RBE/CS549: Deep and Un-Deep Visual Inertial Odometry

Shrishailya Chavan
WPI Robotics Engineering
Worcester Polytechnic Institute
schavan@wpi.edu

Sreejani Chatterjee
WPI Robotics Engineering
Worcester Polytechnic Institute
schatterjee@wpi.edu

***Abstract*—This paper presents a visual-inertial odometry system that combines camera and IMU data using the Multi-State Constraint Kalman Filter (S-MSCKF) approach. The proposed system achieves reliable and accurate odometry for a robot by leveraging the complementary strengths of a camera and an IMU. The S-MSCKF approach is a state-of-the-art technique that uses a stereo camera and an IMU to estimate the robot's motion and location with high precision. The system is particularly useful for Micro Aerial Vehicle (MAV) platforms, as it can operate effectively in GPS-denied environments and requires a smaller and lighter sensor package than lidar-based techniques. The experimental results demonstrate the effectiveness of the proposed system in achieving robust odometry for a robot in challenging environments.**

***Index Terms*—Multi-State Constraint Kalman Filter, S-MSCKF, Robust Odometry, IMU, Stereo**

I. PHASE 1: CLASSICAL VISUAL-INERTIAL ODOMETRY

A. Introduction:

In this project, we implemented the visual-inertial odometry system proposed in the paper "Robust Stereo Visual Inertial Odometry for Fast Autonomous Flight." To develop this system, we referred to the mathematical model presented in the paper "A Multi-State Constraint Kalman Filter for Vision-aided Inertial Navigation." However, due to the complexity of the technique, we had to implement several functions ourselves, such as initialize_gravity_and_bias, batch_imu_processing, process_model, predict_new_state, state_augmentation, add_feature_observations, measurement_update, and predict_new_state (using 4th order RungeKutta).

The msckf.py file contains all the implemented functions. These functions enable the system to estimate the gravity and bias using initial IMU measurements, process the IMU messages, and update the state of the robot. Additionally, they add new camera and image feature observations to the state and perform measurement updates to refine the state estimates. Finally, the predict_new_state function propagates the state using 4th order RungeKutta. The effectiveness of the proposed system was evaluated through experiments, and the results demonstrated its robustness in achieving accurate odometry for a robot in various environments.

B. Dataset:

To evaluate the performance of our implemented visual-inertial odometry system, we utilized the Machine Hall 01 easy subset of the EuRoC dataset, denoted as MH_01_easy. This dataset comprises VI sensor data collected by a quadrotor flying along a specific trajectory, with ground truth measurements provided by a sub-mm accurate Vicon Motion capture system. By using this dataset, we were able to test our system's ability to accurately estimate the robot's motion and location in various conditions, and compare it with the ground truth measurements to evaluate its effectiveness.

C. Error Calculation

The RMSE ATE (Absolute Trajectory Error) is calculated using the RPG repository provided. The RPG repository implements commonly used trajectory evaluation methods for visual-inertial odometry. The results are illustrated in the Results below.

To run the toolbox, the ground truth text files must first be converted to the required file formats. Additionally, the estimated IMU states are added to a CSV file, which is then converted to a .txt file. The .yaml file is modified to specify SE3 alignment.

