Date: 26 May -30 May

Time In:

Work Done during the day:

The first week of my internship was both introductory and insightful. I had the opportunity to visit the BARC (Bhabha Atomic Research Centre) campus. I was introduced to my division of work for internship in DRHR (Division of Remote Handling and Robotics). I’m working on Kimera-VIO which is a real-time visual-inertial odometry system that estimates a robot's 3D pose by combining camera(visual) and IMU data. It is used in applications like autonomous robots, drones, and AR/VR, especially where GPS is unavailable. The system tracks visual features using various ML algorithms using camera (mono or stereo) and uses IMU for motion estimation, and refines the trajectory through optimization. It runs on ROS in Linux operating system mainly Ubuntu and is suitable for real-time robotic navigation and mapping (SLAM).

Comments by the supervisor:

June 2  
I began my technical work by reviewing the official documentation and GitHub repositories related to Kimera-VIO and its ROS wrapper. I explored the purpose and structure of the main components OF Kimera VIO including the visual-inertial frontend, optimization backend, loop closure module, and the integration with Kimera-RPGO. I went through the README files, installation instructions, configuration settings, and example commands provided in the repositories to understand the dependencies and expected workflow. I also had a detailed discussion with my guides to clarify the overall objectives of the internship, my role in the project, and how my tasks would contribute to the larger goals of the Division of Remote Handling and Robotics (DRHR).

June 3

Learnt about the project pipeline and the package repositories. Went through the codes in each folder in the ROS workspace.

My technical work began with learning to run the already built development environment for Kimera-VIO that integrates with ROS using Kimera-VIO-ROS wrapper on the laptop given by my guide. I familiarized myself with the system's architecture, including its main pipeline—frontend for visual feature tracking, backend for IMU-based pose estimation and optimization, and the loop closure detector (LCD) for reducing long-term error drift.

June 4  
I focused on understanding the specific machine learning and computer vision algorithms used in the frontend of Kimera-VIO. This module is responsible for detecting and tracking visual features across frames, which are essential for estimating motion. Through online research and watching youtube videos I studied key techniques such as ORB (Oriented FAST and Rotated BRIEF) for feature detection and description, and how these features are matched between frames to compute stereo depth and motion estimates. I also explored how these tracked features are passed to the backend for fusion with IMU data. This helped me gain a clearer picture of how the visual frontend contributes to inertial odometry in SLAM systems like Kimera.

June 5

I explored the concept of IMU preintegration, which is a key part of the Kimera-VIO frontend used to incorporate high-frequency inertial measurements into the visual-inertial fusion process. I referred to the research papers and resources linked in the Kimera-VIO GitHub repository to understand how preintegration allows efficient use of IMU data between keyframes, without requiring full integration from the last known state each time. This improves computational performance while maintaining accuracy in motion estimation. After reinforcing my theoretical understanding, I proceeded to run the already-built Kimera-VIO workspace on the provided laptop using terminal commands and verified its basic execution flow.

June 6

I focused on understanding the Kimera-RPGO (Robust Pose Graph Optimization) module, which is used to refine the global pose trajectory by integrating loop closure constraints. I studied how RPGO leverages techniques like Dynamic Covariance Scaling (DCS) to prevent the influence of incorrect loop closures and maintain global consistency in the pose graph. This robustness is critical in real-world environments where false-positive loop closures can corrupt trajectory estimation. I reviewed the role of RPGO in the Kimera pipeline and how it connects with the loop closure detector and backend optimizer to perform global optimization reliably.

June 9

I explored the various dataset sequences available for visual-inertial odometry benchmarking, with a focus on the EuRoC and KITTI datasets. I studied the key differences between them—EuRoC provides synchronized stereo images and IMU data with accurate ground truth, while KITTI primarily focuses on outdoor driving scenarios with LiDAR and GPS data. I noted that EuRoC is fully compatible with Kimera-VIO due to its synchronized visual-inertial format, whereas KITTI is less suited without modification. I also began reviewing the Kimera-VIO source code and documentation to understand how dataset parameters are configured and managed within the ROS workspace.

June 10

I shifted focus to the loop closure detection (LCD) module of Kimera-VIO, which plays a crucial role in minimizing long-term drift by recognizing previously visited locations. I learned that it uses ORB feature descriptors along with a Bag-of-Words (BoW) model to perform place recognition. Upon detecting a valid loop closure, it sends a constraint to the pose graph to reduce drift and correct the path using Kimera-RPGO package. To deepen my understanding, I read through related research papers and the LCD section of the Kimera documentation.

June 11

Downloaded the different Euroc dataset rosbags like V1\_01, V2\_01 and MH\_01. These are the easy datasets of Euroc. Ran these using the Kimera-VIO-ROS wrapper. 3 commands needed to run it, 1st one being the roslaunch command to launch the main code kimera\_vio\_euroc.launch, other one being rosbag play which plays the euroc rosbag downloaded and the last one to launch the rviz environment for visualization. The V series ran successfully but the MH series was causing problem and crashing in between.

June 12

Downloaded all the other euroc rosbag sequences like V1\_02, V1\_03, V2\_02, V2\_03, MH\_02, MH\_03, MH\_04, MH\_05. These are all medium and difficult euroc datasets except for MH\_02 which is the easy bag. It took a long time to download. Till the download I was figuring out the problem with the MH dataset. Looked into codes and the parameters file of kimera vio in the params folder. It consists of …… Leart about the evaluation methods to check if the output I’m getting after running the euroc datasets on Kimera VIO is with minimal error. There are 2 parameters ATE and RTE to evaluate the result.

June 13

Created a new launch file to run the 2 commands out of the 3 commands by 1 command to make the steps required less to run it each time. The main launch file and the rviz I compiled into 1 launch file and othe command to play rosbag I kept it different to make it easier to see the errors in console log in terminal. Ran some of the downloaded datasets successfully while some gave error. The errors were due to the default parameters set which needs to be changes and tested.

June 16

June 17

June 18

June 19

June 20

Started building the whole new workspace from scratch with newer github commit packages, the latest one as the program if enabled lcd in the previous one was crashing. I saw the issues mentioned in github and saw it was a common problem and it was mentioned it may be because of older version and there were heavy rewrites in the package code after that. So am trying a newer version if it solves the crashing problem.

June 23

Chose and downloaded newest package and while building faced compatibility issues with other packages in the same new workspace with the same library dependencies from previous workspace. Hus went through github issues and github commit history to check package compatibility according to commit number and tag versions.

June 24

Faced many errors while building like gcc compatibility errors so opted for a little older version packages for kimera vio. Also updated the Kimera RPGO package to latest one which compiled without errors. After resolving errors for Kimera VIO package while compiling the ROS wrapper for it which I had kept the same were giving errors. Thus I updated this package too but still giving some compatibility errors so changed things in code according to the Kimera VIO package compatibility based on errors. The catkin build command to build for all libraries take more than 30 minutes and for Kimera VIO itself it takes 20 min. So it took long to build and resolve errors.

June 25

Built the new workspace with latest commit which is compatible with the library dependencies like gtsam, opencv and the other packages like RPGO and ROS wrapper on the Ubuntu version I’m working on successfully. Started testing it with different Euroc datasets. Without lcd error decreased compared to the previous older version package.

June 26