Project 4 -Deep and Un-Deep Visual Inertial Odometry

Harmeet Dhillon
Robotics Engineering
Worcester Polytechnic Institute
Worcester, MA - 01609
Email: hdhillon@wpi.edu

Rohan Walia
Robotics Engineering
Worcester Polytechnic Institute
Worcester, MA - 01609
Email: rwalia@wpi.edu

***Abstract*—This report details the development of a stereo visual-inertial odometry system utilizing the Multi-State Constraint Kalman Filter (MSCKF). The project expands upon a pre-existing codebase that served as the foundation. It discusses the mathematical concepts underpinning the stereo MSCKF algorithm and describes how these concepts were incorporated into the existing framework.**

I. Introduction

In this project, we complete seven functions in the given starter code that implements the work in [2]. We only test the code on the "Machine Hall 01" data from the EuRoC MAV Dataset [4]. In this episode, drone starts stationary, then it flies around before landing back to the same spot.

II. Initialize Gravity and Bias

As the initial part of the data corresponds to a short duration where the drone remains stationary, we take the first 200 readings, compute their average, and use this to correct for the gyroscope bias and the gravity vector of the accelerometer sensor.

The gyroscope bias is set to the average values of the gyroscope readings as

$$[b_{\text{avg}-gx},\ b_{\text{avg}-gy},\ b_{\text{avg}-gz}]$$

The accelerometer gravity vector's z-axis is set to the norm of the average accelerometer readings as

$$[0,\ 0,\ -\text{gravity}_{\text{avg-norm}}]$$

Finally, the initial orientation of the IMU is initialized to be the vector between the average accelerometer readings and the gravity vector.

III. Batch IMU Processing

In this function, we read the IMU readings (gyroscope and accelerometer) from the buffer and perform processing on them using the `process_model` function.

IV. Process Model

The estimated IMU state is modeled with continuous dynamics as:

$$ {}_G^I\dot{\hat{q}} = \frac{1}{2}\Omega(\hat{\omega})\,{}_G^I\hat{q} \tag{1}$$
$$ \dot{\hat{b}}_g = \mathbf{0}_{3\times1} \tag{2}$$
$$ {}^G\dot{\hat{v}} = C\left({}_G^I\hat{q}\right)^\top \hat{a} + {}^G g \tag{3}$$
$$ \dot{\hat{b}}_a = \mathbf{0}_{3\times1} \tag{4}$$
$$ {}^G\dot{\hat{p}}_I = {}^G\hat{v} \tag{5}$$
$$ {}_C^I\dot{\hat{q}} = \mathbf{0}_{3\times1} \tag{6}$$
$$ {}^I\dot{\hat{p}}_C = \mathbf{0}_{3\times1} \tag{7}$$

where $\hat{\omega}$ and $\hat{a}$ are the IMU measurements for angular velocity and linear acceleration respectively:

$$\hat{\omega} = \omega_m - \hat{b}_g \tag{8}$$
$$\hat{a} = a_m - \hat{b}_a \tag{9}$$

The matrix $\Omega(\hat{\omega})$ is defined as:

$$\Omega(\hat{\omega}) = \begin{bmatrix} -[\hat{\omega}\times] & \hat{\omega} \\ -\hat{\omega}^\top & 0 \end{bmatrix} \tag{10}$$

The continuous dynamics of the IMU state are linearized as:

$$\dot{\tilde{x}}_I = F\tilde{x}_I + Gn_I \tag{11}$$

where

$$n_I^\top = \begin{bmatrix} n_g^\top & n_{wg}^\top & n_a^\top & n_{wa}^\top \end{bmatrix}$$

The vectors n_g and n_a denote the Gaussian noise associated with the gyroscope and accelerometer measurements, respectively, while n_{wg} and n_{wa} represent the random walk components of the gyroscope and accelerometer biases.

The discrete transition matrix F and the noise covariance matrix G are defined as:

$$F = \begin{bmatrix} -\lfloor\hat{\omega}\times\rfloor & -I_3 & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ -C\left({}^I_G\hat{q}\right)^\top \lfloor\hat{a}\times\rfloor & 0_{3\times3} & 0_{3\times3} & -C\left({}^I_G\hat{q}\right)^\top & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & I_3 & 0_{3\times3} & 0_{3\times3} \end{bmatrix} \tag{12}$$

$$G = \begin{bmatrix} -I_3 & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & I_3 & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & -C\left({}^I_G q\right)^\top & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & I_3 \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \end{bmatrix} \tag{13}$$

