Computer Vision Project
Wintersemester 2017/2018
Tobias Geimer, André Aichert, Vincent Christlein,
Peter Fürsattel, Franz Köferl, Franziska Schirrmacher

Sheet 3, starting from November 20th, 2017, due February 5th, 2018

Introduction and General Comments:

- This exercise covers two separate topics: camera calibration and stereo vision.
- The folder '*ex03.zip*' contains code skeletons and data for their respective exercises and can be downloaded from StudOn. However, you are not obliged to use these templates if you prefer to work independently.
- For a deeper understanding, please have a look at the literature:
  - Calibration: Zhang's Technical Report [1]
    `https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/tr98-71.pdf`
  - Stereo Vision: Hartley's Part II: Two-View Geometry [2]
    `http://cvrs.whu.edu.cn/downloads/ebooks/Multiple%20View%20Geometry%20in%20Computer%20Vision%20(Second%20Edition).pdf`

Exercise 3.1: Camera Calibration

In this exercise you will implement an initialization step for camera calibration in Matlab following Zhengyou Zhang's algebraic closed-form solution [1].

a) Preprocessing

The folder '*data/ex03_1*' contains $n = 15$ images of the calibration pattern from different orientations. Point correspondences ($\boldsymbol{x} \in \mathbb{R}^2 \leftrightarrow \boldsymbol{X} \in \mathbb{R}^3$) are already provided in *corPts.mat*. Points are euclidean and need to be transformed to homogeneous coordinates. If you want to implement a feature extraction of your own, *checkerboard.txt* specifies the needed phantom dimensions to model the world points.

b) Homography

The detected image points $\boldsymbol{x} = (u, v, 1)^T$ relate to their world correspondences $\boldsymbol{X}_w = (x, y, z, 1)^T$. Defining the board to be in the xy-plane (i.e. $z = 0 \,\forall\, \boldsymbol{X}_w$), this can be described as a 3×3 homography:

$$\boldsymbol{x} = \boldsymbol{H}\boldsymbol{X}, \quad \text{where } \boldsymbol{X} = (x, y, 1)^T . \tag{1}$$

You already know, that *DLT* can be used to solve this type of problem. Remember to **normalize** your data points to avoid a numerically unstable system matrix. Determine $\boldsymbol{H}$ for each of the 15 views.

c) Absolute Conic

The resulting homography contains information about the intrinsic and extrinsic parameters. It was shown in the lecture, that $\boldsymbol{H}$ can be related to the absolute conic $\boldsymbol{B} = \boldsymbol{K}^{-T}\boldsymbol{K}^{-1}$ by exploiting the orthonormality of the rotation vectors $\boldsymbol{r}_1, \boldsymbol{r}_2$, resulting in a system of equations linear in $\boldsymbol{b}$.

$$\begin{bmatrix} \boldsymbol{g}_{12}^T \\ (\boldsymbol{g}_{11} - \boldsymbol{g}_{22})^T \end{bmatrix} \boldsymbol{b} = \boldsymbol{0} \tag{2}$$

Fig. 1: Extracted image points and reprojected world points for view 1 and 2.

where

- $\boldsymbol{g}_{ij} = \begin{pmatrix} h_{i1}h_{j1}, & h_{i1}h_{j2} + h_{i2}h_{j1}, & h_{i2}h_{j2}, & h_{i3}h_{j1} + h_{i1}h_{j3}, & h_{i3}h_{j2} + h_{i2}h_{j3}, & h_{i3}h_{j3} \end{pmatrix}^T$
- $\boldsymbol{b} = \begin{pmatrix} B_{11}, & B_{12}, & B_{22}, & B_{13}, & B_{23}, & B_{33} \end{pmatrix}^T$

Build $\boldsymbol{g}_{(i)}$ from the columns $\boldsymbol{H}_{(i)}$ for each image and solve the stacked equations

$$\boldsymbol{G}\boldsymbol{b} = \boldsymbol{0}, \quad \boldsymbol{G} \in \mathbb{R}^{2n \times 6}. \tag{3}$$

d) Camera Parameters

Recover the intrinsic parameters (`ex03_1d_intrinsicParamsFromConic.m`) from $\boldsymbol{b}$ and fill $\boldsymbol{K}$ accordingly. Now, you can compute the extrinsic parameters (`ex03_1d_extrinsicParamsFromHom.m`) for each image (see slides 18-19). Make sure the rotation matrices $R_{(i)}$ actually describe an orthonormal system.

