Vision-Based Navigation
V. Usenko, N. Demmel, D. Cremers
Summer Semester 2019

Computer Vision Group
Department of Informatics
Technical University of Munich

Exercise Sheet 3

Topic: Feature Detectors, Descriptors, Epipolar Geometry, RANSAC

Submission deadline: Sunday, 26.05.2019, 23:59

Hand-in via merge request

General Notice

The exercises should be done by yourself. We will use Ubuntu 18.04 in this lab course. It is already installed on the lab computers. If you want to use your own laptop, you will need to install Ubuntu yourself. Ubuntu 16.04 and macOS *may* also work, but might require some more manual tweaking.

Exercise 1: ORB Descriptors

In the lecture we explained you how to detect keypoints in images and compute their descriptors. In this exercise you will implement keypoint detection, descriptor computation and matching. We provide you the skeleton code that loads and visualizes the dataset. The code framework is provided in `src/sfm.cpp`. In the next exercise we will then build on these results to create a 3D map of camera poses and matched feature points.

1. Before computing the ORB descriptors [2] we first need to select keypoints and estimate their orientations. In this exercise we already provide you a list of the keypoints detected with the Shi-Tomasi algorithm [3] implemented in OpenCV. Inspect the function `detectKeypoints` in `include/visnav/keypoints.h` for the details. In the same file you should implement the angle computation in the `computeAngles` function. The angle θ of a keypoint can be computed as (see [2] for details):

$$m_{pq} = \sum_{(x,y)\in N} x^p y^q I(x+u, y+v), \tag{1}$$

$$\theta = \text{atan2}(m_{01}, m_{10}), \tag{2}$$

where (u, v) is the location of the detected keypoint and m_{01} and m_{10} are the intensity moments. Here, we choose a circular patch N around a keypoint with radius 15 (`HALF_PATCH_SIZE`). It means that for the keypoint at (u, v), we take all points $(x+u, y+v)$ where $x^2+y^2 <= 15^2$. After implementing the angle computation you should be able to see the orientation of the features as shown in Figure 1.

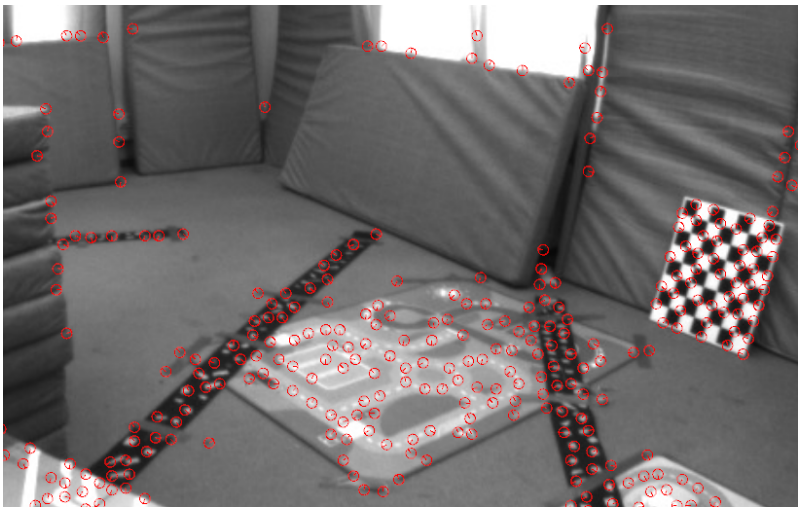

Figure 1: The keypoint orientation is indicated by a line within the circles.

2. ORB uses a BRIEF descriptor, which is a binary descriptor with 256 bits containing 0 and 1. Each bit is the result of a simple binary test that compares image intensities around the detected keypoint. The algorithm is described in the following:

