Exercise 3

Deadline: January 19, 2018

Please send your solutions to threedcv@dfki.uni-kl.de

Theory

Assume two fully calibrated cameras with intrinsics K_i and extrinsics $[R_i|t_i]$, $i = 0, 1$ and no distortion.

[T1] Given an image point x_0 in the first view, how does this constrain the position of the corresponding point x_1 in the second image?

[T2] Assume corresponding image points $x_0 \leftrightarrow x_1$ are given. Describe in words how the 3D world point X with $x_i = K_i[R_i|t_i] \cdot (X, 1)^T$ can be computed (only the idea, no mathematical derivation).

[T3] How can the epipoles be computed for the cameras? How are epipolar lines and the epipoles related?

[T4] How can the fundamental matrix be computed when no calibration is given (only the idea, no mathematical derivation)?

[T5] How can the fundamental matrix be computed when the calibration (intrinsics and pose) is given?

Implementation

The goal of the exercises below is to match detected features and reconstruct the original 3D structure. For this task, two calibrated cameras are given (intrinsics K_0, K_1 and extrinsics R_1, t_1 are provided in `data.mat`). Camera 0 coincides with the world coordinate system, thus $R_0 = I$ and $t_0 = 0$. The features originated from a chessboard seen by both cameras (`Camera00.jpg` and `Camera01.jpg`) and are also provided in `data.mat` ($cornersCam0, cornersCam1$). To load the information in `data.mat` you may run `dict = io.loadmat('./data/data.mat')` (io is scipy.io). Then, dict is a dictionary and the necessary information can be accessed via the following keys: `'K_0'`, `'K_1'`, `'R_1'`, `'t_1'`, `'cornersCam0'`, `'cornersCam1'`.

Feature matching using the epipolar constraint

[I1] A corresponding pair of chessboard corners $x \leftrightarrow x'$ satisfies $x'^T F x = x'^T l' = 0$. Use this property to implement a simple matching algorithm. Remark: If $a = (x, y, z)^T$, then

$$[a]_\times = \begin{bmatrix} 0 & -z & y \\ z & 0 & -x \\ -y & x & 0 \end{bmatrix}$$

- Compute the fundamental matrix F and print it to the console.
- For each chessboard feature in camera image 0 (`Camera00.jpg`), compute the corresponding epipolar line $l' = (a, b, c)$ in the image of camera 1 (`Camera01.jpg`).
- Draw each epipolar line computed above. It should intersect the corresponding chessboard corner in the image of camera 1. Note that the image resolution is 4752×3168 pixels. This might be useful to define valid image borders for drawing a line. Save the image with the epipolar lines to `epilines.jpg`.
- Compute the matching feature as the one in $cornersCam1$ with minimal algebraic distance to l'.

- Create a new image with camera 0's image on top and camera 1's image on the bottom (image2 is a "stack" with image 0 on top). Connect corresponding points between the images with lines. Use random line colors and do not forget the y-displacement for the correspondences in the lower image. Save this image to `matches.jpg`.

Structure reconstruction

[**I2**] Use the correspondences to triangulate the corners of the chessboards in the world coordinate system. Use the method that was introduced in the lecture. Each camera image yields 2 linearly independent equations. Thus, an overdetermined system with 4 equations and 3 unknowns for each correspondence has to be solved. Plot your reconstructed chessboard points in 3D space. For that it is recommended to use `matplotlib` library. Also visualize the optical center and the optical axis (the z-axis) of each camera in the world coordinate system.

Theory

[**T6**] Name a fundamental problem of the matching technique used in [**I1**]. When does it make sense to match features this way?

Remark:

1. Make sure your code executes the tasks above sequentially by simply calling `python main.py` (include `data.mat` alongside `main.py` in your `.zip` file).

Good Luck!