P4-Phase1
Deep adn Un-Deep Visual Inertial Odometry

Smit M Shah
Email: smshah1@wpi.edu
Worcester Polytechnic Institute

Rigved Sanku
Email: rsanku@wpi.edu
Worcester Polytechnic Institute

I. Introduction

The main goal of this paper is to implement and recreate the Stereo Multi-State Contraint Kalman Filter(MSCKF). The following functions of the MSCKF python implementation were changed initialize_gravity_and_bias, batch_imu_processing, process_model, predict_new_state, state_augmentation, add_feature_observations, measurement_update and predict_new_state.

II. Initialize Gravity and Bias

A. Gyroscope Initialization

The gyroscope measures the rate of rotation around the IMU's axes. However, even when the IMU is stationary, the gyroscope might show some readings due to bias. We average the angular velocity of initial 200 readings from the IMU while it is assumed to be stationary. This average gives you an estimate of the gyroscope bias.

B. Accelerometer Bias and Gravity Initialization

The accelerometer measures the acceleration in all three axes of the IMU. When stationary, the only acceleration an IMU should theoretically measure is the acceleration due to gravity pointing downwards. By averaging the accelerometer readings(initial 200) during a period when the IMU is static, you can estimate the direction and magnitude of gravity. This average should approximate the gravitational acceleration vector, typically around 9.81 m/s² pointing toward the earth.

C. Calculating Initial Orientation

The IMU needs to know its orientation relative to the world frame. Without this, you can't accurately translate the IMU's readings into movements in your SLAM map. With the estimated gravity vector from the accelerometer and the known gravity vector in the world frame, you can compute the initial orientation. This is done by finding the rotation that aligns the +z vector (from the IMU frame) to the measured gravity direction.

III. Batch IMU Processing

This function processes the buffered IMU data up to a given time bound. The function is designed to propagate the IMU's state based on incoming IMU data up to a specified time. It updates the system's understanding of the IMU's current orientation, position, and velocity based on the angular velocities and linear accelerations measured by the IMU.

IV. Process Model

Method designed to update the IMU's state based on the current IMU readings (gyroscope and accelerometer), correct these readings for biases, and propagate these updates through the system using a mathematical model of IMU dynamics. The mathematical model in continuous time is described equation 1.

$$\begin{aligned} {}^I_G\dot{\bar{q}}(t) &= \frac{1}{2}\Omega(\omega(t))^I_G\bar{q}(t), \\ \dot{b}_g(t) &= n_{wg}(t), \\ {}^G\dot{v}_I(t) &= {}^G a(t), \\ \dot{b}_a(t) &= n_{wa}(t), \\ {}^G\dot{p}^I(t) &= {}^G v_I(t) \end{aligned} \tag{1}$$

Where ${}^I_G\bar{q}(t)$ is the unit quaternion for rotation from Global frame G to IMU frame I.

$$\Omega(\omega) = \begin{bmatrix} \omega & \hat{\omega} \\ \omega^T & 0 \end{bmatrix} \tag{2}$$

Here $\hat{\omega}$ is a skew-symmetric matrix of the ω vector.
The gyroscope measurements w_m are written as

$$\omega_m = \omega + b_g + n_g \tag{3}$$

The filter propagation equations are derived by discretization of the continuous-time IMU system model. The time evolution of IMU state dynamics is given by

$$\begin{aligned} {}^I_G\dot{\hat{q}}(t) &= \frac{1}{2}\Omega(\hat{\omega}^I_G)\hat{q}, \\ \dot{\hat{b}}_g &= 0_{3\times 1}, \\ {}^G\dot{\hat{v}}_I = C^{\hat{q}}_T\hat{a} - 2\lfloor\omega_{G\times}\rfloor^G\hat{v}_I + \lfloor\omega_{G\times}\rfloor^{2G}\hat{p}_I &+^G g, \\ \dot{\hat{b}}_a &= 0_{3\times 1}, \\ {}^G\dot{\hat{p}}_I &=^G \hat{v}_I \end{aligned} \tag{4}$$

And the error dynamics of IMU error state are given by

$$\dot{\tilde{X}}_I = F\tilde{X} + Gn_I \tag{5}$$

Where F and G are

$$F = \begin{bmatrix} \lfloor \hat{\omega} \times \rfloor & -I_3 & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ -C({}^I_G\hat{q})^T \lfloor \hat{a}_\times \rfloor & 0_{3\times3} & 0_{3\times3} & -C({}^I_G\hat{q})^T & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & I_3 & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \end{bmatrix} \quad (6)$$

$$G = \begin{bmatrix} -I_3 & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & I_3 & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & -C({}^I_G\hat{q})^T & 0_{3\times3} \\ 0_{3\times3} & I_3 & 0_{3\times3} & I_3 \\ 0_{3\times3} & I_3 & 0_{3\times3} & 0_{3\times3} \end{bmatrix} \quad (7)$$

