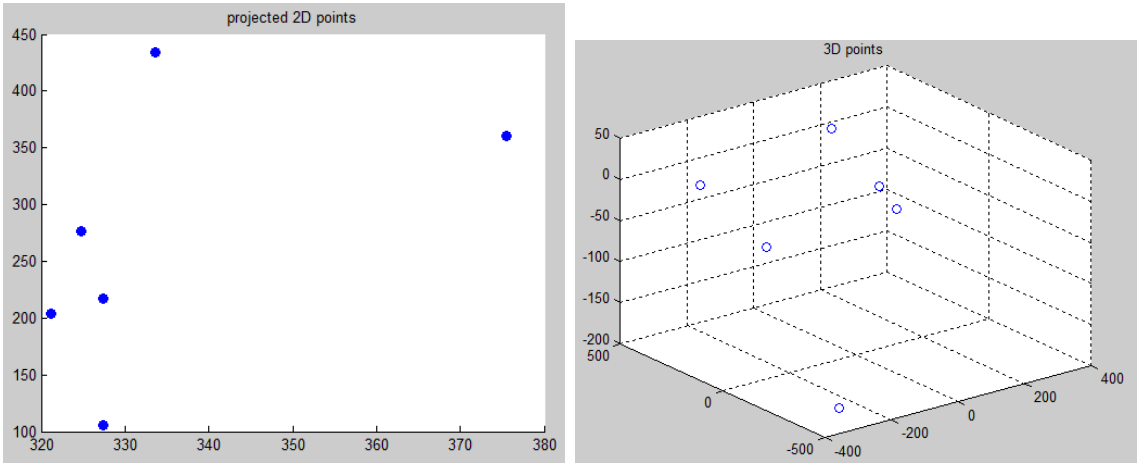

Figure 4: Left: (a)2D points; Right: (b)3D points

2.3 Step 6 and 7

The hallmatrix has to be computed with generated random points and the normalised matrix. The hall matrix is computed as shown below. Pseudo-inverse of matrix Q is computed by below equation

$$A = (Q^t Q)^{-1} Q^t B \tag{1}$$

where Q and B are:

$$Q_{2i-1} = ({}^W P_{wi}^t \; 1 \; 0 \; 0 \; 0 \; 0 \; -{}^I X_{ui} \, {}^W P_{wi}^t);$$

$$Q_{2i} = (0 \; 0 \; 0 \; 0 \; {}^W P_{wi}^t \; 1 \; -{}^I Y_{ui} \, {}^W P_{wi}^t)$$

$$B_{2i-1} = ({}^I X_{ui});$$

$$B_{2i} = ({}^I Y_{ui})$$

Figure 5: Matlab code

The matlab code is shown as below:

```
Q=[];B=[];
for i=1:6

    Q = [Q;randmatrix(1,i) randmatrix(2,i) randmatrix(3,i) 1 0 0 0 0
        -(projNorm(1,i)*randmatrix(1:3,i))'; 0 0 0 0
        randmatrix(1,i) randmatrix(2,i) randmatrix(3,i) 1 -(projNorm(2,i)*randmatrix(1:3,i))' ]
    B = [B; projNorm(1,i); projNorm(2,i);]

end
  A = Q\B;
%step 7
tr = I * E;
tr = tr(:,:)/tr(3, 4);
% Hallmat and tr are the same;
```

Figure 6: Matlab code

According to the step 6 and 7 both the matrices A from the random values and the tr will be same as shown below:

```
A =

    0.2125    0.0156    0.3740  363.5215
    0.4346   -0.2493   -0.1207  298.6679
    0.0005    0.0004    0.0001    1.0000

>> tr

tr =

    0.2125    0.0156    0.3740  363.5215
    0.4346   -0.2493   -0.1207  298.6679
    0.0005    0.0004    0.0001    1.0000
```

Figure 7: Matrix from given values and hall transformation matrix

2.4 Step 8 and 9

In this section we will add the gaussian noise to the points then check the accuracy of the caliberated camera. The below figure the difference between the points with noise and without noise and also the projected points. We have added noide bu using stand deviation of 0.5. Now the accuracy is calculated by the mean of the distance between two points. The mean error is calculated for six points, ten points and 50 points. The mean error was given as 0.000000000945, 0.00000000000004123 and 0.0000000000000000886245 respectively for 6, 10 and 50 points.

