Answer Sheet 1

Topic: Introduction to SLAM and Lie Group

Yihao Wang 03740227

November 5, 2022

Part 1: setup, git, merge-requests, cmake, gcc

- Set(CMAKE_MODULE_PATH "$CMAKE_CURRENT_SOURCE_DIR/cmake_modules/" $CMAKE_MODULE_PATH)
  ⇒Set the cmake module path to the correspond address.

- set(CMAKE_CXX_STANDARD 14)
  set(CMAKE_CXX_STANDARD_REQUIRED ON)
  set(CMAKE_CXX_EXTENSIONS OFF)
  ⇒Specify the C++ standard: use C++ 14; compiler specific extensions are not requested.

- 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}")
  ⇒These are CMake environment variables. The flags in these variables will be passed to the complier before those in the per-configuration CMAKE_CXX_FLAGS_<CONFIG> variants.
  O0 and O3 are about different optimization degree.
  -O0: does not make any optimization.
  -O3: adopt more optimization option than O1 and O2.
  -Wall: show all warnings after compilation

- add_executable(calibration src/calibration.cpp)
  target_link_libraries(calibration Ceres::ceres pangolin TBB::tbb)
  ⇒Generate executable files. Calibration is the name of executable file, and src/calibration.cpp is the source file. Then we can get a calibration.exe in correspond folder. The project name is calibration. Then add the library Ceres::ceres pangolin and TBB::tbb to project calibration.

Part 2: SO(3) and SE(3) Lie groups

Prove: Let $\theta = \|\boldsymbol{w}\|$ and $\boldsymbol{v} = \frac{\boldsymbol{w}}{\|\boldsymbol{w}\|}$ then
$\hat{\boldsymbol{v}}^2 = \boldsymbol{v}\boldsymbol{v}^T - I, \hat{\boldsymbol{v}}^3 = -\hat{\boldsymbol{v}}$,

$$\begin{aligned}
\sum_{n=0}^{\infty} \frac{1}{(n+1)!}\hat{\boldsymbol{w}}^n &= \frac{1}{1!}\hat{\boldsymbol{w}}^0 + \frac{1}{2!}\hat{\boldsymbol{w}}^1 + \frac{1}{3!}\hat{\boldsymbol{w}}^2 + \frac{1}{4!}\hat{\boldsymbol{w}}^3 + \frac{1}{5!}\hat{\boldsymbol{w}}^4 + ... \\
&= I + (\frac{\theta^2}{2!} - \frac{\theta^4}{4!} + ...)\frac{\hat{\boldsymbol{v}}}{\theta} + (\frac{\theta^3}{3!} - \frac{\theta^5}{5!} + ...)\frac{\hat{\boldsymbol{v}}^2}{\theta} \\
&= I + \frac{1-\cos\theta}{\theta}\hat{\boldsymbol{v}} + \frac{\theta - \sin\theta}{\theta}\hat{\boldsymbol{v}}^2 \\
&= I + \frac{1-\cos\theta}{\theta^2}\hat{\boldsymbol{w}} + \frac{\theta - \sin\theta}{\theta^3}\hat{\boldsymbol{w}}^2 \\
&= \boldsymbol{J}
\end{aligned}$$

Part 3: What is SLAM?

1. Why would a SLAM system need a map?

- Support other tasks. e.g. inform path planning or provide an intuitive visualization for a human operator.
- Limit the error committed by state estimation. e.g., a set of distinguishable landmarks, the robot can "reset" its localization error by revisiting known areas using the map (so-called loop closure).

2. How can we apply SLAM technology into real-world applications?
It can be applied in all scenarios in which a prior map needs to be built from scratch.
One example is indoor applications: Indoor operation rules out the use of GPS to bound the localization error; furthermore, SLAM provides an appealing alternative to user-built maps, showing that robot operation is possible in the absence of an ad hoc localization infrastructure.

3. Describe the history of SLAM.
The history can be divided into three eras:

- Classical age (1986–2004): saw the introduction of the main probabilistic formulations for SLAM, including approaches based on extended Kalman filters (EKF), Rao–Blackwellized particle filters, and maximum likelihood estimation; moreover, it delineated the basic challenges connected to efficiency and robust data association
- Algorithmic-analysis age (2004–2015): saw the study of fundamental properties of SLAM, including observability, convergence, and consistency. In this period, the key role of sparsity toward efficient SLAM solvers was also understood, and the main open-source SLAM libraries were developed.
- Robust-perception age (2015-present), which is characterized by Robust Performance, High-Level Understanding, Resource Awareness and Task-Driven Perception

Reference

[1] Cesar Cadena et al. Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age. IEEE Transactions on Robotics, vol. 32, No. 6, Dec. 2016.