The discrete-time state transition matrix is given by:

$$\Phi_k = \Phi(t_{k+1}, t_k) = \exp\left(\int_{t_k}^{t_{k+1}} F(\tau)d\tau\right) \tag{14}$$

Here, the matrix exponential is approximated by the first three terms of the power series expansion:

$$\exp\left(\int_{t_k}^{t_{k+1}} F(\tau)d\tau\right) \approx I + F(\tau)d\tau + \frac{1}{2}(F(\tau)d\tau)^2 + \frac{1}{6}(F(\tau)d\tau)^3 \tag{15}$$

The discrete-time noise covariance matrix is given by:

$$Q_k = \int_{t_k}^{t_{k+1}} \Phi(t_{k+1},\tau)GQG^\top\Phi(t_{k+1},\tau)^\top d\tau \tag{16}$$

Here, Q is the continuous-time noise covariance matrix. The propagated covariance of the IMU state is:

$$P_{II_{k+1|k}} = \Phi_k P_{II_{k|k}} \Phi_k^\top + Q_k \tag{17}$$

Partitioning the covariance of the whole state as:

$$P_{k|k} = \begin{bmatrix} P_{II_{k|k}} & P_{IC_{k|k}} \\ P_{IC_{k|k}}^\top & P_{CC_{k|k}} \end{bmatrix}$$

the full uncertainty propagation can be written as:

$$P_{k+1|k} = \begin{bmatrix} P_{II_{k+1|k}} & \Phi_k P_{IC_{k|k}} \\ P_{IC_{k|k}}^\top \Phi_k^\top & P_{CC_{k|k}} \end{bmatrix}$$

All these computations occur in the `process_model` function.

V. Predict New State

Since the matrices are approximated in discrete time, we propagate the state using the 4th order Runge-Kutta method for numerical integration in the `predict_new_states` function.

Applying the 4th order Runge-Kutta method, we compute the coefficients:

$$k_1 = f(t_n, y_n) \tag{18}$$

$$k_2 = f\left(t_n + \frac{\Delta t}{2},\ y_n + k_1\frac{\Delta t}{2}\right) \tag{19}$$

$$k_3 = f\left(t_n + \frac{\Delta t}{2},\ y_n + k_2\frac{\Delta t}{2}\right) \tag{20}$$

$$k_4 = f(t_n + \Delta t,\ y_n + k_3\Delta t) \tag{21}$$

The state is then propagated using the equation:

$$y_{n+1} = y_n + \frac{\Delta t}{6}(k_1 + 2k_2 + 2k_3 + k_4) \tag{22}$$

VI. State Augmentation

In the `state_augmentation` function, the camera position ${}^G p_C$ and orientation ${}^C_G q$ are updated using the last IMU state. The state covariance matrix P is also augmented to include these new states.

A. Camera Pose Computation

The pose of the camera is computed based on the latest IMU state using the following equations:

$${}^G p_C = {}^G p_I + C\left({}^C_G q\right)^\top {}^I p_C \tag{23}$$

$${}^C_G q = {}^C_I q \otimes {}^I_G q \tag{24}$$

where ${}^G p_I$ is the position of the IMU, ${}^C_I q$ and ${}^I_G q$ are the quaternions representing the rotations from the camera to the IMU and from the IMU to the global frame, respectively, and ${}^I p_C$ is the position of the camera relative to the IMU.

B. Jacobian Matrix Computation

The Jacobian J of the transformation is computed to facilitate the update of the covariance matrix P. The Jacobian matrix is given by:

$$J = \begin{bmatrix} J_1 & 0_{6\times6N} \end{bmatrix} \tag{25}$$

where

$$J_1 = \begin{bmatrix} C\left({}^I_C q\right) & 0_{3\times9} & 0_{3\times3} & I_3 & 0_{3\times3} \\ -C\left({}^I_G q\right)^\top \lfloor {}^I p_C \times\rfloor & 0_{3\times9} & I_3 & 0_{3\times3} & I_3 \end{bmatrix} \tag{26}$$

Here, $0_{3\times9}$ and $0_{3\times3}$ are zero matrices of appropriate sizes, I_3 is the 3×3 identity matrix, and $\times$ denotes the skew-symmetric cross product matrix of ${}^I p_C$.

C. Covariance Matrix Augmentation

Using the Jacobian matrix J, the state covariance matrix P is augmented to reflect the increased uncertainty due to the addition of the camera state:

$$P_{k|k} = \begin{bmatrix} I_{21+6N} & J \end{bmatrix} P_{k|k} \begin{bmatrix} I_{21+6N} \\ J^\top \end{bmatrix} \tag{27}$$

where I_{21+6N} is the identity matrix sized to match the augmented state dimensions, incorporating all previous states and the newly added camera state.