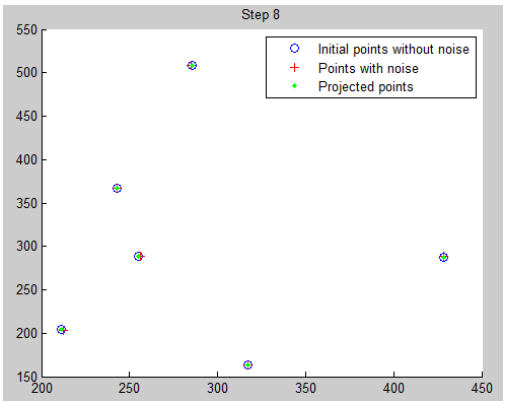

Figure 8: Points with noise and without and the projected points

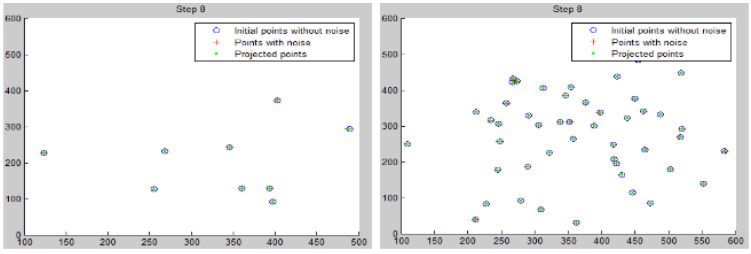

Figure 9: 10 and 50 points projection

3 Part 2

3.1 Step 10

For this step we have to repeat the above work again by implementing for *Faugeras – Tuscani* method. The transformation matrix of this methos is described as:

$$X = (Q^t Q)^{-1} Q^t B = \begin{pmatrix} T_1 \\ T_2 \\ T_3 \\ C_1 \\ C_2 \end{pmatrix} \tag{2}$$

where Q and B are:

$$Q_{2i-1} = ({}^W P_{wi}^t \quad -{}^I X_{ui}\, {}^W P_{wi}^t \; 0\; 0\; 0\; 1\; 0)$$

$$Q_{2i} = (0\; 0\; 0 \; -{}^I Y_{ui}\, {}^W P_{wi}^t \quad {}^W P_{wi}^t \; 0\; 1)$$

$$B_{2i-1} = ({}^I X_{ui})$$

$$B_{2i} = ({}^I Y_{ui})$$

```
Q = zeros(2*pN, 11);
B = zeros(2*pN, 1);
  for i = 1:pN
      %Q(2*i-1, :)
      Q= [Q;randmatrix(i,:) -(projNorm(1,i)*randmatrix(i,:))' 0 0 0 1 0;0 0 0 -(projNorm(2,i)*randmatrix(i,:))' randmatrix(i,:) 0 1];
     % Q(2*i, :) = [0, 0, 0, -projNorm(2,i)*randmatrix(i,:), randmatrix(i,:), 0., 1.];
     % B(2*i-1) =

      B= [B;projNorm(1,i);projNorm(2,i)];

  end;
  X = Q\B

  T1 = X(1:3)';
  T2 = X(4:6)';
  T3 = X(7:9)';
  C1 = X(10);
  C2 = X(11);
  %compute intrinsic parameters:
  normT2 = norm(T2,2)^2;
  U0 = (T1*T2')/normT2;
  V0 = (T2*T3')/normT2;
  Au = norm(cross(T1',T2'))/normT2;
  Av = norm(cross(T2',T3'))/normT2;
  intrinsics = [Au, 0, U0, 0; 0, Av, V0, 0; 0, 0, 1, 0];
```

4 Part 3

In final step we have to simulate the world coordinate system, the camera system, the focal point, the image point and the both 3D points and their projections accordingly with respect to the image plane. The results are plotted in below figure.

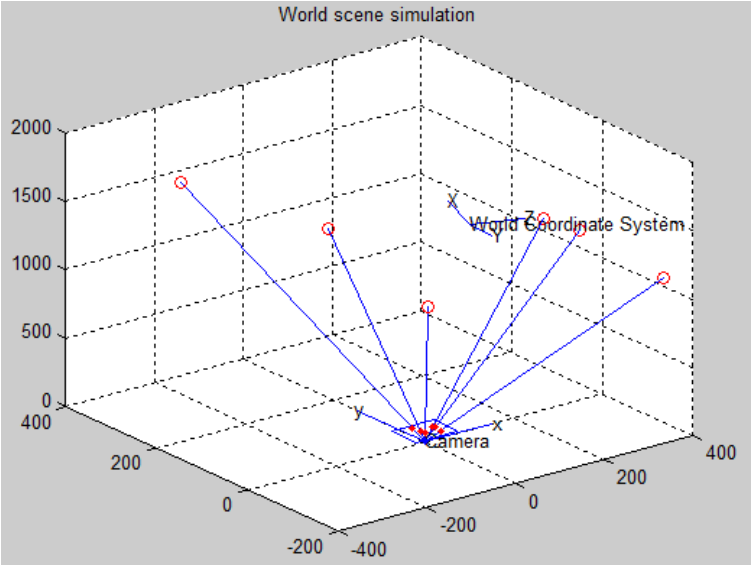

Figure 10: Camera Calibration Simulation

5 Conclusion

In this lab we have calibrated camera by usinh Hall and Fougeras method. The results are labelled and plotted accordingly in every step by using different set of points and also adding gaussian noise with standard deviation. The plotted results are observed and matched with respect to the randomly generated points and the implemented methods.