Vision-Based Navigation
V. Usenko, N. Demmel, D. Cremers
Winter Semester 2018/2019

Computer Vision Group
Department of Informatics
Technical University of Munich

Exercise Sheet 1

Topic: Introduction to SLAM and Lie Groups

Submission deadline: Sunday, 04.11.2018, 23:59 pm

Hand-in via merge request and email to `visnav_ws2018@vision.in.tum.de`

General Notice

The exercises should be done by yourself. We will use Ubuntu 16.04 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.

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.

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

Follow the instructions provided in the wiki page `https://gitlab9.in.tum.de/visnav_ws18/visnav_ws18/wikis/home` to build the code for this lab course.

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`.

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 11)
  set(CMAKE_CXX_STANDARD_REQUIRED ON)
  set(CMAKE_CXX_EXTENSIONS OFF)
  ```

- ```
  SET(CMAKE_CXX_FLAGS_DEBUG "-O0 -g")
  SET(CMAKE_CXX_FLAGS_RELEASE "-O3 -DNDEBUG -ftree-
     vectorize -march=${CXX_MARCH}")
  SET(CMAKE_CXX_FLAGS_RELWITHDEBINFO "-O3 -g -DNDEBUG -
     ftree-vectorize -march=${CXX_MARCH}")
  SET(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} -ftemplate-
     backtrace-limit=0")
  ```

- ```
  add_executable(calibration src/calibration.cpp)
  target_link_libraries(calibration ceres ${
     Pangolin_LIBRARIES})
  ```

The submission of the code in this lab course is done via merge requests. After you've made a merge request and 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. From the beginning there should be `test_ex0.cpp` available for testing. Please create a branch and make a merge request against the `master` branch (within your personal repository) to verify that the automatic build system works for you.

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 the exponential map for $SE(3)$ group. Assume $\boldsymbol{\xi} = [\boldsymbol{\rho}, \boldsymbol{\phi}]^{\mathrm{T}} \in \mathfrak{se}(3)$, where $\boldsymbol{\rho}$ is the translation part and $\boldsymbol{\phi}$ is the rotation part. We know its exponential map is:

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

Let $\boldsymbol{\phi} = \theta \boldsymbol{a}$, then we have:

$$\sum_{n=0}^{\infty} \frac{1}{(n+1)!} (\boldsymbol{\phi}^{\wedge})^n = \frac{\sin\theta}{\theta} I + \left(1 - \frac{\sin\theta}{\theta}\right) \boldsymbol{a}\boldsymbol{a}^T + \frac{1-\cos\theta}{\theta} \boldsymbol{a}^{\wedge} \triangleq \boldsymbol{J}. \tag{2}$$

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/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.
  ```

```
add_test(test_ex1 test_ex1 COMMAND Test)
```

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/test/
ctest -v
```

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. Please submit your PDF file with solutions via email to `visnav_ws2018@vision.in.tum.de`.

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`.