P4: Un-deep VIO(Phase 1)

1st Venkateshkrishna
Masters in Robotics
Worcester Polytechnic Institute
Worcester, MA 01609
vparsuram@wpi.edu

2nd Mayank, Bansal
Masters in Robotics
Worcester Polytechnic Institute
Worcester, MA 01609
mbansal1@wpi.edu

***Abstract*—Phase 1 is centered on establishing a stereo vision-aided odometry system by using a Multi-state Constraint Kalman Filter (MSCKF). The project employs a strategy that combines data from a stereo camera and an Inertial Measurement Unit (IMU). We have developed eight distinct functions as part of the MSCKF framework. By harnessing the information gathered from both sensors, our objective is to precisely ascertain the robot's state and pinpoint its location.**

I. Phase 1: Classical Approach

A. Initialize Gravity and Bias

The 6-DOF IMU sensor, which is utilized for measuring rotation (via the gyroscope) and acceleration (via the accelerometer), necessitates a calibration process to compensate for any inherent biases. This process involves determining the bias by averaging the sensor readings while at rest and then adjusting future readings by subtracting this bias. Ideally, the gyroscope should show [0, 0, 0] when stationary, but minor variations can occur. Likewise, the accelerometer should read [0, 0, -g] relative to the world frame, but it too can display fluctuations owing to noise and bias. Calibration takes place before flight initiation to neutralize any biases in the readings from both the gyroscope and accelerometer. The function "initialize gravity and bias" sets up the IMU's gravity vector and bias as well as the robot's initial orientation using the initial set of IMU readings. It computes the gyro bias and assesses the gravity vector within the IMU's frame by taking the mean of the angular velocity and linear acceleration from the IMU data. The initial orientation is aligned with the inertial frame. This critical step ensures that the Visual-Inertial Odometry system begins operation on a sound and accurate basis.

B. Batch IMU Processing

The function for batch processing IMU data operates by continuously reading IMU messages until a fresh batch of images from the stereo camera is obtained. The vector that is utilized to estimate subsequent states comprises elements associated with both the IMU and the camera. This includes the quaternion for orientation, bias values for both the gyroscope and the accelerometer, as well as position and velocities.

$$x_I = \left({}^I\mathbf{q}_G^T,\, {}^B\mathbf{b}_g^T,\, {}^G\mathbf{v}_I^T,\, {}^G\mathbf{b}_a^T,\, {}^I\mathbf{p}_C^T,\, {}^I\mathbf{q}_C^T\right)^T$$

The "batch imu processing" operation is an integral component of the Visual-Inertial Odometry system. It advances the IMU's state by analyzing the IMU data within a set timeframe. This function cycles through the buffer's IMU data, discarding messages that have already been addressed and ceasing once it reaches the predetermined time limit. For every IMU message that hasn't been processed, the function employs a process model to revise the IMU's state, taking into account both the angular velocity and linear acceleration data. It updates the timestamp and ID of the IMU state and then purges the processed messages from the buffer. This process is essential for ensuring precise state advancement and maintaining synchronization between the IMU and the Visual Odometry system components, thereby playing a key role in dependable sensor fusion and localization.

C. Process Model

The "process model" function advances the system's state and uncertainty using a fourth-order Runge-Kutta integration technique. It begins by gathering pertinent details from the existing system state, such as the IMU state encompassing orientation, velocity, position, and biases for the gyroscope and accelerometer. It then determines the time interval using the given time and the IMU state's timestamp.

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

Following that, the function calculates the discrete system dynamics matrix (F) and the noise covariance matrix (G), which delineate the system's behavior and noise properties. The system dynamics matrix is estimated by employing a third-order matrix exponential approach, tailored for brief time intervals (dt). To achieve this, intermediary matrices Fdt, the square of Fdt, and the cube of Fdt are computed.

