VISNAV Exercise01

Yiğit Aras Tunalı

April 2021

1 Part 1

-
  ```
  set(CMAKE_MODULE_PATH "${CMAKE_CURRENT_SOURCE_DIR}/cmake_modules/" ${CMAKE_MODULE_PATH})
  ```
  Sets up the search path for the include() and find_package() commands, the added paths are looked before the default ones.

-
  ```
  set(CMAKE_CXX_STANDARD 14)
  set(CMAKE_CXX_STANDARD_REQUIRED ON)
  set(CMAKE_CXX_EXTENSIONS OFF)
  ```
  First line sets C++ standard to 14, second line makes C++ 14 a requirement for the project and finally the last line turns off compiler specific extensions.

-
  ```
  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}")
  ```
  First line sets up the flags for the DEBUG build mode, -O0 is optimization level 0, -g flag tells the compiler to generate debug information and finally -DEIGEN_INITIALIZE_MATRICES_BY_NAN initializes Eigen matrices with NaN values. Second line similarly sets flags for RELEASE WITH DEBUG INFO build mode, with level 3 optimization from -O3 while DNDEBUG flag removes all assert statements from the code. Third line does the same things for the set flags but for the RELEASE mode. Finally last line is used to append more flags to the general flags used by cmake in this project. Ftemplate backtrace flag sets the maximum number of template instantiation notes for a single warning or error to 0. -Wall and -Wextra turns on quite a lot of the warnings from the compiler and march generates machine code for the specificed cpu types.

-
  ```
  add_executable(calibration src/calibration.cpp)
  target_link_libraries(calibration Ceres::ceres pangolin TBB)
  ```
  First command adds an executable to the project using the second argument, which is the path to the source file. Second command specifies the libraries and flags to be used when linking given target and it's dependents.

2 Part 2

Now we derive the closed-form of exponential map for $SE(3)$. We know that for $\xi = \begin{bmatrix} v^T w^T \end{bmatrix}^T$ and $\xi^\wedge \in se(3)$. Where w is the rotation part and v is the translation. Then we have

$$exp(\xi^\wedge) = \begin{bmatrix} \sum_{n=0}^{\infty} \frac{1}{n!}(w^\wedge)^n & \sum_{n=0}^{\infty} \frac{1}{(n+1)!}(w^\wedge)^n v \\ 0^T & 1 \end{bmatrix}$$

For the rotation part we can use the exponential map for $SO(3)$ and use Rodrigues' formula.

$$\sum_{n=0}^{\infty} \frac{1}{n!}(w^\wedge)^n = I + \frac{sin(|w|}{|w|}\hat{w} + \frac{1 - cos(|w|)}{|w|^2}\hat{w}^2$$

Now we show the closed form for the translation part in the exponential map for $SE(3)$. Similar to the slides we use $t = |w|$ and $v = \frac{w}{|w|}$. Moreover we will use $\hat{v}^2 = vv^T - I$ and $\hat{v}^3 = -\hat{v}$ for grouping.

$$\sum_{n=0}^{\infty} \frac{1}{(n+1)!}(w^\wedge)^n = \sum_{n=0}^{\infty} \frac{1}{(n+1)!}(v^\wedge t)^n = I + \frac{\hat{v}t}{2!} + \frac{\hat{v}^2 t}{3!} \dots$$

Then grouping up the similar terms just like the slides, using the identities written above.

$$= I + (\frac{t}{2!} - \frac{t^3}{4!} \dots)\hat{v} + (\frac{t^2}{3!} - \frac{t^4}{5!} \dots)\hat{v}^2$$

$$= I + \underbrace{\frac{t}{t}(\frac{t}{2!} - \frac{t^3}{4!} \dots)}_{\frac{1-cos(t)}{t}}\hat{v} + \underbrace{\frac{t}{t}(\frac{t^2}{3!} - \frac{t^4}{5!} \dots)}_{\frac{t-sin(t)}{t}}\hat{v}^2$$

Now re-writing using $t = |w| = \theta$ and $v = \frac{w}{|w|}$:

$$= I + \frac{1 - \cos\theta}{\theta^2}\hat{w} + \frac{\theta - \sin\theta}{\theta^3}\hat{w}^2 = J$$

3 Part 3

- Why would a SLAM system need a map?

  It is mostly required for two main reasons. First being, it can be used to support other tasks such as informing path planning or providing intuitive visualization for human operator. Second is the limiting of error commited when estimating the state of the robot and resetting the error by visiting known areas (by loop-closure).

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

  SLAM is useful for applications where there isn't an a priori map or localization that can be provided things such as GPS. For this reason in-door applications of mobile robotics.

- Describe the history of SLAM.

  The 1986-2004 time period is called the classical age. Where the main probabilistic formulations were introduce which were based on Extended Kalman Filters, Rao-Blackwellised Particle filters and maximum likelihood estimation. Next period is 2004-2015 era which is called the algorithmic-analysis age. It was the era where the properties of SLAM including observability, convergence and consistency. Also key role of sparsity towards efficient SLAM solvers were understood and the main open-source SLAM libraries were developed.