RBE 549 Computer Vision
Project 4 - Phase 1
Visual and Inertial Odometry

Taruneswar Ramuu
Robotics Engineering Department
Worcester Polytechnic Institute
Worcester, MA, USA
Email: tramuu@wpi.edu

Soumik Saswat Patnaik
Robotics Engineering Department
Worcester Polytechnic Institute
Worcester, MA, USA
Email: sspatnaik@wpi.edu

***Abstract*—This report describes the implementation of a stereo visual-inertial odometry algorithm using the Multi-State Constraint Kalman Filter (MSCKF). The project builds upon an existing codebase provided as a starting point. It elaborates on the mathematical principles that form the basis of the stereo MSCKF algorithm and explains its integration into the existing codebase. The report also documents the results and observations at each stage of the implementation.**

***Index Terms*—Multi-State Constraint Kalman Filter (MSCKF), Stereo Vision-aided Odometry, Visual Inertial Odometry, Sensor Fusion.**

I. *INTRODUCTION*

The primary objective of this project is to estimate depth from images, a challenging task given that depth cannot be directly derived from a single camera without prior environmental knowledge. While a stereo camera with a known pose can estimate depth through feature matching, this approach has limitations including computational expense and susceptibility to motion blur. To address these challenges, the project integrates an Inertial Measurement Unit (IMU), which measures linear and angular acceleration. The IMU excels in capturing fast movements and jerks where cameras struggle, although it may drift over time, an area where cameras perform better. Leveraging the complementary nature of these systems, the project aims to achieve accurate pose estimation and backtrack depth.

In the context of robotic advancements, particularly in aerial vehicles, state estimation is paramount for pose acquisition and stability during flight. Visual Inertial Odometry (VIO) emerges as a solution, combining visual data from cameras with IMU measurements. This project implements VIO using the Multi-State Constraint Kalman Filter (MSCKF) to address the high costs associated with sensors and processing in traditional VIO implementations. By employing MSCKF, the project seeks to determine the state and localization of the robot through sensor fusion of the IMU and a stereo camera. The approach involves modifying Python starter code from the authors of the S-MSCKF paper to implement key functions and testing the algorithm on the Machine Hall 01 easy subset of the EuRoC dataset.

II. *DATASET*

The implementation of this project utilized the Machine Hall 01 easy subset of the EuRoC dataset, collected onboard a Micro Aerial Vehicle (MAV) in flight. This dataset encompasses stereo images, synchronized IMU measurements, and precise ground truth data for motion and structure. The ground truth is provided by a sub-mm accurate Vicon Motion capture system. The VI sensor carried by the quadrotor captures the data as it flies along a designated trajectory, enabling rigorous testing and validation of the implemented algorithm.

III. *FUNCTION IMPLEMENTATIONS*

A. Initialize Gravity and Bias

The 6-DOF IMU sensor, providing measurements for rotation (gyroscope) and acceleration (accelerometer), necessitates calibration to mitigate biases. This calibration involves determining the mean of stationary readings to identify biases, which are subsequently subtracted from future readings. Ideally, the gyroscope should read [0, 0, 0] with minor fluctuations, while the accelerometer should register [0, 0, -g] in the world frame, albeit with potential noise-induced fluctuations.

Prior to flight commencement, calibration is executed to correct biases in both gyroscope and accelerometer outputs. The ”initialize gravity and bias” function is designed for this purpose, establishing the IMU’s gravity and bias along with the robot’s initial orientation based on initial IMU readings. This function computes the gyro bias and estimates gravity within the IMU frame by averaging angular velocity and linear acceleration data from the IMU messages, respectively. The initial orientation aligns with the inertial frame, ensuring a robust initiation for the Visual-Inertial Odometry system.