Subsequently, the function employs the fourth-order Runge-Kutta method to project the new system state by invoking the function for state prediction. This typically revises the system state based on the gyroscopic and accelerometric data, alongside the extant state conjectures. The transition matrix Phi is also adjusted to incorporate the null space, representing the set of states that remain unobservable through the sensor data.

$$F = \begin{pmatrix} -\left[\boldsymbol{\omega}\right]_\times & -I_3 & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ -C(\mathbf{q}_G^I)\left[\mathbf{a}\right]_\times & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & -C(\mathbf{q}_G^I) & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & I_3 & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \end{pmatrix}$$

$$G = \begin{pmatrix} -I_3 & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & I_3 & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & -C(\mathbf{q}_G^I)^T & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & I_3 \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \end{pmatrix}$$

The state covariance matrix (Q) is refreshed using Phi, G, and the continuous noise covariance matrix, illustrating the system model's uncertainties or inaccuracies. The covariance linking the IMU and camera states (if existing) is correspondingly revised. To ensure the state covariance matrix is symmetric, it is averaged with its transpose.

The IMU state is updated with the latest orientation, position, and velocity values, which constitute the null space for the upcoming cycle of the state estimation procedure.

D. *Predict New State*

The "predict new state" function carries out a forecasting phase within an Extended Kalman Filter framework. This phase is centered on integrating the data from the IMU sensors, specifically the gyroscopes and accelerometers, through a specific time increment to project the system's forthcoming state. This future state encompasses the IMU's orientation, speed, and location. The integration employs a quartic Runge-Kutta methodology, which is adaptable in terms of time steps. It also involves computing intermediary factors (k1, k2, k3, k4) grounded on the gyroscope data and the IMU's present state, and then it utilizes these factors to refresh the IMU's orientation, velocity, and position.

E. *State Augmentation*

The "state augmentation" function carries out the augmentation of the state by incorporating a fresh camera state into the state server. It also refines the covariance matrix and verifies its symmetry when new images are introduced. This involves computing the rotational and translational relationship between the IMU and the camera, updating the state of the camera accordingly, and adjusting the covariance matrix to reflect this additional state. This process is essential to ensure coherent alignment of the IMU and camera states within the Inertial Navigation System framework.

$$J = \begin{pmatrix} J_I & 0_{6\times6N} \end{pmatrix}$$

$$J_I = \begin{pmatrix} C(\mathbf{q}_G^I) & 0_{3\times9} & 0_{3\times3} & I_3 & 0_{3\times3} \\ -C(\mathbf{q}_G^I)\left[\mathbf{p}_I^C\right]_\times & 0_{3\times9} & I_3 & 0_{3\times3} & I_3 \end{pmatrix}$$

$$\mathbf{z}_i^j = \begin{pmatrix} u_{i,1}^j \\ v_{i,1}^j \\ u_{i,2}^j \\ v_{i,2}^j \end{pmatrix} = \begin{pmatrix} \frac{1}{^{C_{i,1}}Z_j} & \mathbf{0}_{2\times2} \\ \mathbf{0}_{2\times2} & \frac{1}{^{C_{i,2}}Z_j} \end{pmatrix} \begin{pmatrix} ^{C_{i,1}}X_j \\ ^{C_{i,1}}Y_j \\ ^{C_{i,2}}X_j \\ ^{C_{i,2}}Y_j \end{pmatrix}$$

Fig. 1: Measurements of stereo

$$^{C_{i,1}}\mathbf{p}_j = \begin{pmatrix} ^{C_{i,1}}X_j \\ ^{C_{i,1}}Y_j \\ ^{C_{i,1}}Z_j \end{pmatrix} = C\left(^{C_{i,1}}_{G}\mathbf{q}\right)\left(^{G}\mathbf{p}_j - {^{G}\mathbf{p}_{C_{i,1}}}\right)$$

$$^{C_{i,2}}\mathbf{p}_j = \begin{pmatrix} ^{C_{i,2}}X_j \\ ^{C_{i,2}}Y_j \\ ^{C_{i,2}}Z_j \end{pmatrix} = C\left(^{C_{i,2}}_{G}\mathbf{q}\right)\left(^{G}\mathbf{p}_j - {^{G}\mathbf{p}_{C_{i,2}}}\right)$$
$$= C\left(^{C_{i,2}}_{C_{i,1}}\mathbf{q}\right)\left(^{C_{i,1}}\mathbf{p}_j - {^{C_{i,1}}\mathbf{p}_{C_{i,2}}}\right)$$

Fig. 2: Position of the features in left and right Camera Frame

F. *Adding Feature Observation*

The function "add feature observations" adds feature observations from a new image frame to the map server in a visual-inertial odometry system. It creates new map features for unseen features, updates observations for existing features, and calculates the tracking rate.

