Vision-Based Navigation
V. Usenko, N. Demmel, D. Schubert, D. Cremers
Summer Semester 2020

Computer Vision Group
Department of Informatics
Technical University of Munich

Exercise Sheet 1

Topic: Introduction to SLAM and Lie Groups

Submission deadline: Sunday, 10.05.2020, 23:59

Hand-in via merge request

General Notice

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

Exercise 1: What is SLAM

Survey papers are very useful when you want to quickly know a new research area, especially for large topics like SLAM, which has almost 30 years history and a tremendous number of papers. Read the paper [1] and answer the following questions:

1. Why would a SLAM system need a map?
2. How can we apply SLAM technology into real-world applications?
3. Describe the history of SLAM.

Hint: You can find the paper online, but for your convenience we have also included this and other relevant papers in the `literature` folder of your code repository.

Exercise 2: git, cmake, gcc, merge-requests

Setup Follow the instructions provided in the documentation page in your repository[1] to build the code for this lab course.

CMake CMake is a commonly used and convenient tool for organizing C++ programs under Linux. There are many libraries, such as OpenCV, Ceres, etc., that use cmake to build the project. Whether you are using someone else's library or writing your own one, you need to know the basics of cmake. You can find cmake tutorials in the following pages: `https://cmake.org/cmake-tutorial/`, `https://github.com/ttroy50/cmake-examples`.

[1] `https://gitlab.vision.in.tum.de/visnav_ss20/<username>/visnav/blob/master/wiki/Setup.md`

Inspect the `CMakeLists.txt` file of the lab course code and explain in your PDF file what the following lines do:

- ```
  set(CMAKE_MODULE_PATH "${CMAKE_CURRENT_SOURCE_DIR}/cmake_modules/" ${
      CMAKE_MODULE_PATH})
  ```

- ```
  set(CMAKE_CXX_STANDARD 14)
  set(CMAKE_CXX_STANDARD_REQUIRED ON)
  set(CMAKE_CXX_EXTENSIONS OFF)
  ```

- ```
  set(CMAKE_CXX_FLAGS_DEBUG  "-O0 -g -DEIGEN_INITIALIZE_MATRICES_BY_NAN")
  set(CMAKE_CXX_FLAGS_RELWITHDEBINFO "-O3 -DNDEBUG -g -
      DEIGEN_INITIALIZE_MATRICES_BY_NAN")
  set(CMAKE_CXX_FLAGS_RELEASE "-O3 -DNDEBUG")
  SET(CMAKE_CXX_FLAGS " -ftemplate-backtrace-limit=0 -Wall -Wextra ${
      EXTRA_WARNING_FLAGS} -march=${CXX_MARCH} ${CMAKE_CXX_FLAGS}")
  ```

- ```
  add_executable(calibration src/calibration.cpp)
  target_link_libraries(calibration ceres pangolin TBB)
  ```

Test Submission The submission of the code in this lab course is done via merge requests on Gitlab. After you've made a merge request, the automatic build system compiles the code and runs the unit tests. You can inspect the code of the tests in the `test` folder. **Do not modify the content of this folder, except to enable the tests for subsequent exercises.**

From the beginning there should be `test_ex0.cpp` available. To test the submission system, uncomment the following line in `test/CMakeLists.txt`, commit the code to your own branch (e.g. `ex0`) and push it to the server.

```
gtest_discover_tests(test_ex0 ...
```

Then make a merge request against the `master` branch to verify that the automatic build system works for you.

On the merge request page you should see the changed files and result of running the tests. If all tests passed your solution is likely correct and we will merge the changes in your `master` branch. It is best to start a new branch for every exercise sheet (e.g. `ex1`, `ex2`, ...), such that you can start work on the next exercise while waiting for your merge request to be verified and merged. Since the exercise sheets build on top of each other, start the next branch based on the previous one, if that isn't merged into master yet, e.g.:

```
git checkout -b ex1 ex0
```

Hint: You should aim to complete Exercise 2 during the first tutor session to make sure your setup is complete and you are ready for the other exercises.

Exercise 3: $SO(3)$ and $SE(3)$ Lie groups

We have shown how to compute the exponential map for $SO(3)$ group in the lecture. In this exercise please derive a closed-form solution to compute the exponential map for the $SE(3)$ group. Assume $\boldsymbol{\xi}^\wedge \in \mathfrak{se}(3)$, $\boldsymbol{\xi} = [\boldsymbol{v}^\top, \boldsymbol{w}^\top]^\top$, where $\boldsymbol{v}$ is the translation part and $\boldsymbol{w}$ is the rotation part. From the general definition of the exponential map,

show we can rewrite it for $SE(3)$ as:

$$\exp\left(\boldsymbol{\xi}^{\wedge}\right) = \begin{bmatrix} \sum_{n=0}^{\infty} \frac{1}{n!}(\boldsymbol{w}^{\wedge})^n & \sum_{n=0}^{\infty} \frac{1}{(n+1)!}(\boldsymbol{w}^{\wedge})^n \boldsymbol{v} \\ \mathbf{0}^{\mathrm{T}} & 1 \end{bmatrix}. \tag{1}$$

The rotational part we recognize as the $SO(3)$ exponential map that can be computed using Rodrigues' formula. For the translational part, let $\theta = \|\boldsymbol{w}\|$, and show that we have:

$$\sum_{n=0}^{\infty} \frac{1}{(n+1)!}(\boldsymbol{w}^{\wedge})^n = I + \frac{1-\cos(\theta)}{\theta^2}\boldsymbol{w}^{\wedge} + \frac{\theta - \sin(\theta)}{\theta^3}(\boldsymbol{w}^{\wedge})^2 = \boldsymbol{J}. \tag{2}$$

Please include the proof of this equation in the submitted PDF.

Hint: Use Taylor expansion and put the odd and even items together, just like what we do in $SO(3)$'s case.

In the source code find the `include/visnav/ex1.h` and implement exp and log maps for $SO(3)$ and $SE(3)$ without Sophus library. After that, uncomment the following line in `test/CMakeLists.txt`, commit the code to your own branch (not `master`) and push it to the server.

```
gtest_discover_tests(test_ex1 ...
```

If your implementation is correct it should successfully pass all tests. You can also run the tests on the local PC by executing

```
cd build
ctest
```

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] Cesar Cadena et al. "Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age". In: *IEEE Transactions on Robotics* 32.6 (2016), pp. 1309–1332. eprint: `https://arxiv.org/abs/1606.05830`.