visual navigation exercise 01

Name: Chenguang Huang Matriculate number: 03709255

May 2019

1 Exercise 1

1.1 Why would a SLAM system need a map?

First of all, the map is needed for supporting some tasks like path planning and providing intuitive visualization for a human operator. Secondly, the map limits the errors of estimation of the state of robot. The robot can reset its localization error by revisiting known areas.

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

SLAM can be applied in real-world applications which doesn't has a prior map. In some scenarios SLAM is provided with a set of landmarks as a priori like manually-built map. In some cases GPS sensor is available in the system so the localization can be done reliably with respect to the known landmarks.

1.3 Describe the history of SLAM

The history of SLAM system can be divided into two ages: The classical age(1986-2004) and the algorithmic-analysis age(2004-2015). In the classical age the main probabilistic formulations for SLAM like extended Kalman Filters, Rao-Blackwellised Particle Filters and maximum likelihood estimation are introduced . Meanwhile, basic challenges related to efficiency and robustness of data association are stated. In the second period of the history, researchers studied the properties of SLAM including observability, convergence and consistency.

2 Exercise 2

2.1

This line adds "cmake_modules" folder in current source directory to the searching path of cmake modules.

2.2

The first line sets the C++ standard to 11 and the second line enables the standard library of C++ 11. The third line disables compiler specific extensions.

2.3

The first line sets the debug options to reduce compilation time and make debugging produce the expected results and produce debugging information in the operating system's native format. GDB can work with this debugging information. It also sets the matrix initialization in Eigen with Nan. The second line sets the release with debug information options to open level 3 optimization and produce debugging information. It also sets the matrix initialization in Eigen with Nan. The third line sets the release options to open level 3 optimization and turn off debugging and error-reporting code. The fourth line adds some compile options to set the maximum instantiation depth for template classes to 0, turn on some warning flags and some extra warning flags, set the march to certain features extensions.

2.4

The first line specifies the name of the built executable based on source "src/calibration.cpp". The second line link the executable with some libraries such as Ceres, pangolin and TBB.

3 Exercise 3

Proof

$$\sum_{n=0}^{\infty} \frac{1}{(n+1)!} (\phi^{\wedge})^n = I + \frac{1}{2!}\theta a^{\wedge} + \frac{1}{3!}\theta^2 a^{\wedge}a^{\wedge} + \frac{1}{4!}\theta^3 a^{\wedge}a^{\wedge}a^{\wedge} + \frac{1}{5!}\theta^4 a^{\wedge}a^{\wedge}a^{\wedge}a^{\wedge} + ... \tag{1}$$

$$= aa^T - \frac{\theta}{\theta}a^{\wedge}a^{\wedge} + \frac{1}{3!}\frac{\theta^3}{\theta}a^{\wedge}a^{\wedge} - \frac{1}{5!}\frac{\theta^5}{\theta}a^{\wedge}a^{\wedge} + ... \tag{2}$$

$$+ \frac{a^{\wedge}}{\theta} - \frac{a^{\wedge}}{\theta} + \frac{1}{2!}\frac{\theta^2}{\theta}a^{\wedge} - \frac{1}{4!}\frac{\theta^4}{\theta}a^{\wedge} + \frac{1}{6!}\frac{\theta^6}{\theta}a^{\wedge} + ... \tag{3}$$

$$= aa^T - \frac{\sin\theta}{\theta}(aa^T - I) - \frac{a^{\wedge}}{\theta}\cos\theta + \frac{a^{\wedge}}{\theta} \tag{4}$$

$$= \frac{\sin\theta}{\theta}I + (1 - \frac{\sin\theta}{\theta})aa^T + (1 - \cos\theta)\frac{a^{\wedge}}{\theta} \tag{5}$$

$$= \mathbf{J} \tag{6}$$