Multiple View Geometry

Lab1 – Camera Calibration

Lab Sessions: 2 (4 hours)
Total Workload: 10 hours
Programming Language: MATLAB

Setembre 2022 (v1)

1. Objective

The objective of these two lab sessions is to gain experience on the basic concepts of camera projection geometry, through the calibration of a simulated camera.

In this exercise you will:

1. Create a camera projection matrix from a set of parameters.
2. Define 3D points and their 2D image projections.
3. Calibrate using the method of Hall.
4. Check the accuracy of this calibration against increasing level of noise in the image points.

2. Implementation

Part 1

Step 1.
Consider the intrinsic parameters and extrinsic parameters with the following values:

```
% 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
% Euler XYX1 angles
Phix = 0.8*pi/2;
Phiy = -1.8*pi/2;
Phix1 = pi/5;
```

Step 2.
Create the camera intrinsic matrix K and the extrinsic camera transformation ${}^{c}T_{w}$ from the values defined above.
Remember that K and ${}^{c}T_{w}$ are components of the projection matrix P and are defined as

$$K=\begin{bmatrix}\alpha_u & \gamma & u_0\\ 0 & \alpha_v & v_0\\ 0 & 0 & 1\end{bmatrix}$$

$${}^{c}T_{w}=\begin{bmatrix}{}^{c}R_{w} & {}^{c}t_{w}\\ 0\ \ 0\ \ 0 & 1\end{bmatrix}$$

$$P = K \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot {}^{c}T_{w}$$

Step 3.
Define a set of 3D points in the range [-480:480; -480:480; -480:480]. You need a set of at least 6 points, which cannot be co-linear nor co-coplanar.

Note: There is no need to test the co-linearity or the co-planarity of the points. You can just define 6 or more points randomly in the 3D space.

Step 4.
Compute the projection of the 3D points on the image plane by using the camera projection matrix. Do not apply any rounding to the projected points (i.e. do not remove the subpixel precision of the obtained points).

Step 5.
Open a window in Matlab which will be used to show the image plane. Plot the 2D points. Confirm that all the points are well spread in the image plane.

Questions

Q1: Will the distribution of points in the image affect the accuracy in the computation? Why?

Step 6.
By using the 3D points created in Step 3 and their projection obtained in Step 5, estimate the 3x4 camera projection matrix P by using the method of Hall.

Step 7.
Compare the matrix obtained in Step 6 to the one defined in step 2.

Extract the intrinsic parameter matrix K, and the camera rotation matrix ${}^{c}R_{w}$ from the camera projection matrix P (see appendix)

Part 2

Step 8.
Add Gaussian noise to all the 2D points producing discrepancies between the range [-1,+1] pixels for the 95% of points. You can use the function `normrnd`.

Again, repeat step 6 with the noisy 2D points and the 3D points defined in step 3.

Compare the projection matrix you obtain with the one you got in step 6, with the noise-free points.

Extract the intrinsic parameter matrix *K*, and the camera rotation matrix ${}^{c}R_{w}$ from the camera projection you just computed.

Questions

Q2: How did the intrinsic parameters change?
Q3: Why is the axis skew parameter γ different from zero?

Step 9.
Now compute the 2D points with the projection matrix and compare them to those obtained in step 4. For this comparison compute the average projection error. The average projection error is defined as the mean of the Euclidean distance between the 2D points of step 4 and those of step 8.

Step 10.
Increase the number of 3D points up to 10 points and then up to 50 points and repeat step 8 and 9.
Confirm that the higher the number of points used, the more accurate is the obtained projection matrix.

Plot a graph of the average projection error as a function of the number of points

3. Deliverable

This Lab exercise is to be done in groups of 2 students.

Your deliverable will consist of one report in PDF format, and one or more matlab .m files with your code. You will have to upload these files as a single ZIP file to Moodle before the deadline.

The deadline is one week after the last lab session of this exercise.

The PDF should include the names of both members of the group and:

- **Should detail how you have solved every step**
- Include the .m code that solve each step
- Include the obtained results per step
- **Discuss, when necessary, the results obtained**.

The Matlab code should be commented. There is no need for extensive comments, but it should be clearly stated what each part of the code is doing.

Your Matlab code should run as is. You are encouraged to create your own functions to better structure the code. The main execution script should be clearly identified.

Appendix

The following function allows to extract the intrinsic parameters and the rotation matrix from a projection matrix. It uses the RQ decomposition of a square matrix M = R·Q , where R is a upper triangular matrix and Q is and orthogonal matrix.

```
function [K,cRw] = get_intrinsics_from_proj_matrix(P)

% Get left-side 3x3 block of P
M = P(:,1:3);

% This implements the RQ decomposition from QR in matlab
[Q,R] = qr(rot90(M,3));
R = rot90(R,2)';
Q = rot90(Q);

% Check the determinant of Q to make cRw a proper rotation
if det(Q) < 0
    cRw = -Q;
else
    cRw = Q;
end

% Get the normalized intrinsics
K = R/R(3,3);
```