Camera Calibration - Visual Perception

Daniel Barmaimon, Vibot Master Student, Promotion 8

March 9, 2014

Introduction

The utilization of several cameras and the processing speed of current computers allows the implementation of systems that can manage a big amount of spatial information and that are quite useful for several applications such as the reconstruction of 3D models, backtracking or prediction of object movement, etc.
To achieve these targets it is necessary to know the relation between the reference systems that are going to be used during. The acquisition of the parameters that matches these reference systems is called *camera calibration*.

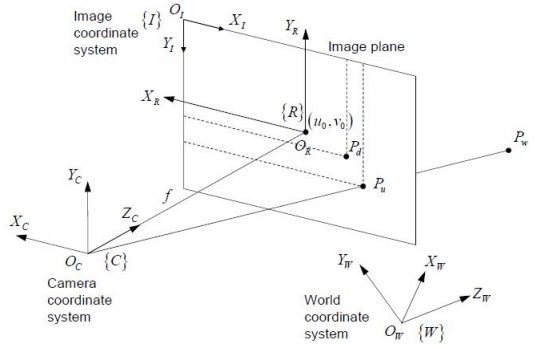

Figure 1: Systems of reference for camera image acquisition

First one will be the *world* coordinate system (W), usually static, and will allow to reference any point that belongs to 3D space.
The *camera* should have its own coordinate system (C), and its origin its called *focal point*. This system allows to reference any point in 3D space, and has a different orientation than the previous one.
Last coordinate system is the one that is related with the sensor plane or *image plane*) of the camera and it will be the one that allows the conversion from pixel to area. The image plane will have its origin in its upper right corner, and this plane will have a minimal distance to the camera coordinate system at the *principal point*, which coordinates are (u0, v0) into the image coordinate system. The orientation of this plane will be perpendicular to z axis of camera reference system.

Part 1 - Hall Method

In this section the target is to generate some projections of random points over the image plane to get the characterization of the camera using Hall's method. Once computed the objective will be to check the accuracy of the method with noisy points and the effect over the estimation of these characterization of increasing the number of points.

Step 1 - Define intrinsic and extrinsic parameters

```
%% Define intrinsec and extrinsic parameters
au = 557.0943; % Relation: f. length and pixel/area
av = 712.9824; % Relation: f. length and pixel/area
u0 = 326.3819; % Principal p. (x, i. plane)
v0 = 298.6679; % Principal p. (y, i. plane)
f = 80;        % Focal length in mm
Tx = 100;      % Traslation in mm
Ty = 0;        % Traslation in mm
Tz = 1500;     % Traslation in mm
Phix = 0.8*pi/2;    % Rotation in rad
Phiy = -1.8*pi/2;   % Rotation in rad
Phix1 = pi/5;       % Rotation in rad
% Euler XYX1    - Reference for rotation
% Image size    - 640 X 480
```

This paragraph represents the introduction of the given data into Matlab code.

Step 2 - Get intrinsic and extrinsic transformation matrices

For the representation of the camera coordinates in term of world coordinates it would be necessary to adjust the orientation and perform a translation. As it is needed to know not only the orientation, but the offset also, it will be ideal to work with homogeneous coordinates. This will give as result the *extrinsic* transformation matrix.

$$ {}^{C}K_{W} = \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (1) $$

It is quite important to take into account the order in which the axis are going to be rotated, otherwise the rotation could end up in a very different orientation.

```
% Calculation of extrinsic paremeters
R1 = [1 0 0;...
        0 cos(Phix) -sin(Phix);...
        0 sin(Phix) cos(Phix)];
R2 = [cos(Phiy) 0 sin(Phiy);...
        0 1 0;...
        -sin(Phiy) 0 cos(Phiy)];
R3 = [1 0 0;...
        0 cos(Phix1) -sin(Phix1);...
        0 sin(Phix1) cos(Phix1)];
R = R3*R2*R1;          % Rotation Matrix
T = [100 0 1500 1]';   % Translation
R1 = [R; 0 0 0];
cKw = [R1 T];          % Tranformation camera to world
```

The image plane is a representation of the sensor matrix and will be the place where the information of the 3D points is collected by the camera. This information will be a projection of these 3D points. The position of the size of the plane, the location of the principal point in this plane, and the focal lent are the parameters that are needed to find the *intrinsic* transformation matrix.