This organized approach guarantees that the visual information, encapsulated by the camera pose, is properly incorporated into the overall state estimate of the system. Such integration aids in maintaining a precise and robust estimation of the system's state within a dynamic environment.

VII. Add Feature Observation

Newly identified features are incorporated into the feature map along with their observed measurements and corresponding uncertainties. For each feature detected in the image frame at time t, its location in pixel coordinates is included in the state vector, accompanied by a unique identifier. The state update is expressed as:

$$Z_i^j = \begin{bmatrix} u_{i,1}^j \\ v_{i,1}^j \\ u_{i,2}^j \\ v_{i,2}^j \end{bmatrix} \tag{28}$$

where $u_{i,1}^j$ and $v_{i,1}^j$ are the coordinates in the left camera, and $u_{i,2}^j$ and $v_{i,2}^j$ in the right camera for the i-th observation of feature j.

VIII. Measurement Update

The measurement update function processes the measurement matrix H and the residual vector r to compute the Kalman gain K, and subsequently updates the IMU state X_{IMU}, the camera state, and the state covariance matrix P.

A. Constructing the Measurement Matrix H

The measurement matrix H consists of block rows $H(j)$, where $j = 1, \dots, L$ corresponds to all detected features. When the number of measurements exceeds the number of state components, which is often the case, QR decomposition is applied to matrix H to manage its dimensionality and improve numerical stability:

$$H = \begin{bmatrix} Q_1 & Q_2 \end{bmatrix} \begin{bmatrix} R \\ 0 \end{bmatrix} \tag{29}$$

Here, Q_1 and Q_2 are orthogonal matrices from the QR decomposition, and R is an upper triangular matrix representing the reduced form of H.

B. Computing the Residual r_n

The residual r_n is calculated by projecting the original residual r onto the column space of Q_1, effectively reducing the influence of measurement noise:

$$r_n = Q_1^\top r = R\tilde{X} + n_n \tag{30}$$

where $\tilde{X}$ denotes the state error and n_n represents the noise in the measurement process.

C. Covariance Matrix of the Noise Vector n_n

The covariance matrix R_n of the noise vector n_n is derived from the noise characteristics of the measurements, typically expressed as:

$$R_n = \sigma_{\text{im}}^2 I_q \times q \tag{31}$$

where σ_{im}^2 is the variance of the measurement noise and q is the dimensionality of the subspace spanned by Q_1.

D. Kalman Gain K

The Kalman gain K is typically computed using:

$$K = PH^\top (RR^\top + R_n)^{-1} \tag{32}$$

To circumvent numerical instabilities inherent in inverting large matrices directly, the computation is reformulated as solving the linear system $Ax = b$ where:

$$A = RPR^\top + R_n, \quad b = RP^\top \tag{33}$$

Here, A is symmetric, ensuring a more stable solution when computing K as:

$$K^\top = A^{-1} b \tag{34}$$

E. State Correction ΔX

Once K is obtained, it is used to compute the correction for the state:

$$\Delta X = K r_n \tag{35}$$

F. Updating the Covariance Matrix P

Finally, the state covariance matrix P is updated to reflect the reduced uncertainty after the measurement update:

$$P_{k+1|k+1} = (I - KH) P_{k|k} (I - KH)^\top + K R_n K^\top \tag{36}$$

This meticulous approach to the measurement update guarantees that visual information is efficiently incorporated into the state estimate, enhancing the accuracy and robustness of the visual-inertial odometry system.

IX. Results

We used the [rpg_trajectory_visualization], repository for comparing our estimated trajectory against the ground truth trajectory. This repository matches estimated trajectory waypoints to groundtruth waypoints by comparing the timestamps. We used SE_3 alignment for generating the following results:

RMSE	**Value**
Rotation	153.82
Scale	2.20
Translation	0.14

TABLE I: RMSE values for Rotation, Scale, and Translation.

