Students:
Mir Mohibullah Sazid [sazidarnob@gmail.com]
Syma Afsha [symaafsha.eece@gmail.com]

Lab1 : Camera Modelling and Calibration

Introduction

By calibrating a simulated camera, the purpose of these two lab sessions is to gain practical knowledge about basic camera projection geometry. As we work on this task, we will learn:

1. With a set of parameters, build a camera projection matrix.
2. Define 3D points and the projections of matching 2D images.
3. Apply the Hall method for calibration
4. As noise in the picture points grows, assess how accurate this calibration is.

Implementation

0.1 Step 1

Take into account the following values for the extrinsic and intrinsic parameters:

- Intrinsic parameters

au = 557.0943; av = 712.9824; u0 = 326.3819; v0 = 298.6679;

- Location of the world reference frame in camera coordinates in mm

Tx = 100; Ty = 0; Tz = 1500;

- World rotation w.r.t. camera coordinates

Phix = 0.8*pi/2; Phiy = -1.8*pi/2; Phix1 = pi/5;

0.2 Step 2

To begin, the camera intrinsic matrix K and the extrinsic camera transformation cTw are generated using the values specified in step-1. Subsequently, these K and cTw components are employed to calculate the projection matrix P. The resulting K, cTw, and P are depicted in 1 via the Matlab command window.

0.3 Step 3

Six sets of randomly selected 3D points in the interval [-480:480; -480:480; -480:480] have been determined. 2shows the code piece responsible for producing these random spots.

```
Command Window
>> K

K =

  557.0943         0  326.3819
         0  712.9824  298.6679
         0         0    1.0000

>> cTw

cTw =

   1.0e+03 *

    0.0010   -0.0000   -0.0000    0.1000
    0.0000    0.0010   -0.0000         0
    0.0000    0.0000    0.0010    1.5000
         0         0         0    0.0010

>> P

P =

   1.0e+05 *

    0.0057    0.0000    0.0030    5.4528
    0.0002    0.0072    0.0028    4.4800
    0.0000    0.0000    0.0000    0.0150
```

Figure 1: K, cTw and P matrix

```
points_3D=zeros(4,6);
for i=1:3
    points_3D(i,:)=-480+rand(1,6)*(480-(-480));
end
points_3D(4,:)=ones(1,6);
```

Figure 2: Generation of 6 set of random 3D points.

```
>> points_3d

points_3d =

  385.1665 -479.6490  -16.0512 -338.4242 -177.6076 -315.4796
   89.1907  151.3037 -144.9958 -428.4543 -434.7187 -426.3765
 -411.3485 -464.9515  454.7318  225.4049  242.2902 -448.0189
    1.0000    1.0000    1.0000    1.0000    1.0000    1.0000

>> points_2d

points_2d =

  579.2713  130.0597  344.4148  246.2451  299.0510  224.3228
  364.1586  408.7233  241.9733  115.4597  115.9985    7.6449
```

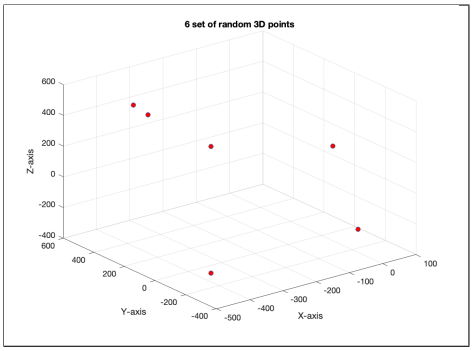
Figure 3: A random instance of 6 set of 3D points and their projected 2D points .

0.4 Step 4

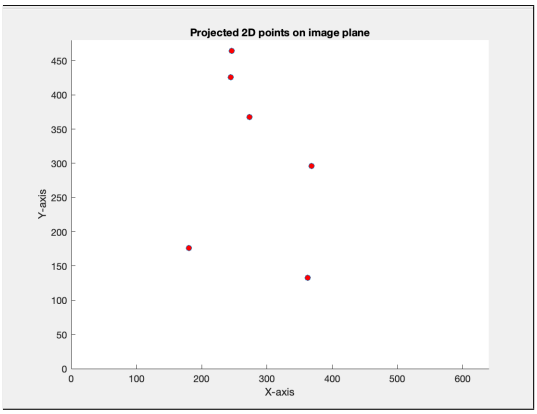
The camera projection matrix P, which is obtained from step-2, is used to calculate the projection of the 3D points onto the image plane. 3 shows an example of a random collection of 3D points and the projected 2D points that correspond to them. Each column of the matrix represents a different point.