G. *Measurement Update*

The "measurement update" function performs the update based on measurements from visual features and inertial sensors.

To simplify the computational demands, the system initially applies QR decomposition to the Jacobian matrix H, provided there are more rows than columns in H. This action produces a condensed H matrix, referred to as H thin, and modifies the measurement vector to r thin accordingly.

The Kalman gain is calculated utilizing the compact H thin matrix, the covariance of the state P, and the covariance of the observation noise. This gain is pivotal in determining the extent to which the measurements are factored into the update phase.

The state's deviation, denoted as delta x, is determined by the product of the Kalman gain and the adapted measurement vector r thin. Delta x is then segmented into smaller vectors to update the states of the IMU and the camera independently.

The IMU's state receives an update through the execution of small-angle quaternion adjustments that refine the IMU state's orientation, gyro bias, velocity, accelerometer bias, and position. Moreover, adjustments are made to the extrinsic rotation and translation that link the IMU and camera. The camera states, which include orientation and position, are

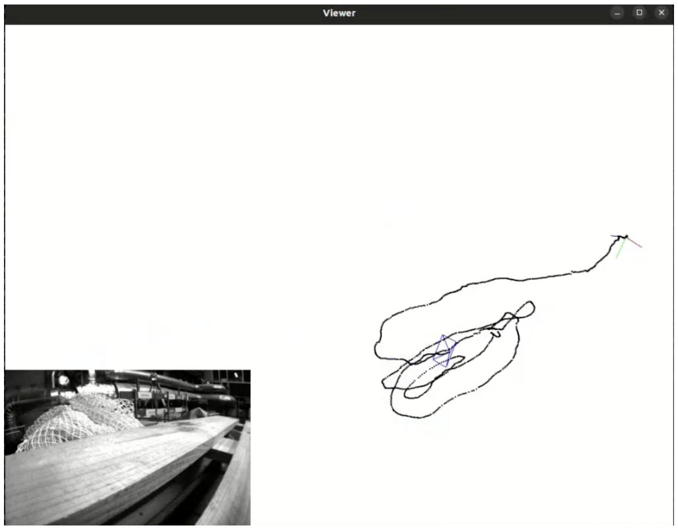

Fig. 3: Visualization

refined using small-angle quaternion maneuvers based on the delta x cam sub-vector.

The state covariance is updated with the help of the Kalman gain and the slimmed-down H thin matrix, resulting in the IKH matrix, which in turn is employed to bring the state covariance up to date. To maintain its symmetry, the newly updated state covariance is then rectified.

H. Results

The outcomes of our implementation are depicted in the following results. Additionally, we have visualized the errors relative to the ground truth using the MH 01 easy EuROC dataset.