$$ I = \begin{pmatrix} a_u & 0 & u_0 & 0 \\ 0 & a_v & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \quad (2) $$

```
% Calculation of the intrinsec parameters
I = [au 0 u0 0; 0 av v0 0; 0 0 1 0];
```

Step 3 - Define a set of 6 3D points

For the creation of a given number of points in 3D into a square space as a range is useful to have a function. This function will receive the number of points as arguments and will return the random points in the 3D reference system.

```
Points = points(6); number = size(Points,1);

function Points = points(number);
up = 480; low = -480;
K = rand(number, 3);
K = K * (up - low);
K = K + low;
Points = [K ones(number,1)];
end
```

Step 4 - Get projection of points

To get the projection of the points into the image plane is necessary to define the transformation from world to image. The computation of the product of the intrinsic by the extrinsic transformation matrices will end up with the *camera transformation matrix*.

```
I1 = I * cKw;          % Camera transformation
I1 = I1 / I1(3,4);     % Scale normalization
```

In order to compare the obtained matrix with Hall, and with Faugeras results, a normalization of the values should be computed.

```
[iPw, iPwNorm] = projectingPoints(Points, I1);

function [iPw, iPwnorm] = projectingPoints(Points, I1)
% Projecting and normalizing points over the image
    plane
iPw = I1 * Points';
iPwnorm = iPw;
number = size(Points, 1);
for i=1:number
        iPwnorm(:,i) = iPwnorm(:,i)/iPwnorm(3,i);
        % Devided by scale factor to normalize
end
end
```

Step 5 - Draw 2D points

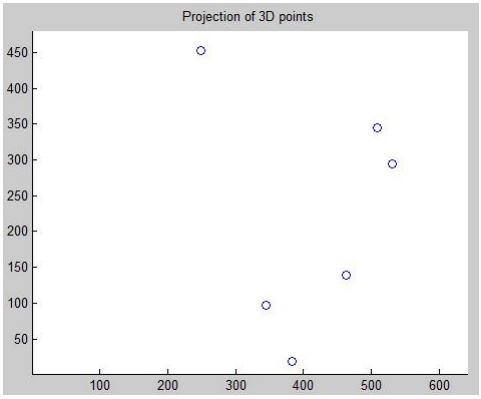

Figure 2: Projection of 6 random 3D points into image plane

The more points in one are the more accurate will be the calibration around that area. But if it is wanted to calibrate the camera for a region where there are no references, the results won't be very accurate. It is important the distribution of the points selected around one area.

Step 6 - Camera transformation using Hall's method

The simplest model to get the image plane reference system from the world coordinate system is the following:

$$\begin{pmatrix} {}^{I}X_{ds} \\ {}^{I}Y_{ds} \\ s \end{pmatrix} = \begin{pmatrix} A_{11} & A_{12} & A_{13} & A_{14} \\ A_{21} & A_{22} & A_{23} & A_{24} \\ A_{31} & A_{32} & A_{33} & A_{34} \end{pmatrix} \begin{pmatrix} {}^{W}X_{W} \\ {}^{W}Y_{W} \\ {}^{W}Z_{W} \\ 1 \end{pmatrix} \quad (3)$$

The idea is to get these unknown variables using least square method, so it will be necessary, at least, 6 projected points. The variable A_{34} is considered equal to 1, for calculus simplification and for avoiding to obtaining infinite number of solutions depending in the value of the scale.

Reshaping the matrix A, as an array of 11 elements and, using equation 3, is possible to apply the least square method to compute the values of A. The only inputs values should be world reference points and projected points.

```
function AAA = Hall(Points, iPwnorm)
A = zeros (12,1);
A(12,1) = 1;  % Avoid infinite solution
number = size(Points, 1);
Q1 = [Points ...
    zeros(number,4)...
        -(iPwnorm(1,:)').*Points(:,1)...
    -(iPwnorm(1,:)').*Points(:,2)...
    -(iPwnorm(1,:)').*Points(:,3)...
        -(iPwnorm(1,:)')];
Q2 = [zeros(number,4)...
    Points...
        -(iPwnorm(2,:)').*Points(:,1)...
    -(iPwnorm(2,:)').*Points(:,2)...
    -(iPwnorm(2,:)').*Points(:,3)...
    -(iPwnorm(2,:)')];
% But as we know that A(3,4) = 1,
Q = zeros(number*2,11);
Q11 = Q1(:,1:11);
Q22 = Q2(:,1:11);
B = zeros(number*2,1);
for i=1:size(Q11,1)
 Q((1+(2*(i-1))),:)=...
 [Points(i,:),
 zeros(1,4),...
 -(iPwnorm(1,i)*Points(i,1)),...
 -(iPwnorm(1,i)*Points(i,2)),...
 -(iPwnorm(1,i)*Points(i,3))];
 Q(2*i,:)=...
 [zeros(1,4),...
 Points(i,:),...
 -(iPwnorm(2,i)*Points(i,1)),...
 -(iPwnorm(2,i)*Points(i,2)),...
 -(iPwnorm(2,i)*Points(i,3))];
end
for i=1:(size(B,1)/2)
        B(1+(2*(i-1)),:)=iPwnorm(1,i);
    B(2*i,:)=iPwnorm(2,i);
end
AA = pinv(Q)*B;
AAA = [AA; 1];
AAA = reshape(AAA,4,3);
AAA = AAA';
end
```

