Part 1

// set a variable cmake_module_path

set (CMAKE_MODULE_PATH "${ CMAKE_CURRENT_SOURCE_DIR }/ cmake_modules /" ${ CMAKE_MODULE_PATH })

// make some standards for the CXX compiler

• set (CMAKE_CXX_STANDARD 14)

set (CMAKE_CXX_STANDARD_REQUIRED ON)

set (CMAKE_CXX_EXTENSIONS OFF)

• 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 }")

// add executable files with its location

• add_executable (calibration src / calibration . cpp)

//link the libraries to the executable files

target_link_libraries (calibration Ceres :: ceres pangolin TBB)

part 2:

$\xi = \begin{pmatrix} v \\ w \end{pmatrix} \quad \hat{\xi} = \begin{pmatrix} \hat{w} & v \\ 0^T & 0 \end{pmatrix}$

$\therefore \hat{\xi}^2 = \begin{pmatrix} \hat{w} & v \\ 0^T & 0 \end{pmatrix}\begin{pmatrix} \hat{w} & v \\ 0^T & 0 \end{pmatrix} = \begin{pmatrix} \hat{w}^2 & \hat{w}v \\ 0^T & 0 \end{pmatrix}$

$\hat{\xi}^3 = \hat{\xi}^2\hat{\xi} = \begin{pmatrix} \hat{w}^2 & \hat{w}v \\ 0^T & 0 \end{pmatrix}\begin{pmatrix} \hat{w} & v \\ 0^T & 0 \end{pmatrix} = \begin{pmatrix} \hat{w}^3 & \hat{w}^2v \\ 0^T & 0 \end{pmatrix}$

$\therefore \hat{\xi}^n = \begin{pmatrix} \hat{w}^n & \hat{w}^{n-1}v \\ 0^T & 0 \end{pmatrix}$

$\therefore \exp(\hat{\xi}) = \sum_{n=0}^{\infty} \frac{(\xi)^n}{n!} = I + \hat{\xi} + \frac{\hat{\xi}^2}{2!} + \cdots = \begin{pmatrix} I & 0 \\ 0 & 1 \end{pmatrix} + \begin{pmatrix} \hat{w} & v \\ 0^T & 0 \end{pmatrix} + \frac{1}{2!}\begin{pmatrix} \hat{w}^2 & \hat{w}v \\ 0^T & 0 \end{pmatrix} + \cdots$

$= \begin{pmatrix} I + \hat{w} + \frac{1}{2!}\hat{w}^2 + \cdots & v + \frac{1}{2!}\hat{w}v + \frac{1}{3!}\hat{w}^2v + \cdots \\ 0^T & 1 \end{pmatrix}$

$= \begin{pmatrix} \sum_{n=0}^{\infty} \frac{1}{n!}(\hat{w})^n & \sum_{n=0}^{\infty} \frac{1}{(n+1)!}(\hat{w})^n v \\ 0^T & 1 \end{pmatrix}$

for $\sum_{n=0}^{\infty} \frac{1}{(n+1)!}(\hat{w})^n v$ let $\hat{w} = \|\hat{w}\| \cdot \frac{\hat{w}}{\|\hat{w}\|} = \theta \cdot \hat{a}$ $\quad \begin{cases} \theta = \|w\| \\ \hat{a} = \frac{\hat{w}}{\|w\|} \end{cases}$ $\quad \hat{a}^2 = aa^T - I \quad \hat{a}^3 = -\hat{a}$

$\therefore = \sum_{n=0}^{\infty} \frac{(\theta\hat{a})^n}{(n+1)!} = I + \frac{1}{2!}\theta\cdot\hat{a} + \frac{1}{3!}\theta^2\hat{a}^2 + \cdots$

$= I + \frac{1}{\theta}\left(\frac{\theta^2}{2!}\hat{a} + \frac{\theta^4}{4!}\hat{a}^3 + \cdots\right) + \frac{1}{\theta}\left(\frac{\theta^3}{3!}\hat{a}^2 + \frac{\theta^5}{5!}\hat{a}^4 + \cdots\right)$

$= I + \frac{1}{\theta}\left(\frac{\theta^2}{2!} - \frac{\theta^4}{4!} + \cdots\right)\hat{a} + \frac{1}{\theta}\left(\frac{\theta^3}{3!} - \frac{\theta^5}{5!} \cdots\right)\hat{a}^2$

recall: $\sin\theta = \theta - \frac{\theta^3}{3!} + \frac{\theta^5}{5!} - \cdots$

$\cos\theta = 1 - \frac{\theta^2}{2!} + \frac{\theta^4}{4!} - \cdots$

$= I + \frac{1}{\theta}(1-\cos\theta)\hat{a} + \frac{1}{\theta}(\theta - \sin\theta)\hat{a}^2$

$= I + \frac{1}{\theta}(1-\cos\theta)\cdot\frac{\hat{w}}{\theta} + \frac{1}{\theta}(\theta - \sin\theta)\cdot\frac{(\hat{w})^2}{\theta^2}$

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

Part 3

1. Why would a SLAM system need a map?

First, the map is often required to support other tasks; for instance, a map can inform path planning or provide an intuitive visualization for a human operator. Second, the map allows limiting the error committed in estimating the state of the robot. In the absence of a map, dead reckoning would quickly drift over time; on the other hand, using a map, e.g., a set of distinguishable landmarks, the robot can "reset" its localization error by re-visiting known areas (so-called loop closure). Therefore, SLAM finds applications in all scenarios in which a prior map is not available and needs to be built.

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

Either implicitly or explicitly, do require a globally consistent map. For instance, in many military and civilian applications, the goal of the robot is to explore an environment and report a map to the human operator, ensuring that full coverage of the environment has been obtained. Another example is the case in which the robot has to perform structural inspection (of a building, bridge, etc.); also in this case a globally consistent 3D reconstruction is a requirement for successful operation.

3. Describe the history of SLAM.

classical age (1986-2004); the classical age saw the introduction of the main probabilistic formulations for SLAM, including approaches based on Extended Kalman Filters, Rao- Blackwellised Particle Filters, and maximum likelihood estimation; moreover, it delineated the basic challenges connected to efficiency and robust data association.

algorithmic-analysis age (2004-2015), The algorithmic analysis period saw the study of 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.

The papaer argue that we are entering in a third era for SLAM, the **robust-perception age**, which is characterized by the following key requirements:

1. robust performance: the SLAM system operates with low failure rate for an extended period of time in a broad set of environments; the system includes fail-safe mechanisms and has self-tuning capabilities1 in that it can adapt the selection of the system parameters to the scenario.
2. high-level understanding: the SLAM system goes beyond basic geometry reconstruction to obtain a high-level understanding of the environment (e.g., high-level geometry, senmantics, physics, affordance
3. resource awareness: the SLAM system is tailored to the available sensing and computational resources, and provides means to adjust the computation load depending on the available resources;
4. task-driven perception: the SLAM system is able to select relevant perceptual information and filter out irrelevant sensor data, in order to support the task the

robot has to perform; moreover, the SLAM system produces adaptive map representations, whose complexity may vary depending on the task at hand.