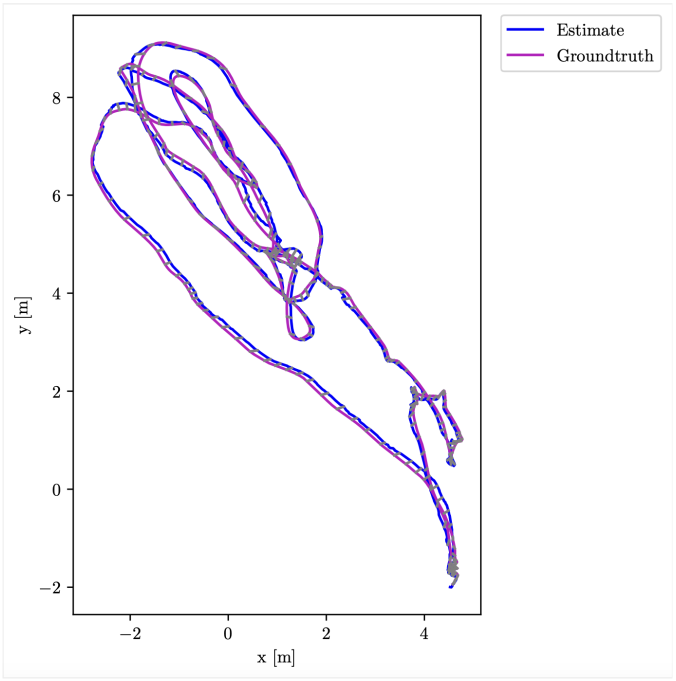

Fig. 4: Ground Truth vs Estimated Trajectory (Top View)

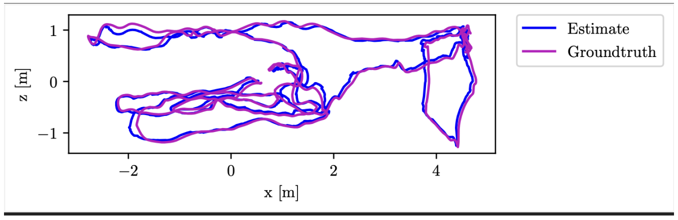

Fig. 5: Ground Truth vs Estimated Trajectory (Side View)

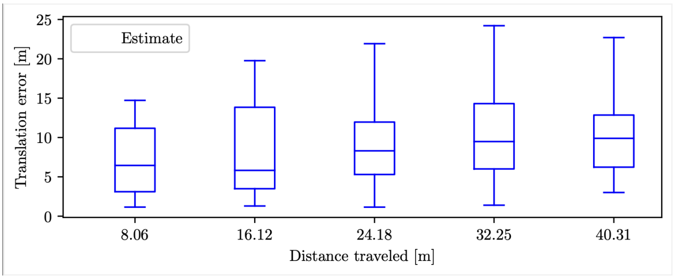

Fig. 6: Translational error

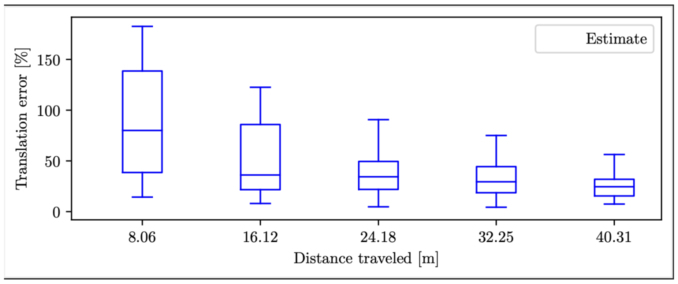

Fig. 7: Translational error percentage

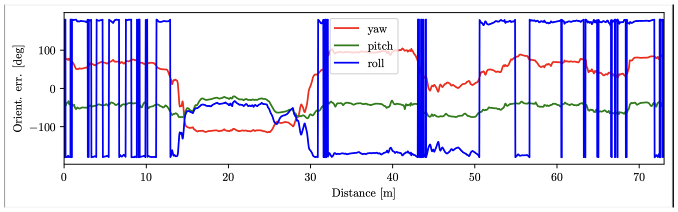

Fig. 8: Orientation error

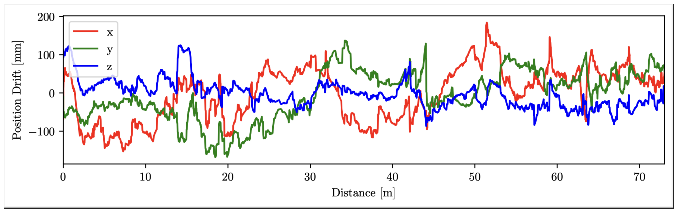

Fig. 9: Position drift