0.5 Step 5

Six sets of 3D points and the corresponding 2D points are randomly sampled and graphically represented in Matlab in Figures 4aand 4b



(a) Six set of random 3D points



(b) "Plotted 2D points in the image plane

Figure 4: A random instance of 3D points and their corresponding projected 2D points

0.5.1 Question 1

Will the distribution of points in the image affect the accuracy in the computation? Why? The points are randomly dispersed across the image, and indeed, the arrangement of these points impacts the precision of the calculations. Having a greater number of points leads to an enhanced accuracy of the projection matrix and, consequently, the overall projection quality.

```
function Projection_matrix = get_projection_matrix_from_2D_3D_points_by_Hall_Method(points_2d,points_3d)
sz=size(points_3d);n=sz(1,2);
p3 = points_3d';
Q=zeros(12,11);
j=1;
%calculating Q matrix
for i=1:n
    Q(j,:)=[p3(i,:) zeros(1,4) -points_2d(1,i).*points_3d(1,i) -points_2d(1,i).*points_3d(2,i) -points_2d(1,i).*points_3d(3,i)];
    Q(j+1,:)=[zeros(1,4) p3(i,:) -points_2d(2,i).*points_3d(1,i) -points_2d(2,i).*points_3d(2,i) -points_2d(2,i).*points_3d(3,i)];
    j=j+2;
end
%calculating B matrix
B=zeros(n*2,1);
k=1;
for i=1:n
    B(k)=points_2d(1,i);
    B(k+1)=points_2d(2,i);
    k=k+2;
end
A=Q\B; % A=Inverse(Q).B
A(12)=1;
%Organizing the unknowns in projection matrix
Projection_matrix=zeros(3,4);
t=1;
for i=1:3
    for j=1:4
     Projection_matrix(i,j)=A(t);
     t=t+1;
    end
end
```

Figure 5: Implementation of calculating projection matrix by Hall Method

0.6 Step 6

By employing the randomly generated 3D points from Step 3 along with their corresponding projections, the 3x4 camera projection matrix P is computed using the Hall method. The code section for executing the Hall method can be found in 5

0.7 Step 7

6 visually displays the differences between the projection matrix P, calculated using the Hall method in step-6, and the projection matrix P1, derived from the provided data in step-2. To facilitate a more straightforward comparison, the original projection matrix is normalized by dividing it by the final element and setting it to 1. The observed differences are quite minimal. Subsequently, the initial camera projection matrix P is utilized to extract the intrinsic parameter matrix K and the camera rotation matrix cRw, with these values presented in 7

0.8 Step 8

After that, we introduced Gaussian noise to every individual 2D point to introduce variations. We conducted experiments with different mean and standard deviation values to achieve variations within the range of [-1, +1] for 95% of the points. Eventually, we settled on mean=0 and standard deviation=0.5. The code segment responsible for applying Gaussian noise to the 2D points can be found in 8. Subsequently, we utilized the noisy 2D points and the random 3D points from step 3 to construct the projection matrix P2. From P2, we derived the intrinsic parameter matrix K2 and the camera rotation matrix cRw2. Figures 9 and 10 depict the disparities between the intrinsic parameters and camera rotation obtained from the projection matrix, with and without Gaussian noise. These differences are negligible when calibrating the camera for projecting the noisy points onto the image plane.