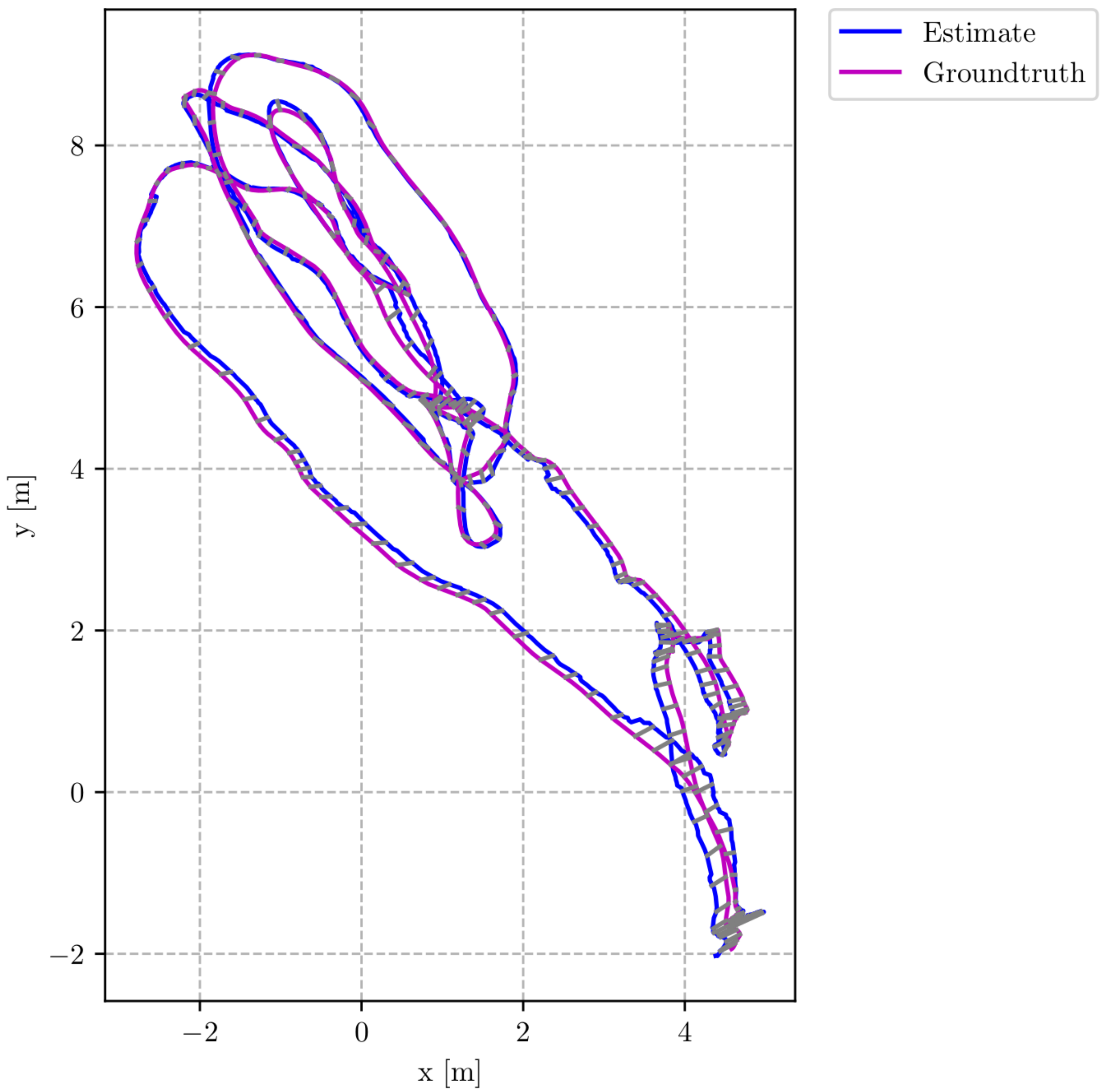

Fig. 1: Groundtruth vs Estimated Trajectory: Top View

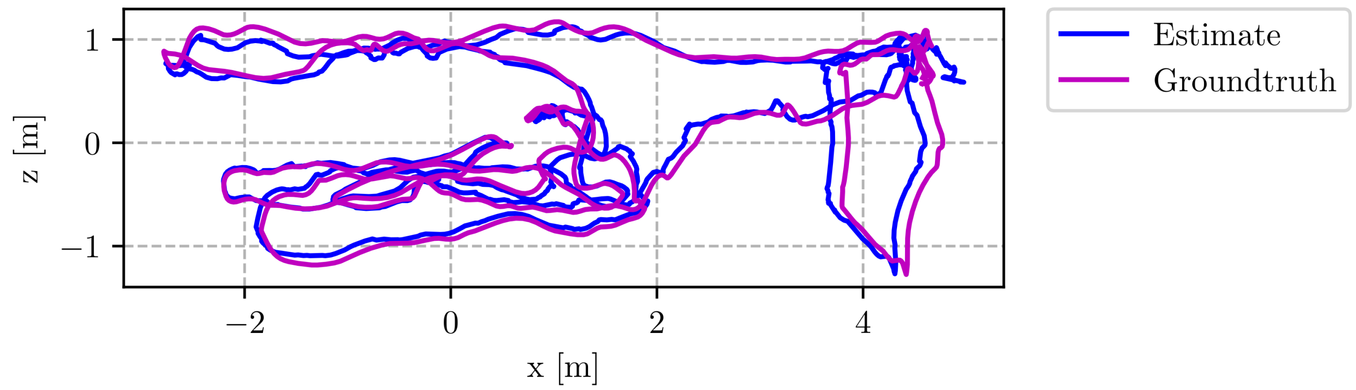

Fig. 2: Groundtruth vs Estimated Trajectory: Side View