B. Batch IMU Processing

The IMU batch processing function serves as a pivotal component within the Visual-Inertial Odometry system, responsible for reading IMU messages up to the point where

the subsequent stereo camera images become available. The state vector employed for predicting the forthcoming states encompasses parameters pertinent to both the camera and IMU systems. These include quaternion representations for rotation, gyroscope and accelerometer biases, as well as positional and velocity components.

$$\mathbf{x}_I = \begin{pmatrix} {}^I_G\mathbf{q}^\top & \mathbf{b}_g^\top & {}^G\mathbf{v}_I^\top & \mathbf{b}_a^\top & {}^G\mathbf{p}_I^\top & {}^I_C\mathbf{q}^\top & {}^I\mathbf{p}_C^\top \end{pmatrix}^\top$$

Fig. 1: *State Vector*

The "batch imu processing" function operates within a predefined time window, facilitating the propagation of the IMU state by processing the accumulated IMU measurements. This function sequentially iterates through the IMU message buffer, filtering out previously processed messages and halting at the designated time boundary. For each unprocessed IMU message encountered, the function employs the process model to update the IMU state using the angular velocity and linear acceleration data. Subsequently, the timestamp and ID of the IMU state are updated, and the processed messages are purged from the buffer.

This meticulous processing mechanism ensures precise state propagation and synchronization between the IMU and Visual Odometry subsystems. Such synchronization is imperative for robust sensor fusion and localization, underscoring the function's significance in achieving reliable and accurate system performance.

C. Process Model

The *process model* function is instrumental in advancing the system state and its associated covariance through a 4th order Runge-Kutta integration technique. Initially, it distills pertinent details from the existing system state, notably the IMU's status encompassing orientation, velocity, position, and biases from both gyroscope and accelerometer readings. Subsequently, the function determines the time step by referencing the provided time and the IMU state's timestamp.

$$\begin{aligned} {}^I_G\dot{\hat{\mathbf{q}}} &= \frac{1}{2}\Omega(\hat{\boldsymbol{\omega}}){}^I_G\hat{\mathbf{q}}, \quad \dot{\hat{\mathbf{b}}}_g = \mathbf{0}_{3\times1}, \\ {}^G\dot{\hat{\mathbf{v}}} &= C\left({}^I_G\hat{\mathbf{q}}\right)^\top \hat{\mathbf{a}} + {}^G\mathbf{g}, \\ \dot{\hat{b}}_a &= \mathbf{0}_{3\times1}, \quad {}^G\dot{\hat{\mathbf{p}}}_I = {}^G\hat{\mathbf{v}}, \\ {}^I_C\dot{\hat{\mathbf{q}}} &= \mathbf{0}_{3\times1}, \quad {}^I\dot{\hat{\mathbf{p}}}_C = \mathbf{0}_{3\times1} \end{aligned}$$

Fig. 2: *Dynamics of IMU*

Following this, the function calculates the discrete transition matrix (F) and the noise covariance matrix (G) to encapsulate the system's dynamics and noise characteristics, respectively. The transition matrix is derived using a 3rd order matrix exponential method, under the assumption of a minimal time step (dt). Intermediate matrices, namely Fdt, Fdt^2, and Fdt^3, are computed to facilitate this computation.

Utilizing the 4th order Runge-Kutta method, the function predicts the forthcoming system state, invoking the "predict new state" function. This operation primarily refines the system state based on gyroscope and accelerometer inputs alongside prevailing state estimates. Concurrently, adjustments are made to the transition matrix, Φ, to incorporate the null space – representing states not directly observable through sensor measurements.

$$\mathbf{F} = \begin{pmatrix} -\lfloor\hat{\boldsymbol{\omega}}_\times\rfloor & -\mathbf{I}_3 & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ -C\left({}^I_G\hat{\mathbf{q}}\right)^\top\lfloor\hat{\mathbf{a}}_\times\rfloor & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & -C\left({}^I_G\hat{\mathbf{q}}\right)^\top & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{I}_3 & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \end{pmatrix}$$

$$\mathbf{G} = \begin{pmatrix} -\mathbf{I}_3 & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{I}_3 & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & -C\left({}^I_G\hat{\mathbf{q}}\right)^\top & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{I}_3 \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \end{pmatrix}$$

Fig. 3: *Discrete Transition and Noise Covariance Matrix*

The state covariance matrix, Q, undergoes updates employing Φ, G, and a continuous noise covariance matrix, capturing the inherent uncertainties in the system model. This update extends to the covariance interlinking IMU and potential camera states. To ensure symmetry, the state covariance matrix is symmetrically adjusted by averaging it with its transpose.

Conclusively, the IMU state is refreshed with the current orientation, position, and velocity metrics, laying the groundwork as null space values for subsequent iterations of the state estimation process.

