Vision-Based Navigation
D. Schubert, J. Chui, N. Demmel, D. Cremers
Summer Semester 2021

Computer Vision Group
Department of Informatics
Technical University of Munich

Exercise Sheet 2

Topic: Camera Models, Optimization, Calibration

Submission deadline: Sunday, 09.05.2021, 23:59
Hand-in via merge request

General Notice

The exercises should be done by yourself. We use Ubuntu (20.04, 18.04) or macOS (Catalina) in this lab course. Other Linux distributions and macOS versions *may* also work, but might require some more manual tweaking.

Part 1: Camera models

Read the paper describing the projection models for wide-angle cameras [1]. Implement the projection and unprojection functions in the `include/visnav/camera_models.h` for the models provided in the file. Inspect the `test/src/test_ex2.cpp` file and describe in the PDF file what exactly it tests. Uncomment the line

```
gtest_discover_tests(test_ex2 ...
```

in the `test/CMakeLists.txt` and run the test. If the models are implemented properly the test should pass. Hints:

- Avoid using `std::pow()` function to maintain the precision. For example, if you need to compute x^2 use multiplication: $x * x$.
- If your compiler complains about Jet types try changing the constants in projection and unprojection functions to `Scalar(<constant>)`. For example, `Scalar(1)` instead of `1`.
- You can use Newton's method `https://en.wikipedia.org/wiki/Newton's_method` to compute a root of the polynomial given a good initialization. Usually 3-5 iterations should be enough for the optimization to converge.
- You can use Horner's method `https://en.wikipedia.org/wiki/Horner's_method` to efficiently compute polynomials.

Part 2: Optimization

In this lab course we will use Ceres as an optimization library. It provides a general interface for solving non-linear least squares problems. Make yourself familiar with the library and follow the tutorial provided here: `http://ceres-solver.org/nnls_`

`tutorial.html#curve-fitting` and `http://ceres-solver.org/nnls_tutorial.html#robust-curve-fitting`. What is the difference between these curve fitting examples? Describe the answer in your PDF.

Part 3: Camera calibration

In this exercise we will implement a stereo camera calibration using Ceres and the camera models that we have implemented in the previous exercises. First, inspect the `src/calibration.cpp` file. What are the command line parameters that it uses? Write the answer in PDF. The code in this file loads the dataset with images, detected corners (red) and approximate calibration. For the calibration grid we know the 3D coordinates of each corner, so we can project it using the current camera calibration and estimate of camera position with respect to the calibration pattern and compute the coordinates of the projected points (magenta). By minimizing the difference between detected points and projected points we can perform the camera calibration.

- Implement the point projection in the `compute_projections()` function and test the code by running

  ```
  ./build/calibration --dataset-path data/euroc_calib/
  ```

  If your implementation is correct you should be able to see the projections as in Figure 1. At the moment it is OK that the fitting is not perfect, because we use the approximate calibration and poses from the initialization procedure.

Figure 1: Point projections with initial calibration. Detected point shown in red and projected point are shown in magenta.

- Implement the `ReprojectionCostFunctor` in `include/visnav/reprojection.h` and `optimize()` function in `src/calibrate.cpp` to minimize the reprojection error using Ceres. If your implementation is correct, after optimization the projected corners should well align with detected corners as shown in Figure 2.

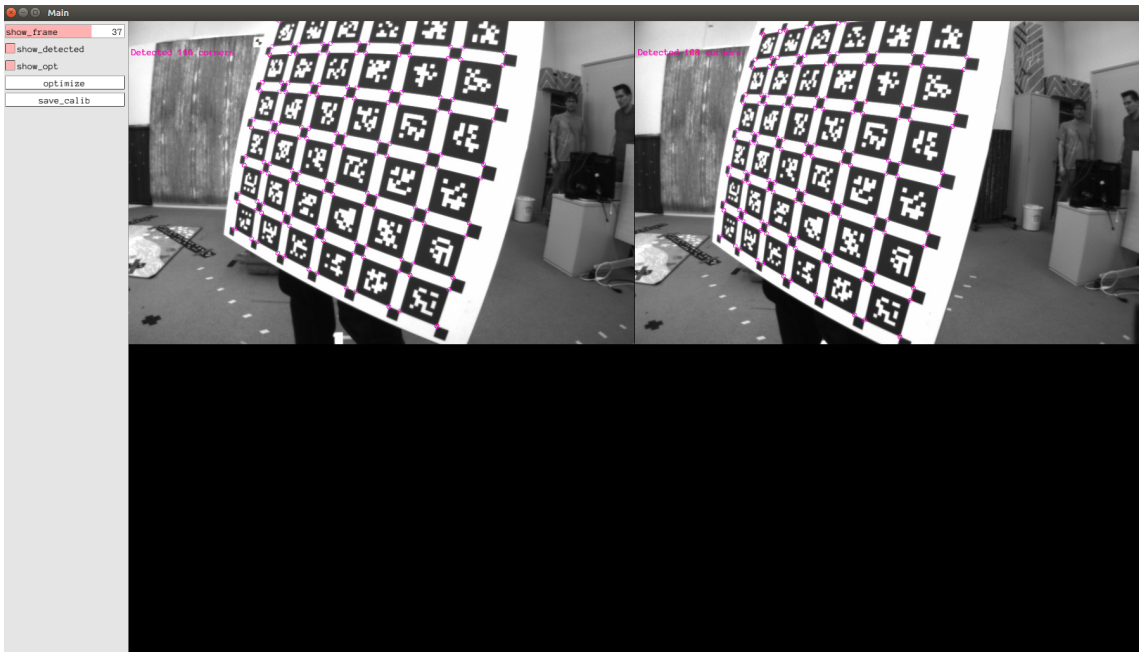

Figure 2: Point projections after optimization.

- As you have noticed the code supports different camera models (pinhole, ds, eucm, kb4) with the command line parameter. For example:

  ```
  ./build/calibration --dataset-path data/euroc_calib/ --cam-model kb4
  ```

  Run the calibration for all models. Inspect the output of the program to find a quantitative measure that can be used to determine how well the camera models fit the lenses that were used to collect the dataset. Provide summary and analysis of the calibration results in the PDF file.

- Uncomment the line

  ```
  add_test(NAME ex2_calibration_test ...
  ```

  in the `test/CMakeLists.txt` to enable the test for this exercise.

Hints:

- For the optimization you should use the local parametrization for $SE(3)$ poses. Please check `src/test_ceres_se3.cpp` to see how you should set up the local parametrization.

- The tests that we provide assume that the transformation from the first camera to the body frame is fixed as in initialization. You should disable the opti-

mization for that parameter using the `problem.SetParameterBlockConstant` method.

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] Vladyslav Usenko, Nikolaus Demmel, and Daniel Cremers. "The Double Sphere Camera Model". In: *Proc. of the Int. Conference on 3D Vision (3DV)*. Sept. 2018. eprint: `http://arxiv.org/abs/1807.08957`.