0.8.1 Question 1

How did the intrinsic parameters change? The intrinsic parameters underwent minor modifications to enhance the camera calibration for improved projection accuracy, especially when dealing with noisy points. These adjustments typically involve alterations to the focal length, which subsequently affect the intrinsic characteristics.

```
>> P=P/P(3,4)

P =

    0.3817    0.0006    0.1990  363.5215
    0.0146    0.4794    0.1884  298.6679
    0.0000    0.0000    0.0007    1.0000

>> P1

P1 =

    0.3817    0.0006    0.1990  363.5215
    0.0146    0.4794    0.1884  298.6679
    0.0000    0.0000    0.0007    1.0000

>> P-P1

ans =

   1.0e-12 *

    0.0026    0.0088   -0.0083   -0.3411
    0.0077    0.0133   -0.0141    0.9095
    0.0000    0.0000   -0.0000         0
```

Figure 6: Comparison of projection matrix

```
K =

  557.0943         0  326.3819
         0  712.9824  298.6679
         0         0    1.0000

>> cRw

cRw =

    0.9987   -0.0110   -0.0493
    0.0099    0.9997   -0.0219
    0.0496    0.0214    0.9985
```

Figure 7: Extracted intrinsic parameters and camera rotation matrix

```
sz=size(points_2d);
r = normrnd(0,0.5,sz);
Gaussian_2d_points=points_2d+r;
```

Figure 8: Enter Caption

```
K1 =

  557.0943   -0.0000  326.3819
         0  712.9824  298.6679
         0         0    1.0000

>> K2

K2 =

  529.2138  -33.5124  339.5815
         0  672.2071  285.9540
         0         0    1.0000

>> K2-K1

ans =

  -27.8805  -33.5124   13.1996
         0  -40.7753  -12.7139
         0         0         0
```

Figure 9: Comparison of Intrinsic parameter matrix K

```
cRw1 =

    0.9987   -0.0110   -0.0493
    0.0099    0.9997   -0.0219
    0.0496    0.0214    0.9985

>> cRw2

cRw2 =

    0.9994    0.0047   -0.0341
   -0.0040    0.9997    0.0221
    0.0342   -0.0219    0.9992

>> cRw2-cRw1

ans =

    0.0007    0.0157    0.0152
   -0.0139    0.0000    0.0440
   -0.0153   -0.0433    0.0006
```

Figure 10: Comparison of camera rotation matrix cRw

```
function avg_error = get_average_projection_error(old_2d_points,new_2d_points)

sz=size(old_2d_points); n=sz(2);
sum=0;

for i=1:n
    sum=sum+ sqrt((old_2d_points(1,i)-new_2d_points(1,i))^2+(old_2d_points(2,i)-new_2d_points(2,i))^2);
end
avg_error=sum/n;
```

Figure 11: Code segment for computing average projection error.

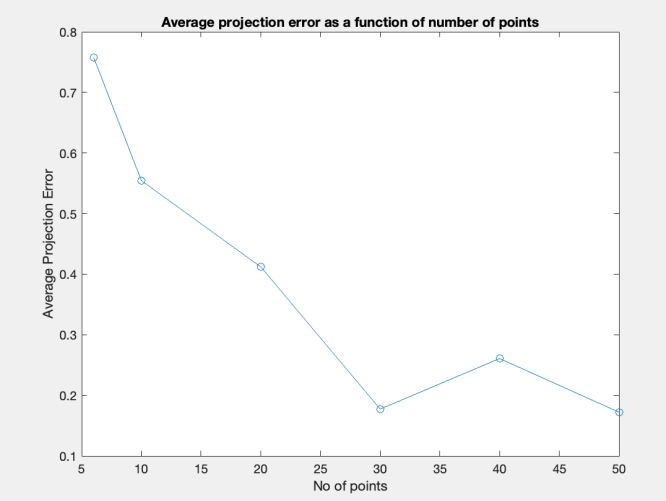


Figure 12: Average projection error vs no of points

0.8.2 Question 2

Why is the axis skew parameter different from zero? The axis skew parameter was initially defined as 0, as illustrated in Figure 2.7. However, it underwent changes because of the camera's skewed coordinate system, where the angle between the camera's x and y axes is not 90 degrees.

0.9 Step 9

In this phase, we computed the 2D points using the projection matrix P2 and compared them with those generated in step-4. The average projection error, defined as the mean of the Euclidean distance between the 2D points from steps 4 and 8, was also calculated. The code segment for determining the average projection error can be found in 11

0.10 Step 10

In the last stage, we increase the number of 3D points to 10, 20, 30, 40, and 50, and then repeat both step-8 and step-9. It becomes evident that as the number of points used increases, the resulting projection matrix becomes more accurate. A graph illustrating the relationship between the average projection error and the number of points is generated and displayed in 12