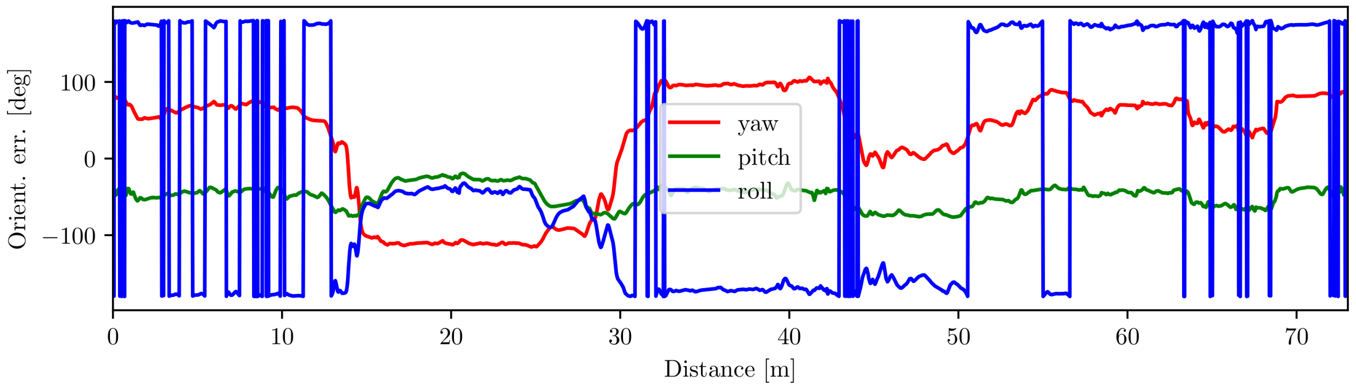

Fig. 3: Rotation Error (after SE_3 trajectory alignment)

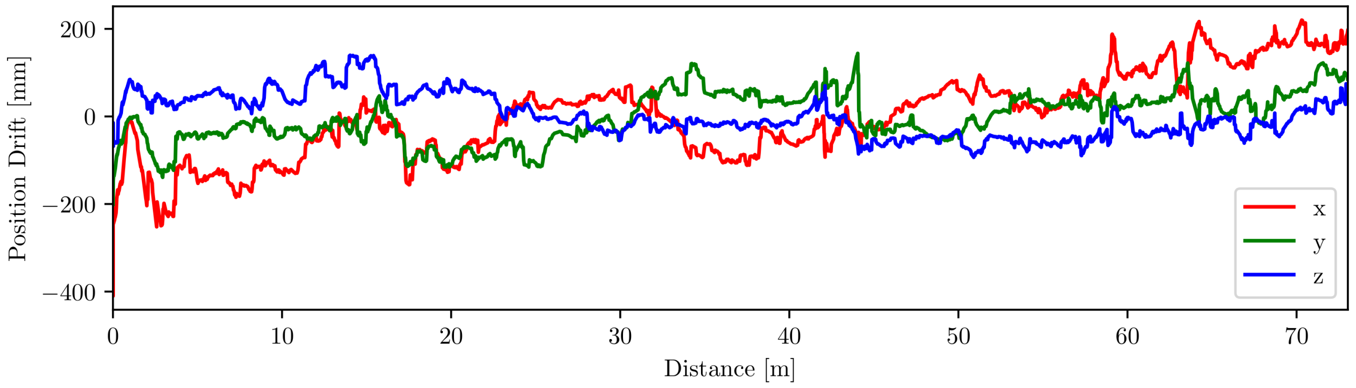

Fig. 4: Translation Error (after SE_3 trajectory alignment