V. Predict State

This function manages the transition and covariance matrices. It begins by calculating the norm of the gyroscope value and retrieving the current orientation, velocity, and position from the IMU. Next, it propagates the state using the 4th order Runge-Kutta method. The updated values for orientation, velocity, and position for the subsequent state are then computed, leading to an updated state for the IMU.

VI. State Augmentation

The state_augmentation function executes the state augmentation process by incorporating a new camera state into the state server, updating the covariance matrix, and maintaining symmetry when new images are added. It computes the rotation and translation from the IMU to the camera, refreshes the camera state, and adjusts the covariance matrix according to the new state. This step is essential for preserving alignment between the IMU and camera states in the INS implementation. The augmented J and J_1 matrix are

$$J = \begin{bmatrix} J_1 & O_{6\times6N} \end{bmatrix} \quad (8)$$

$$J1 = \begin{bmatrix} C({}^I_C q) & 0_{3\times9} & 0_{3\times3} & I_3 & 0_{3\times3} \\ \lfloor C({}^I_G q)^{T\,I} p_{C\times} \rfloor & 0_{3\times9} & I_3 & 0_{3\times3} & I_3 \end{bmatrix} \quad (9)$$

$$P_{k|k} = \begin{bmatrix} I_{21+6N} \\ J \end{bmatrix} P_{K|K} \begin{bmatrix} I_{21+6N} \\ J \end{bmatrix}^T \quad (10)$$

VII. Feature Observation

The add_feature_observations function incorporates feature observations from a new image frame into the map server of a visual-inertial odometry system. It generates new map features for previously unobserved features, updates observations for existing features, and determines the tracking rate.

VIII. Measurement Update

The measurement_update function performs the update based on measurements from visual features and inertial sensors. To reduce computational complexity, the Jacobian matrix H is initially decomposed using QR decomposition when the number of rows in H exceeds the number of columns. This results in a reduced-size matrix, H_{thin}, and a transformed measurement vector, r_{thin}. The Kalman gain, which is crucial for weighing measurements during the update step, is computed using H_{thin}, the state covariance P, and the observation noise covariance.

The state error, denoted as δx, is determined by multiplying the Kalman gain with r_{thin}. δx is then divided into subvectors for separate updates of the IMU and camera states. For the IMU, small-angle quaternion operations are applied to update the orientation, gyro bias, velocity, accelerometer bias, and position. Additionally, the extrinsic rotation and translation between the IMU and camera are updated.

For the camera states, updates to the orientation and position are executed using small-angle quaternion operations based on the sub-vector δx_{cam}.

Finally, the state covariance is updated using the Kalman gain and H_{thin} to compute the $I - KH$ matrix, which is then used to update the state covariance. Adjustments are made to ensure that the updated state covariance remains symmetric.

IX. Results

The absolute median trajectory error (ATE) is 0.08699091908516823 m and the root mean square translation error (RMSE) is 0.09346765838541407 m. 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. Refer Fig 1 to Fig 6