e) Error Evaluation

It's time to evaluate the calibrated parameters (`ex03_1e_reprojectError.m`). Project the 3-D points to the image plane. Calculate the average reprojection error as the mean euclidean distance between the reprojected points $\boldsymbol{P}\boldsymbol{X}$ and the extracted feature points $\boldsymbol{x}$. Take into account that $\boldsymbol{P}$ is a projection **up-to-scale** when comparing the points. For one image of your choice, plot both point types into the image and visualize the error for each point pair as an up-scaled arrow or line (see Fig. 1).

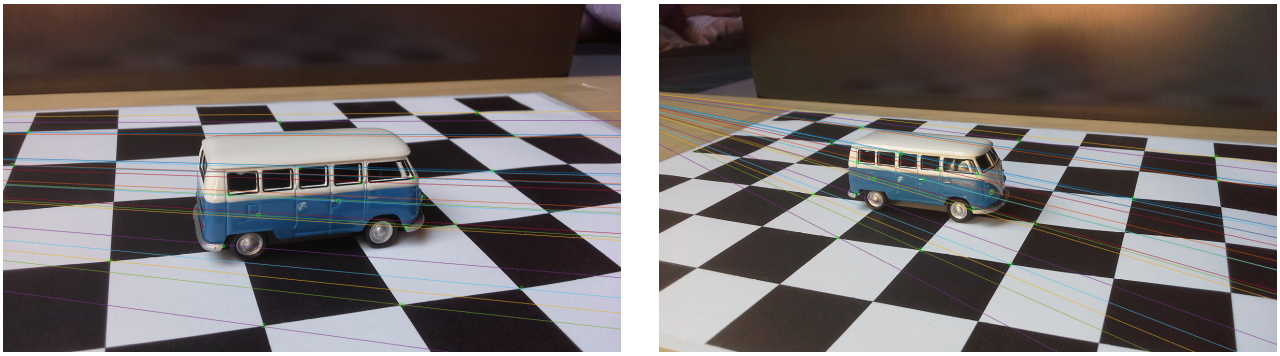

Fig. 2: Left & right view with selected points and epipolar lines.

Exercise 3.2: Stereo Vision

In this exercise you will cover a simple stereo vision setup. The goal is to estimate the relative position of two calibrated cameras using only image-based features and epipolar constraints.

a) Data

The folder '*data/ex03_2*' contains two images of a small VW Bus (also called a "Bulli") acquired from two different positions. These images were taken with the same camera that was already calibrated in exercise 3.1. Consequently, the intrinsic camera matrix $\boldsymbol{K}$ is known (and given to you in *intrinsic_K.txt*). Use `ex03_2_stereovision.m` to load the available data into your Matlab workspace.

b) The Eight Point Algorithm

In order to determine the relative camera position we start by computing the **fundamental matrix** $\boldsymbol{F}$ that describes the epipolar constraints between both views. The fundamental matrix may be estimated up to scale from at least 8 point correspondences in both views.
`20bulliPoints.mat` provides 20 such preselected corresponding points. However, you may also write a function to select a number $n = 8$ of corresponding points in both views (hint: `ginput`) or use the point selection script from exercise sheet 4, if you previously implemented it for generic n.

1. Load the corresponding points in `20bulliPoints.mat`.
2. Implement the Eight-Point-Algorithm in `ex03_2b2_computeF.m` according to the pseudo-code provided in the lecture. Remember to **normalize** your image points.
3. You can check your result by showing that $\boldsymbol{F}$ holds $\boldsymbol{x}'^T \boldsymbol{F} \boldsymbol{x} = 0$.
4. Methods are provided to test your implementation by plotting the selected points and the epipolar lines $\boldsymbol{l}_i = \boldsymbol{F}\boldsymbol{x}_i$ and $\boldsymbol{l}'_i = \boldsymbol{F}^T\boldsymbol{x}'_i$ in the respective other view (see Fig. 2).

