Exercise 1 Solution

Qiao Qiao

Robotics, Cognition, Intelligence - Technische Universität München

1 Exercise 1: What is SLAM

1. Why would a SLAM system need a map?

The map describes the environment in which robot operates. The map can support other applications, for example, robot motion planning. Some applications not only do the robot localization, but also want a 3D model reconstruction. On the other hand, robots can do the localization better with a map. The map can reduce error in state estimating by making loop closure possible. Without a map, SLAM reduces to odometry, which limits many abilities. For instance, finding a better solution for tasks, a shortcut between two goals. Without the map helping robot understand the environment, its performance would be greatly limited.

2. How can we apply SLAM technology into real-world applications?

Outdoor, SLAM can provide environment understanding for automatic driving. Also, it can do the 3D model reconstruction for architectures, cities. Indoor, the use of GPU has been ruled out. So, SLAM can be used for robot navigation instead. For instance, robot server in the hotel to deliver food or other supplies. Also, it can be applied in indoor scene reconstruction. SLAM can be used in many military applications that the robot explore an environment and report a map to the human operator. The robot has to perform structural inspection of a building, bridge, etc. SLAM can also be used in some augmented reality applications.

3. Describe the history of SLAM.

The development of SLAM can roughly divided into three stages. In classical age (1986-2004), researchers developed many probabilistic formulation. Approaches are based on all kinds of filters. And then is algorithmic-analysis age (2004-2015). Researchers start to study fundamental properties of SLAM, including observability, convergence, and consistency.In this period, the key role of sparsity towards efficient SLAM solvers was also understood and the main open-source SLAM libraries were developed. We are now on the robust-perception age, which requires SLAM to has robust performance, high-level understanding, resource awareness and task-driven perception. And yet, SLAM is not solved. For many applications and environments, numerous major challenges and important questions remain open.

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

- set (CMAKE_MODULE_PATH "$ CMAKE_CURRENT_SOURCE_DIR / cmake_-modules /" $ CMAKE_MODULE_PATH)
  Set the list of directories to search for CMake modules to be the cmake_modules directory under the cmake current source directory.

- set (CMAKE_CXX_STANDARD 14)
  Set C++ standard to 14 while building this target.

  set (CMAKE_CXX_STANDARD_REQUIRED ON)
  It is a boolean describing whether the value of CXX_STANDARD is a requirement. Here we set it on. It means it requires we provide the C++ standard number.

  set (CMAKE_CXX_EXTENSIONS OFF)
  We don't want to use compiler specific extensions.

- set (CMAKE_CXX_FLAGS_DEBUG "-O0 -g - DEIGEN_INITIALIZE_MATRICES_BY_NAN ")
  CXX_FLAGS is among the environment variables conventionally used to specify compiler options to a build system when compiling C++ code. -O controls the overall level of optimization. -O0 turns off optimization entirely. This command set several flags in debug mode.

  set (CMAKE_CXX_FLAGS_RELWITHDEBINFO "-O3 -DNDEBUG -g - DEIGEN_-INITIALIZE_MATRICES_BY_NAN ")
  Set flags for RelWithDebInfo type or configuration. -O3 is the highest level of optimization possible. It enables optimizations that are expensive in terms of compile time and memory usage. -DNDEBUG removes assert from the code.

  set (CMAKE_CXX_FLAGS_RELEASE "-O3 -DNDEBUG ")
  Set flags for Release build type to the highest level of optimization possible.

  SET (CMAKE_CXX_FLAGS " -ftemplate - backtrace - limit =0 -Wall -Wextra $ EXTRA_WARNING_FLAGS -march =$ CXX_MARCH $ CMAKE_-CXX_FLAGS ")
  Set several flags.
  -ftemplate - backtrace - limit =0: Set the maximum number of template instantiation notes for a single warning or error to 0. -Wall is named EnableAllWarnings.This enables all the warnings about constructions that some users consider questionable.
  -Wextra enables some extra warning flags that are not enabled by -Wall.
  -march tells the compiler what code it should produce for the system's pro-

cessor architecture.

- add_executable (calibration src / calibration . cpp)
  Add an executable to the project using calibration.cpp in folder src.

  target_link_libraries (calibration ceres pangolin TBB)
  Specify libraries ceres, pangolin, TBB to use when linking calibration.

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

$$\theta = ||w||$$

$$\begin{aligned}
\sum_{n=0}^{\infty} \frac{1}{(n+1)!}\hat{w}^n &= I + \frac{1}{2!}\hat{w} + \frac{1}{3!}\hat{w}^2 + ... \\
&= I + \sum_{n=0}^{\infty}\left[\frac{(\hat{w})^{2i}}{(2i+2)!}\hat{w} + \frac{(\hat{w})^{2i}}{(2i+3)!}\hat{w}^2\right] \\
&= I + \left(\sum_{n=0}^{\infty}\frac{(-1)^i\theta^{2i}}{(2i+2)!}\right)\hat{w} + \left(\sum_{n=0}^{\infty}\frac{(-1)^i\theta^{2i}}{(2i+3)!}\right)\hat{w}^2 \\
&= I + \Big(\frac{1}{2!} - \frac{\theta^2}{4!} + \frac{\theta^4}{6!} + ...\Big)\hat{w} + \Big(\frac{1}{3!} - \frac{\theta^2}{5!} + \frac{\theta^4}{7!} + ...\Big)\hat{w}^2 \\
&= I + \Big(\frac{1-\cos\theta}{\theta^2}\Big)\hat{w} + \Big(\frac{\theta - \sin\theta}{\theta^3}\Big)\hat{w}^2
\end{aligned} \tag{1}$$

As we already know from the Rodrigues' formula:

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

We can derive a closed-form solution of the exponential map for the SE(3) group:

$$\exp(\hat{\xi}) = \left[\begin{array}{c|c} R & Jv \\ \hline 0^T & 1 \end{array}\right]$$

let:

$$A = \frac{\sin\theta}{\theta}$$

$$B = \frac{1-\cos\theta}{\theta^2}$$

$$C = \frac{1-A}{\theta^2}$$

$$R = I + A\hat{w} + B\hat{w}^2$$

$$J = I + B\hat{w} + C\hat{w}^2$$