   - Given an image I, keypoint (u, v) and its angle θ, we compute a 256-bit descriptor. The descriptor is stored as a `std::bitset`

     $$\mathbf{d} = [d_1, d_2, \ldots, d_{256}], d_i = \{0, 1\}.$$

   - For each $i = 1, \ldots, 256$, d_i is computed as follows. We take two points around (u, v), say, $\mathbf{p_a} = (u_a, v_a)$ and $\mathbf{p_b} = (u_b, v_b)$, and rotate them for the angle θ:

     $$\begin{bmatrix} u_a{}' \\ v_a{}' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} u_a \\ v_a \end{bmatrix}. \tag{3}$$

     such that u_a', v_a' are the rotated coordinates of u_a and v_a. The same rotation procedure is applied to (u_b, v_b). We denote the rotated points as $\mathbf{p_a}', \mathbf{p_b}'$ and compare the image intensity of $I(\mathbf{p_a}')$ and $I(\mathbf{p_b}')$. If $I(\mathbf{p_a}') < I(\mathbf{p_b}')$, then $d_i = 1$, otherwise $d_i = 0$. To get integer coordinates of the rotated points please use `round` function.

   Please implement the `computeDescriptors` function in `include/visnav/keypoints.h`. Note that the $\mathbf{p_a}, \mathbf{q_b}$ (ORB pattern) is given in the code. If you are interested how the pattern was selected, see [2].

3. Let us next look at brute force matching of ORB features. After computing the descriptors, we need to match them according to the descriptors. Brute force matching is a simple and commonly used approach for feature matching, especially when the number of features is not large. Given two sets of descriptors, say, $\mathbf{P} = [p_1, \ldots, p_M]$ and $\mathbf{Q} = [q_1, \ldots, q_N]$, for each descriptor in $\mathbf{P}$, we find a

descriptor in **Q** that has the minimum (Hamming) distance. Check the documentation for `std::bitset` to efficiently implement the Hamming distance computation. To filter the wrong matches you should use several checks:

- Discard matches with distance larger or equal to the `threshold`.
- Discard matches if the distance to the second best match is smaller than the smallest distance multiplied by `dist_2_best`.
- Match P to Q and Q to P and consider the match valid only if it agrees between these two matchings.

Please implement the `matchDescriptors` function in `include/visnav/keypoints.h` according to these instructions. The matching results should be similar to Fig. 2. You should press `show_ids` to visualize the matched indices.

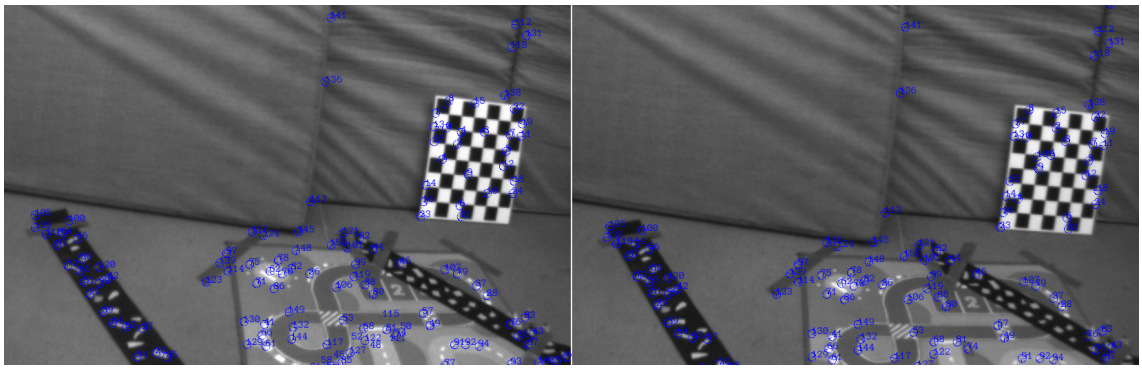

Figure 2: Matched key points.

Exercise 2: Epipolar constraint

Despite the filtering stage our matches still may contain outliers. In this exercise we will use the epipolar constraint to filter them out when matching images from left and right cameras of the stereo setup.

The idea of epipolar geometry is visualized in Figure 3. If point X is observed by two cameras, this point and two focal points O_L and O_R of the cameras form a plane. The epipolar constraint states that the vectors XO_L, XO_R, O_RO_L should lie in the same plane. Note that XO_L, XO_R can be obtained by unprojecting the detected points in the image plane using the camera models as in previous exercise and O_RO_L and R_{LR} (the rotation between cameras) for the case of stereo camera is known from the calibration.