Manual selection is always affected by inaccuracies, which reflects in inaccurate results for the fundamental matrix. Can you think of a suitable method to cope with noise in these correspondences?

c) Possible Camera Matrices

Since the camera is calibrated, the **essential matrix** $\boldsymbol{E} = \boldsymbol{K}^T\boldsymbol{F}\boldsymbol{K}$ is computed in a straight-forward matrix multiplication. With intrinsic parameters removed, the essential matrix encodes only the relative extrinsic parameters between both camera positions (up to a 4-fold ambiguity). Given the first camera $\tilde{\boldsymbol{P}} = [\boldsymbol{I}\ \boldsymbol{0}]$, there are four possibilities $\tilde{\boldsymbol{P}}' = [\boldsymbol{R}\ \boldsymbol{t}]$ for the second one (see slides 36-37). Complete `ex03_2c_possibleProjectionMatrices.m` that accepts an essential matrix E and returns the four possible extrinsic matrices for the second camera.

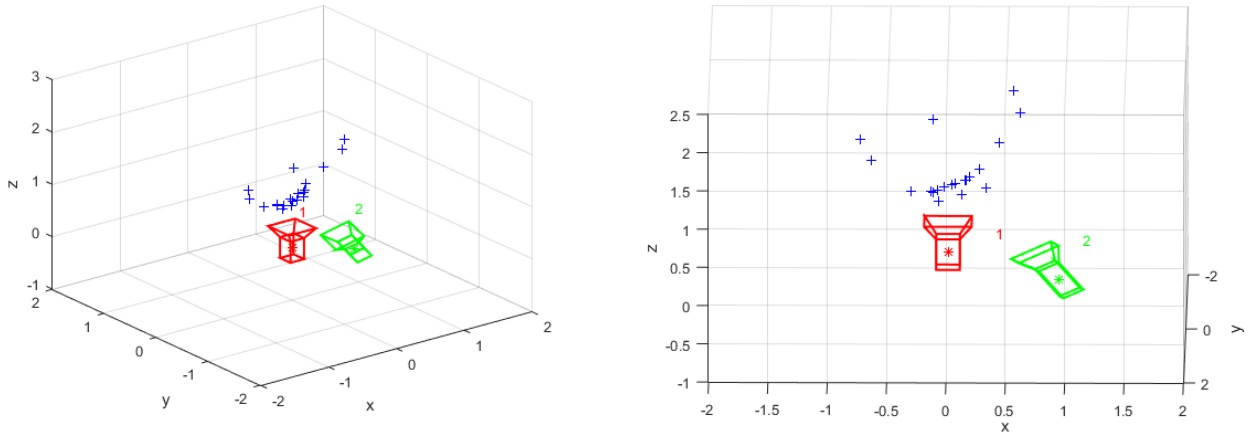

Fig. 3: Relative position of cameras with triangulated points in the first camera's coordinate system.

d) Triangulation

Only one of the four matrices describes a physically meaningful setup (i.e. the one where the world point is in front of both cameras). To solve this ambiguity, an evaluation of triangulated world points (depth value) is necessary. For the general case, implement a triangulation method in `ex03_2d_triangulation.m` (see slides 38-39) that takes two sets of corresponding points $\boldsymbol{x}, \boldsymbol{x}'$ and their respective projection matrices $\boldsymbol{P}, \boldsymbol{P}'$, and returns the set of triangulated 3-D points $\boldsymbol{X}$.

e) Resolve Ambiguity

The correct solution of **c)** is identified by the triangulated world points having **positive z-values** in both **camera coordinate** systems. Use your triangulation method to triangulate a world point from the first camera together with each option of the possible second camera. Decide which setup is correct and print the $[\boldsymbol{R}\,\boldsymbol{t}]$ matrix. `ex03_2e_getCorrectP.m` provides a guideline for this procedure.

e) Scene Visualization

With the correct projection matrix identified, visualize the position and orientation of both cameras and the triangulated points in a 3-D plot (see Fig. 3). For a correct representation, think about how to represent the position and orientation of the second camera in the coordinate system of the first.

Show your implementation and your results to one of the supervisors.

References

[1] Zhengyou Zhang. A flexible new technique for camera calibration. *IEEE Trans. Pattern Anal. Machine Intell.*, 22(11):1330–1334, Nov 2000.

[2] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision.* Cambridge university press, 2003.