Plotting the results obtained with Matlab:

```
The matrix we get from Hall Method is
   -0.2988   -0.2289   -0.2088  363.5215
   -0.0366    0.2620   -0.4423  298.6679
    0.0002   -0.0004   -0.0005    1.0000

The matrix we get on step 2 is
   -0.2988   -0.2289   -0.2088  363.5215
   -0.0366    0.2620   -0.4423  298.6679
    0.0002   -0.0004   -0.0005    1.0000
```

Figure 3: Hall's method results

Step 7 - Compare the matrices obtained

The matrices obtained in Step 3 and in Step 6 are exactly equivalent. This confirms that Hall's method end up with correct solution with only 6 points if there is no noise that affects the measurements.

Step 8 - Add some Gaussian noise and repeat Step 6

The noise is introduce as follows:

```
noise = normrnd(0, 0.5,[2,6]);% 2*sigma = 1 (95%)
noise = [noise; zeros(1,6)];
iPwNoisy = iPwNorm + noise;
```

After the computation over the noisy projections the matrix obtained with Hall method is:

```
The matrix we get from Hall Method, without noise is
   -0.2988   -0.2289   -0.2088  363.5215
   -0.0366    0.2620   -0.4423  298.6679
    0.0002   -0.0004   -0.0005    1.0000

The matrix we get from Hall Method, wiht noise is
   -0.2988   -0.2272   -0.2103  363.2476
   -0.0358    0.2630   -0.4441  298.7773
    0.0002   -0.0004   -0.0005    1.0000
```

Figure 4: Hall's method results (with noise)

```
The mean of distances computed using 6  points is

MeanDistance6 =

    0.5530


SD6 =

    0.1125
```

Figure 5: Statistics for Hall's results

The parameters used to measure how accurate is the method with noise are mean value, standard deviation, minimum and maximum.

Step 9 - Repeat Step 8 with 10 and 50 points

```
The matrix we get from Hall Method, without noise is
   -0.2988   -0.2289   -0.2088  363.5215
   -0.0366    0.2620   -0.4423  298.6679
    0.0002   -0.0004   -0.0005    1.0000

The matrix we get from Hall Method, with noise is
   -0.3001   -0.2299   -0.2097  363.6151
   -0.0358    0.2613   -0.4416  298.5915
    0.0002   -0.0004   -0.0005    1.0000

The mean of distances computed using 10 points is

MeanDistance10 =

    0.5521


SD10 =

    0.1003
```

Figure 6: Results for Hall's method (10 points)

```
The matrix we get from Hall Method, without noise is
   -0.2988   -0.2289   -0.2088  363.5215
   -0.0366    0.2620   -0.4423  298.6679
    0.0002   -0.0004   -0.0005    1.0000

The matrix we get from Hall Method, with noise is
   -0.2980   -0.2287   -0.2092  363.6260
   -0.0361    0.2619   -0.4429  298.6514
    0.0002   -0.0004   -0.0005    1.0000

The mean of distances computed using 10 points is

MeanDistance50 =

    0.2104


SD50 =

    0.0177
```

Figure 7: Results for Hall's method (50 points)

Part 2 - Faugeras Method

Step 10 - Compute Faugeras - No noise

If the multiplication of intrinsic and extrinsic matrices is computed, this is the equation that appears:

$$\begin{pmatrix} {}^{I}X_d s \\ {}^{I}Y_d s \\ s \end{pmatrix} = \begin{pmatrix} a_u r_1 + u_0 r_3 & a_u t_x + u_0 t_z \\ a_v r_2 + v_0 r_3 & a_v t_y + v_0 t_z \\ r_3 & t_z \end{pmatrix} \begin{pmatrix} {}^{W}X_W \\ {}^{W}Y_W \\ {}^{W}Z_W \\ 1 \end{pmatrix} \quad (4)$$

$$A = \begin{pmatrix} a_u r_1 + u_0 r_3 & a_u t_x + u_0 t_z \\ a_v r_2 + v_0 r_3 & a_v t_y + v_0 t_z \\ r_3 & t_z \end{pmatrix} \quad (5)$$