D. Predict New State

Upon acquiring the current system state, we employ the fourth-order Runge-Kutta method to advance the state and anticipate its subsequent value. This is achieved through a dedicated function, termed "predict new state", which assimilates the time step ($d\tau$), gyroscope readings, and acceleration data corresponding to the present state.

Initially, we compute the normalized error state for the angular velocity data, followed by the derivation of the Ω matrix through a systematic procedure. Concurrently, the orientation, velocity, and position data are extracted from the IMU state server. Utilizing the current state and Ω values, we compute the angular velocity and acceleration, subsequently approximating them via the Runge-Kutta methodology.

$$\Omega(\hat{\omega}) = \begin{bmatrix} -[\hat{\omega_\times}] & \omega \\ -\omega^T & 0 \end{bmatrix}$$

Fig. 4: *Omega Matrix*

Upon deriving the estimated orientation, it is transformed into quaternion form. This quaternion representation is lever-

aged to refine the velocity and position metrics within the current IMU state. These refined values serve as the updated state information, thereby informing the subsequent state determination in the sequential process. The intermediate K values are calculated as follows:-

$$k1 = f(t_n, y_n)$$
$$k2 = f(t_n + \frac{d\tau}{2}, y_n + k1 * \frac{d\tau}{2})$$
$$k3 = f(t_n + \frac{d\tau}{2}, y_n + k2 * \frac{d\tau}{2})$$
$$k4 = f(t_n + d\tau, y_n + k3 * d\tau)$$

Fig. 5: *Intermediate Values of K*

E. State Augmentation

The "state augmentation" function orchestrates the integration of a new camera state into the existing state server. This involves updating the covariance matrix and ensuring its symmetry upon the inclusion of new image data. The function calculates the rotation and translation parameters between the IMU and the camera, subsequently updating the camera state. Concurrently, adjustments are made to the covariance matrix to accommodate this new state. This pivotal step ensures coherence between the IMU and camera states, bolstering the integrity of the Inertial Navigation System (INS) implementation.

$$\mathbf{J} = \begin{pmatrix} \mathbf{J}_I & \mathbf{0}_{6\times 6N} \end{pmatrix}$$

$$\mathbf{J}_I = \begin{pmatrix} C\left({}^I_G\hat{\mathbf{q}}\right) & \mathbf{0}_{3\times 9} & \mathbf{0}_{3\times 3} & \mathbf{I}_3 & \mathbf{0}_{3\times 3} \\ -C\left({}^I_G\hat{\mathbf{q}}\right)^\top \lfloor {}^I\hat{\mathbf{p}}_{C\times} \rfloor & \mathbf{0}_{3\times 9} & \mathbf{I}_3 & \mathbf{0}_{3\times 3} & \mathbf{I}_3 \end{pmatrix}$$

Fig. 6: *State Augmentation Jacobian*

F. Adding Feature Observation

The "add feature observations" function integrates feature observations from a new image frame into the map server within a visual-inertial odometry framework. It initiates new map features for previously unseen elements, refines observations for pre-existing features, and computes the tracking rate to assess system performance.

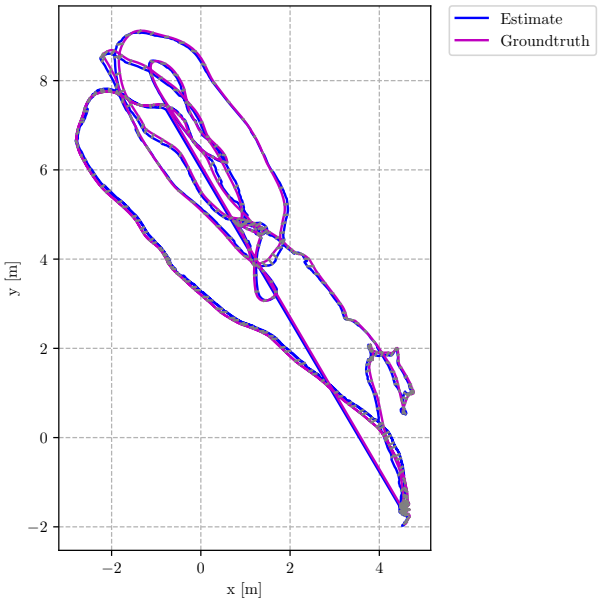
G. Measurement Update

The function leverages a measurement model to refine state estimates through the calculation of a residual, r, which linearly correlates with state errors. To streamline computations, we employ QR decomposition to simplify the Jacobian matrix. Following this simplification, we compute the Kalman gain and derive the state error.