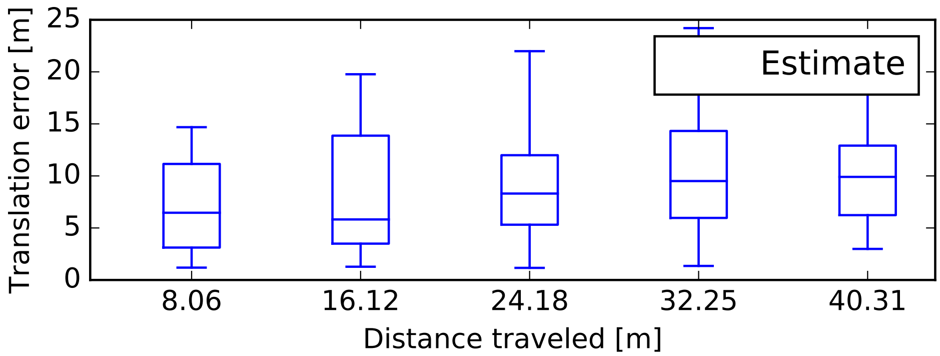

Fig. 1. Relative Translation Error

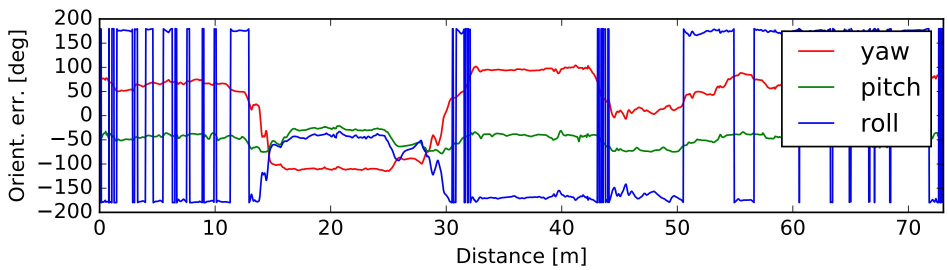

Fig. 2. Rotation Error

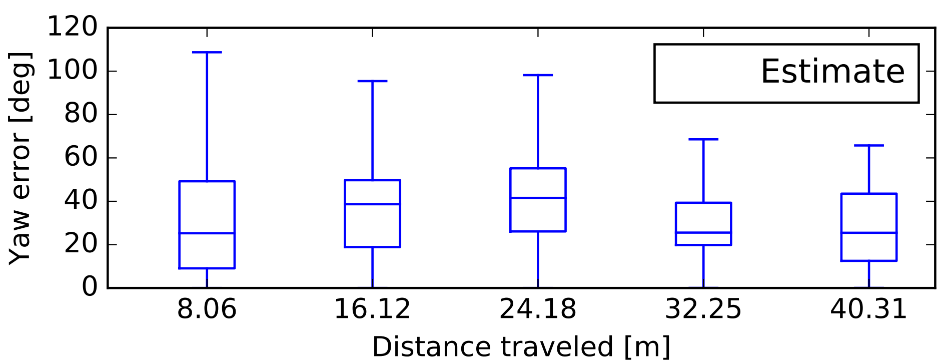

Fig. 3. Relative Yaw Error

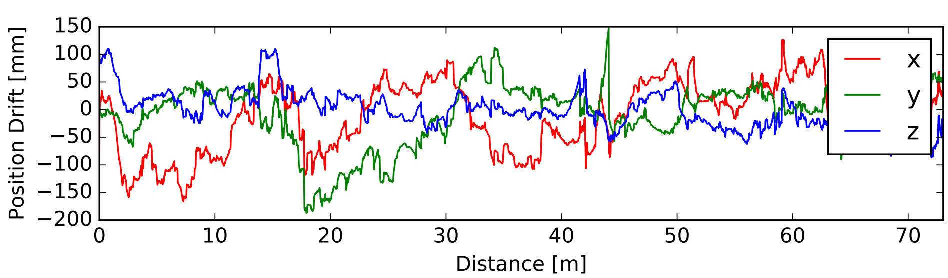