D. Results

Fig. 1. Initial Camera Position

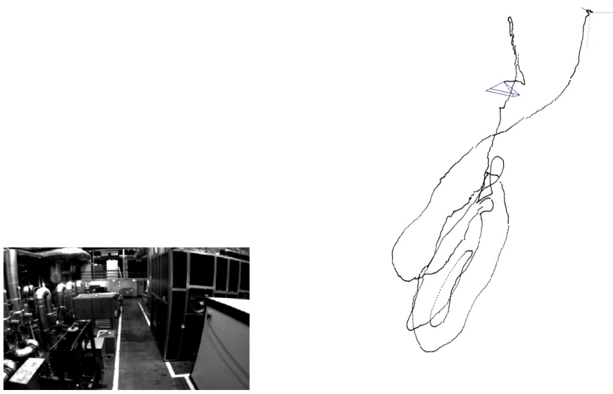

Fig. 2. Final Output

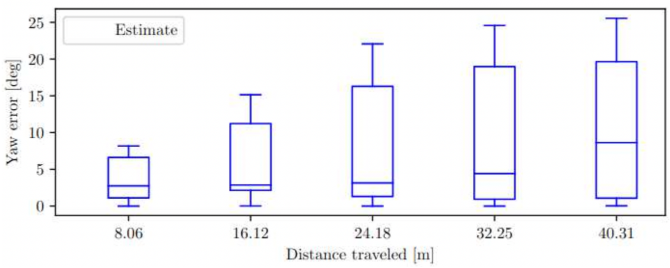

Fig. 3. Relative Yaw error

REFERENCES

[1] https://www.darpa.mil/program/fast-lightweight-autonomy
[2] https://www.youtube.com/watch?v=jxfJFgzmNSw&t=3s
[3] https://nitinjsanket.github.io/tutorials.html
[4] https://arxiv.org/abs/1712.00036
[5] https://www-users.cse.umn.edu/ stergios/papers/ICRA07-MSCKF.pdf
[6] https://projects.asl.ethz.ch/datasets/doku.php?id=kmavvisualinertialdatasets

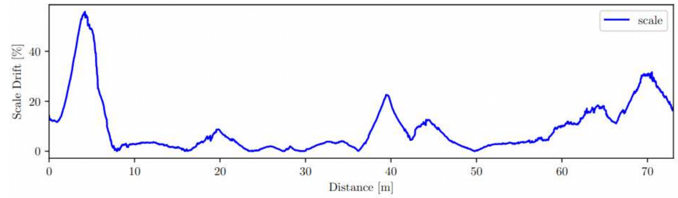

Fig. 4. Scale Error

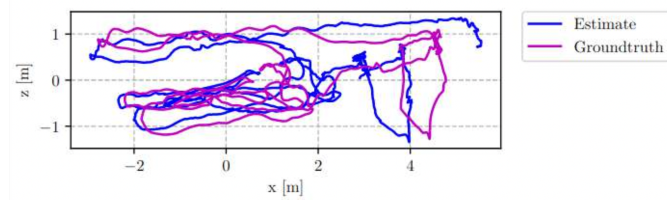

Fig. 5. Side trajectory plot

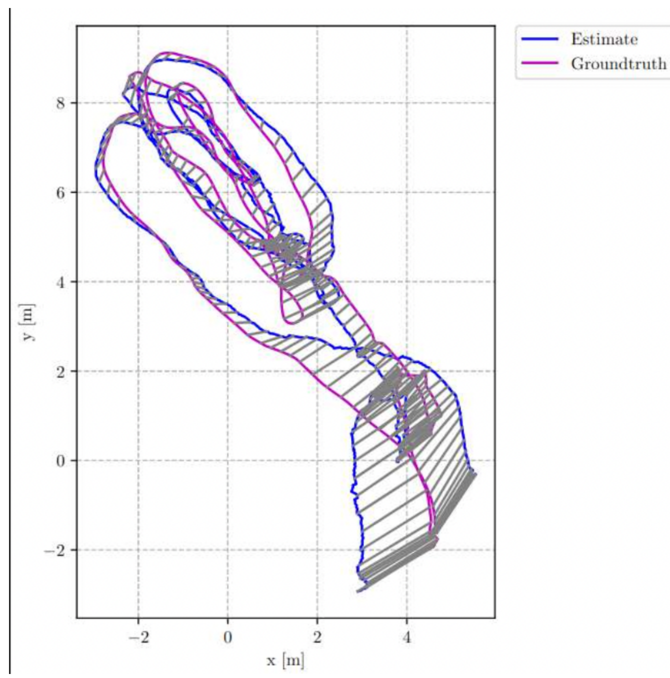

Fig. 6. Top trajectory plot

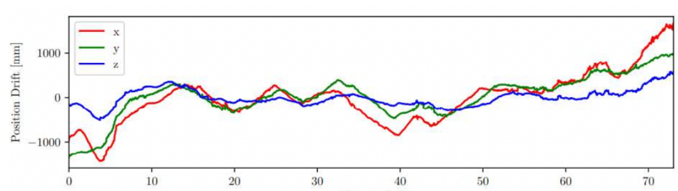

Fig. 7. Translation error