Where

$$r_1 = \frac{1}{a_u}(A_1 - u_0 A_3),\ t_x = \frac{1}{a_u}(A_{14} - u_0 A_3) \quad (6)$$

$$r_2 = \frac{1}{a_v}(A_2 - v0 A_3),\ t_y = \frac{1}{a_u}(A_{24} - v_0 A_3) \quad (7)$$

$$r_3 = A_3,\ t_z = A_{34} \quad (8)$$

It is possible to rearrange these expressions in order to obtain intrinsic and extrinsic parameters.

The following function was implemented in Matlab to compute Faugeres method:

```
function [IFaug, EFaug] = Faugeras( Points, iPwNorm )
A = zeros (12,1);
A(12,1) = 1;    % Approximation
number = size(Points, 1);
Q = zeros(number*2,11);
B = zeros(number*2,1);
for i=1:number
Q((1+(2*(i-1))),:)=...
[Points(i,1:3),...
-(iPwNorm(1,i)*Points(i,1)),...
-(iPwNorm(1,i)*Points(i,2)),...
-(iPwNorm(1,i)*Points(i,3)),...
zeros(1,3), 1, 0];
Q(2*i,:)=...
[zeros(1,3),...
-(iPwNorm(2,i)*Points(i,1)),...
-(iPwNorm(2,i)*Points(i,2)),...
-(iPwNorm(2,i)*Points(i,3)),...
Points(i,1:3), 0, 1];
end
for i=1:(size(B,1)/2)
B(1+(2*(i-1)),:)=iPwNorm(1,i);
B(2*i,:)=iPwNorm(2,i);
end
AA = pinv(Q)*B;
T1 = AA(1:3,1)';
T2 = AA(4:6,1)';
T3 = AA(7:9,1)';
C1 = AA(10,1);
C2 = AA(11,1);
% Calculating the intrinsics parameters
u0 = (T1*T2')/((norm(T2))^2);
v0 = (T2*T3')/((norm(T2))^2);
au = norm(cross(T1',T2'))/((norm(T2))^2);
av = norm(cross(T2',T3'))/((norm(T2))^2);
IFaug = [au 0 u0 0;          0 av v0 0;     0 0 1 0];
% Calculating the extrinsics parameters
r1 = (norm(T2)/norm(cross(T1',T2')))*...
(T1-((T1*T2')/(norm(T2)^2))*T2);
r2 = (norm(T2)/norm(cross(T2',T3')))*...
(T3-((T2*T3')/(norm(T2)^2))*T2);
r3 = T2/norm(T2);
tx = (norm(T2)/norm(cross(T1',T2')))*...
(C1-((T1*T2')/(norm(T2)^2)));
ty = (norm(T2)/norm(cross(T2',T3')))*...
(C2-((T2*T3')/(norm(T2)^2)));
tz = 1 / norm(T2);
EFaug = [r1 tx; r2 ty; r3 tz; 0 0 0 1];
end
```

After the computation of the intrinsic and extrinsic parameters using this function, the following results are obtained:

```
The original intrinsec parameters are:

I =

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

The intrinsec parameters obtained by Faugeres method are:

IFaug =

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

Figure 8: Comparative of intrinsic parameters

```
The original extrinsec parameters are:

cKw =

   1.0e+03 *

   -0.0010   -0.0003   -0.0001    0.1000
   -0.0002    0.0008   -0.0006         0
    0.0003   -0.0006   -0.0008    1.5000
         0         0         0    0.0010

The intrinsec parameters obtained by Faugeres method are:

EFaug =

   1.0e+03 *

   -0.0010   -0.0003   -0.0001    0.1000
   -0.0002    0.0008   -0.0006   -0.0000
    0.0003   -0.0006   -0.0008    1.5000
         0         0         0    0.0010
```

Figure 9: Comparative of extrinsic parameters

As was expected the intrinsic and extrinsic parameters are the same that the ones obtained with the original features of our system.

Step 11 - Gaussian noise with the methods

Different characterization for the Gaussian noise is going to give different results in the methods. The more is the typical deviation, the less accurate are going to be the methods. The idea is compare the result for a Gaussian noise with several sigma values (0.5, 1, 1.5) and analyze the results.

For a range of 10 noisy points the statistics for Hall and Faugeras are:

```
FAUGERAS RESULTS
The mean of distances for range

Distances =

    0.6454    0.7506    2.1838

The standard deviations for range

StandardDev =

    0.0880    0.1515    0.4130

The minimum of distances for range

Minima =

    0.3182    0.3592    1.0053

The minimum of distances for range

Maxima =

    0.8541    1.5289    4.1425
```

```
HALL RESULTS
The mean of distances for range

Distances =

    0.6475    0.7506    2.2229

The standard deviations for range

StandardDev =

    0.0833    0.1512    0.1718

The minimum of distances for range

Minima =

    0.3107    0.3662    1.7226

The minimum of distances for range

Maxima =

    0.8541    1.5289    4.1425
```

Figure 10: Statistics for sigma = 0.5, 1, 1.5

After several computations the conclusion is that the two methods are quite similar.

Step 12 - Graphical representation

In this step there is a graphical representation of the system. Six 3D points are represented over around the space, and the camera is located and orientated as was defined at the beginning of this lab. the projections over the image plane are obtained with intrinsic and extrinsic matrices. The way to check that all the calculations have been done properly is to extend the lines that connect each point with its projection over the image plane. If these extensions cross at one unique point (focal point), that means that the calculations are good. If there is no one only point were the beams intersect it means that the measurement were noisy.

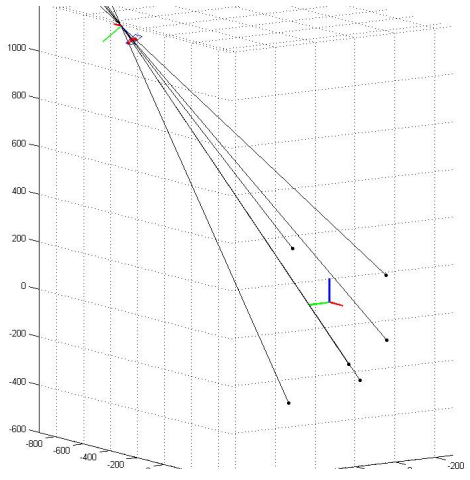

Figure 11: Representation of camera calibration

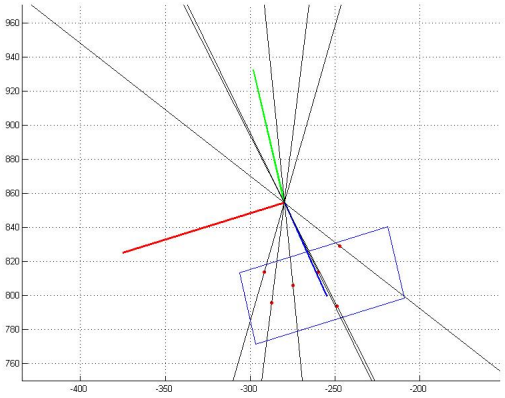

Figure 12: Detail of focal point

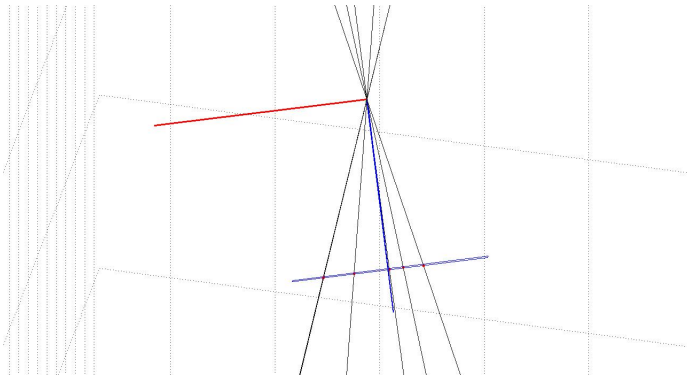

Figure 13: Detail of focal distance

Conclusions

1.- The parameters of the camera, both intrinsic and extrinsic could be calculated from 3D points and their projections into the image plane.
2.- Hall and Faugeras have similar performance with similar noise (Gaussian noise).
3.- The more points are used during the calibration process, the more accurate will be the estimation.
4.- The points used to calibrate the camera should be spread all around the area. If they are not well spread the estimation wouldn't be much accurate.