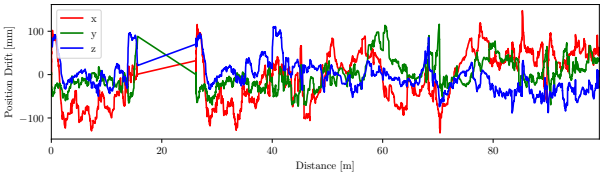
With this state error in hand, we prioritize the update of the IMU state, subsequently refining the camera states. Concluding this process, we update the state covariance, ensuring the covariance matrix maintains symmetry.

IV. ***RESULTS***

This project utilizes input data from the MH01easy subset of the EuRoC dataset. The output generated by the project for this data is depicted in the accompanying figure and is consistent with the anticipated output.



((a)) Trajectory Plot (Top View) in SE3



((b)) Translation error in SE3

Fig. 7: Error Plots from rpg toolbox

REFERENCES

[1] Ke Sun, Kartik Mohta, Bernd Pfrommer, Michael Watterson, Sikang Liu, Yash Mulgaonkar, Camillo J. Taylor, and Vijay Kumar, "Robust Stereo Visual Inertial Odometry for Fast Autonomous Flight."

[2] Anastasios I. Mourikis and Stergios I. Roumeliotis, "A Multi-State Constraint Kalman Filter for Vision-aided Inertial Navigation."

[3] "StereoMSCKF: Stereo Multi-State Constraint Kalman Filter," GitHub repository. [Online]. Available: https://github.com/uoip/stereomsckf.

[4] "RPG Trajectory Evaluation," GitHub repository. [Online]. Available: https://github.com/uzh-rpg/rpgtrajectoryevaluation.

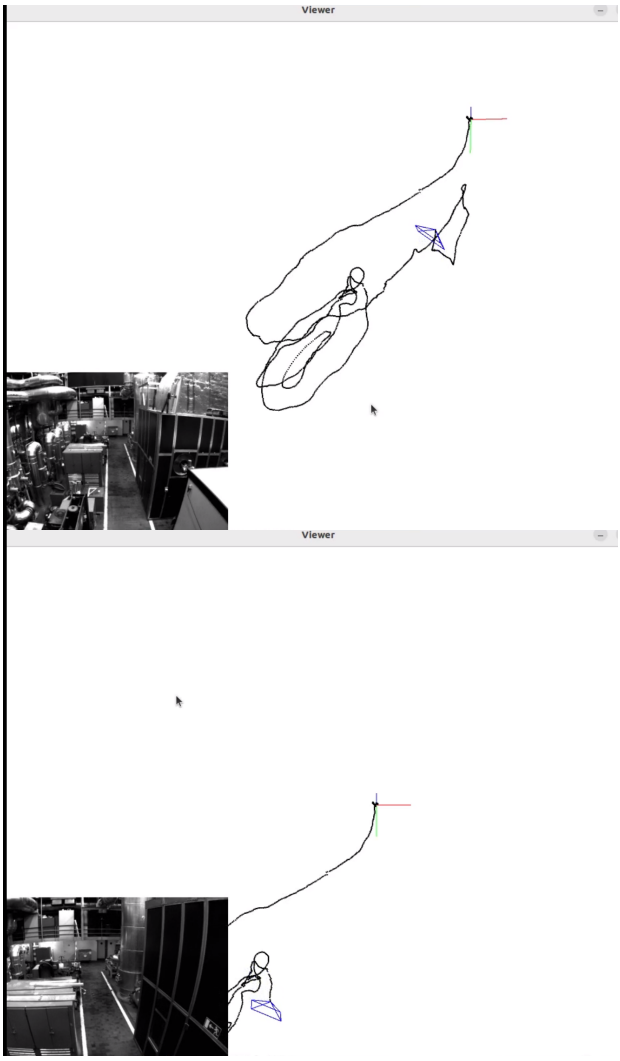


Fig. 8: Trajectory Visualization