The epipolar constraint can be formulated using the essential matrix as follows

$$(XO_L)^T E(XO_R) = 0. \tag{4}$$

First derive the computation of the essential matrix from the transformation between two cameras in the PDF file. Then implement the `computeEssential` and

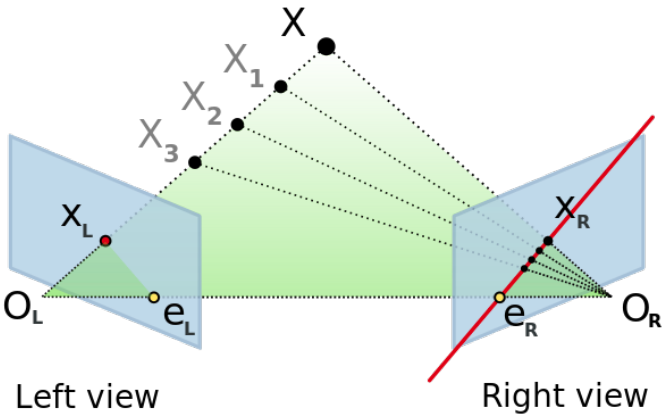

Figure 3: Epipolar constraint.

`findInliersEssential` functions in `include/visnav/matching_utils.h`. When computing the essential matrix please **normalize the translation vector**. For the inliers the epipolar constraint should be smaller than the threshold. After implementing the epipolar constraint you should be able to see (Figure 4) that it helps to reject the wrong matches even when the descriptors are very similar.

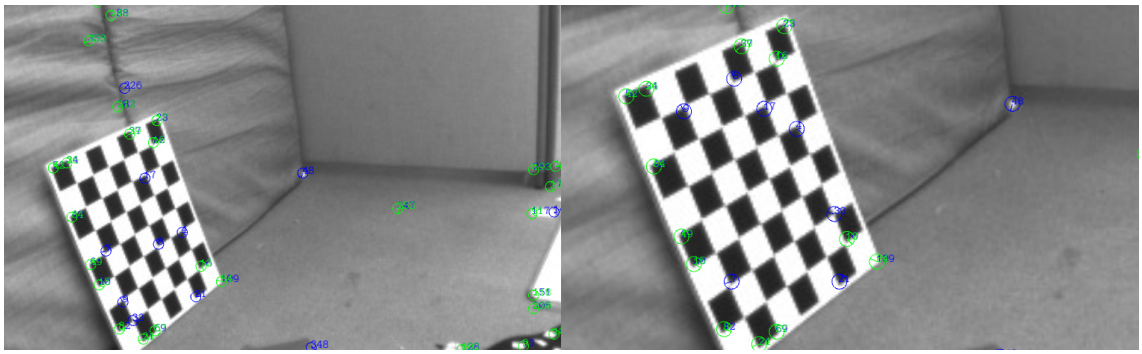

Figure 4: Inliers shown in green. Rejected matches shown in blue.

Exercise 3: Five-point algorithm and RANSAC

For the previous exercise we assumed that we know the transformation between the cameras. It turns out that it is possible to compute it using 5 matching points between the images [1] and use the same math to find inliers. Since the algorithm is hard to implement you should use the OpenGV library. Read the documentation at `https://laurentkneip.github.io/opengv/page_how_to_use.html` and implement the central relative pose problem with RANSAC in `findInliersRansac` functions in `include/visnav/matching_utils.h`.

Before submitting the exercise uncomment the following line in `test/CMakeLists.txt`.

```
gtest_discover_tests(test_ex3)
```

If everything is implemented correctly the system should pass all tests.

Submission Instructions

A complete submission consists both of a PDF file with the solutions/answers to the questions on the exercise sheet and a merge request against the `master` branch with the source code that you used to solve the given problems. Please note your name in the PDF file and submit it as part of the merge request by placing it in the `submission` folder.

References

[1] David Nistér. "An efficient solution to the five-point relative pose problem". In: *IEEE transactions on pattern analysis and machine intelligence* 26.6 (2004), pp. 756–770.

[2] E. Rublee et al. "ORB: An efficient alternative to SIFT or SURF". In: *2011 International Conference on Computer Vision*. 2011, pp. 2564–2571. DOI: 10.1109/ICCV.2011.6126544.

[3] Jianbo Shi and Carlo Tomasi. "Good features to track". In: *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on*. IEEE. 1994, pp. 593–600.