Fig. 4. Translation Error

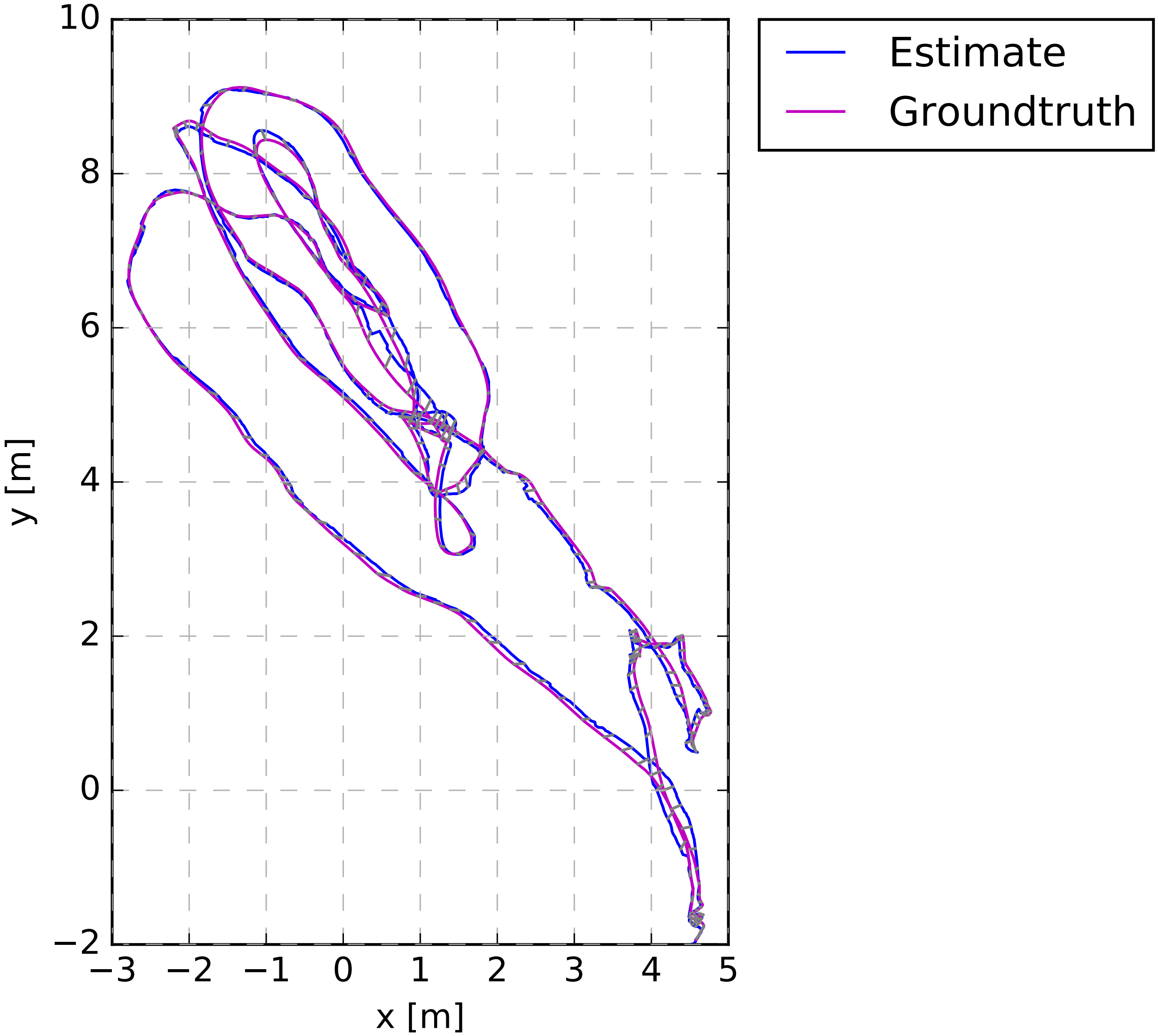

Fig. 5. Ground truth vs Estimated Trajectory (Side View)

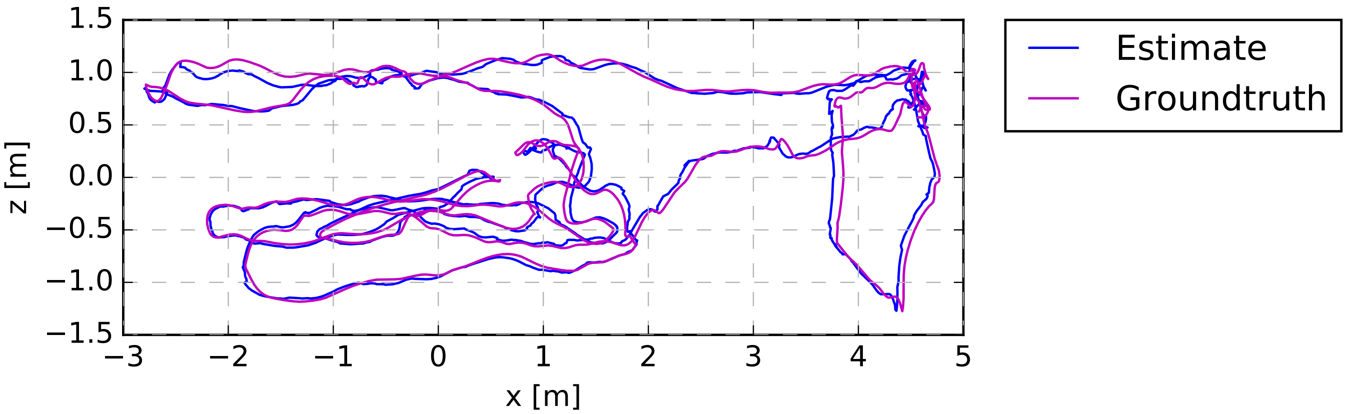

Fig. 6. Ground Truth vs Estimated Trajectory (Top View)