Exercise1 – Yi-Hsiang Fang

Part 1

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

Set a path for CMake modules to be loaded by the include() or find_package() commands before checking the default modules that come with CMake

set (CMAKE_CXX_STANDARD 14)

Set C++14 on all tartgets

set (CMAKE_CXX_STANDARD_REQUIRED ON)

To prevent CMake fallback behavior

set (CMAKE_CXX_EXTENSIONS OFF)

This results in modifying the flag used to set the language standard. The default behavior is for C++ extensions to be enabled, but for broadest compatibility across compilers, projects should prefer to explicitly disable them.

set (CMAKE_CXX_FLAGS_DEBUG "-O0 -g - DEIGEN_INITIALIZE_MATRICES_BY_NAN ")

Flags used by the compiler during debug builds.
The build system will compile the tools and libraries un-optimized, with debugging information, and asserts enabled.

set (CMAKE_CXX_FLAGS_RELWITHDEBINFO "-O3 -DNDEBUG -g -

DEIGEN_INITIALIZE_MATRICES_BY_NAN ")

Set the optimized binaries with debug information

set (CMAKE_CXX_FLAGS_RELEASE "-O3 -DNDEBUG ")

This command allow the build syste compile the tools and libraries with optimizations enabled and not generate debug info.

SET (CMAKE_CXX_FLAGS " -ftemplate - backtrace - limit =0 -Wall -Wextra ${

EXTRA_WARNING_FLAGS } -march =${ CXX_MARCH } ${ CMAKE_CXX_FLAGS }")

Set the content as the compilation flags

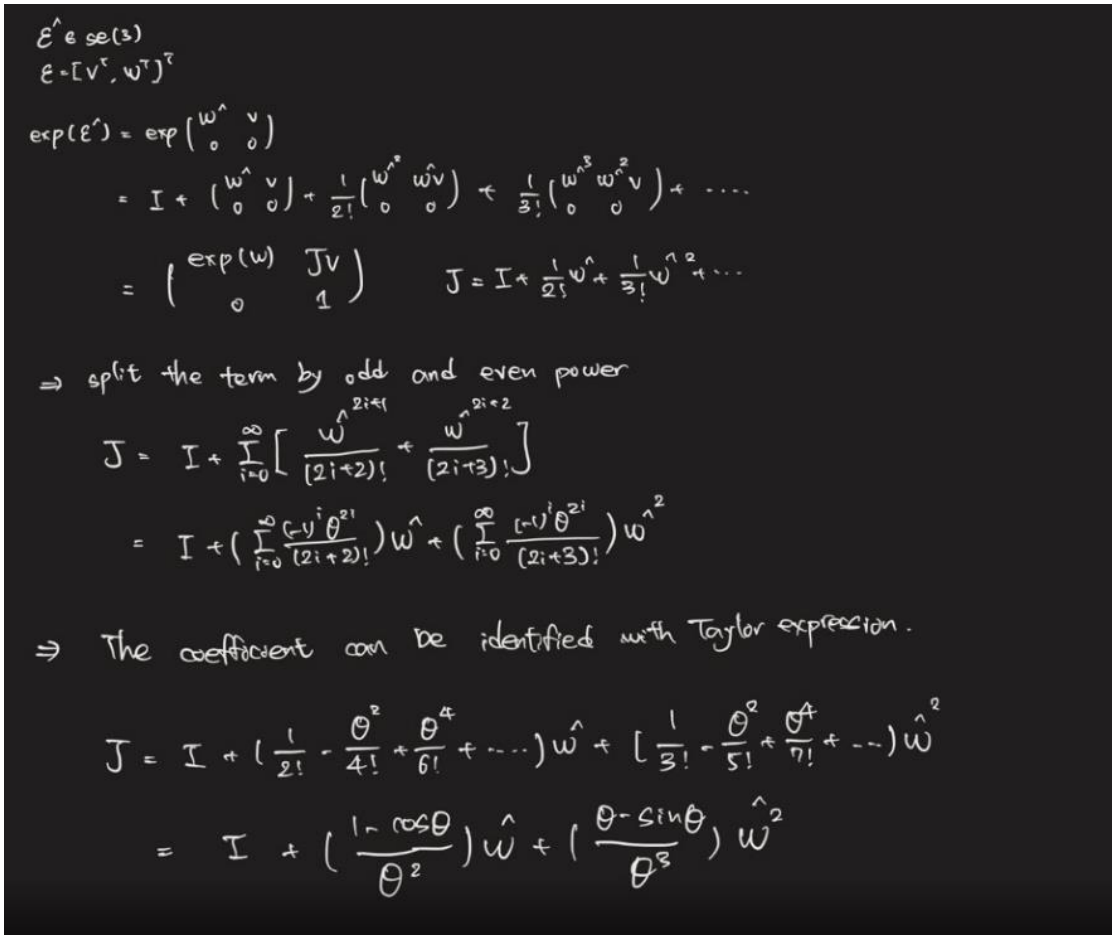
add_executable (calibration src / calibration . cpp)

Add an executable to the project using the calibration.cpp

target_link_libraries (calibration Ceres :: ceres pangolin TBB)

Add libraries Ceres to the target calibration.

Part 2



Part 3

1. Why would a SLAM system need a map?

The map, on the other hand, is a representation of aspects of interest (e.g., position of landmarks, obstacles) describing the environment in which the robot operates. The need to use a map of the environment is twofold. 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 revisiting 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?

First of all, SLAM research done over the last decade has itself produced the visual-inertial odometry algorithms that currently represent the state-of-the-art. SLAM has directly led to the study of sensor fusion under more challenging setups (i.e., no GPS, low quality sensors) than previously considered in other literature (e.g., inertial navigation in aerospace engineering).

Also, SLAM help finding loop closures. Therefore, the robot can understand the real topology of the environment, and is able to find shortcuts between locations
The second answer regards the true topology of the environment. As the result, the SLAM map provides a way to predict and validate future measurements: we believe that this mechanism is key to robust operation.
By the advantage that provided by SLAM, SLAM can use in many real-world applications. 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 three-dimensional (3-D) reconstruction is a requirement for successful operation.

3. Describe the history of SLAM.

The First stage of SLAM is called the classical age which start from 1986 and end at 2004. The classical age 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.

The second stage of SLAM is called the 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 toward efficient SLAM solvers was also understood, and the main open-source SLAM libraries were developed.