Project 4: Visual Inertial Odometry

Ankit Mittal
Department of Robotics Engineering
Worcester Polytechnic Institute
Email: amittal@wpi.edu

Rutwik Kulkarni
Department of Robotics Engineering
Worcester Polytechnic Institute
Email: rkulkarni1@wpi.edu

***Abstract*—This project aims to showcase the integration of camera and IMU data to accurately determine pose and orientation in real-time. By harnessing the rapid and precise measurements from the IMU, complemented by the stable yet slower camera data, we achieve this objective. We utilize the Stereo MSCKF algorithm to implement this fusion effectively.**

I. Implementation Overview

The following sections give an overview about each function that was implemented as part of this project. We have taken reference of the follwing paper [1][2][3]

A. Initialize Gravity and Biases

This technique establishes the initial gravity and bias by analyzing the first 200 data points from the IMU. During this phase, the camera-IMU system remains stationary, allowing us to calculate gravity by averaging the acceleration measurements at specific timestamps. Setting the initial gravity vector is crucial as it serves as a reference for detecting changes in the IMU's orientation throughout the rest of the program. The orientation of a non-accelerating IMU is determined by comparing the current accelerometer readings with the initial gravity vector. Additionally, it's essential to account for the gyro bias, as failing to subtract this bias would result in a slight but constant error in the gyro readings. This could mislead the system into perceiving continuous rotation, leading to drift, even when the robot is stationary.

B. Batch IMU processing

In this module, the IMU messages stored in the IMU message buffer are processed sequentially. For each IMU message, the process model is executed, and the state information is updated accordingly. This procedure continues until a predefined time limit is reached, which is set according to the timestamp of the current image message. This ensures that all IMU messages received prior to the image message are processed. After processing, these IMU messages are then removed from the buffer.

C. Process Model

The process model is a crucial component of the Kalman Filter, responsible for updating the IMU error state dynamics, state covariance matrix, and setting the null-space projections for the IMU's orientation, position, and velocity. This method adheres to the equations found in section III.A of the MSCKF paper. Initially, it involves subtracting the biases from the raw gyro and accelerometer readings. The time difference (dt) between the current IMU message and the last processed timestamp for the IMU-state is calculated.

$$\omega = \omega_m - \hat{b}_g$$

$$\hat{a} = a_m - \hat{b}_a$$

Following this, the F and G matrices, as detailed in appendix A of the paper, are constructed. The F matrix, a 21x21 matrix, includes the screw matrix of the gyro omega vector, several identity matrices, and the rotated acceleration vector.

$$F = \begin{bmatrix} -[\hat{\omega}]_\times & -I_3 & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ -C(\hat{}_G^I q) & 0_{3\times3} & 0_{3\times3} & -C(\hat{}_G^I q) & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & I_3 & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \end{bmatrix}$$

The G matrix is constructed of size (21, 12) with several identity matricies and a rotation matrix of the orientation, as seen in the paper.

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

Once we have the matrix $\mathbf{F}$, we estimate Φ using the third-order matrix exponential approximated by a Taylor series expansion: $\mathbf{F}\Delta t$, $\mathbf{F}^2\Delta t^2$, and $\mathbf{F}^3\Delta t^3$. We then execute `PredictNextState` on the IMU readings to propagate the state using the fourth-order Runge-Kutta method. Following the methodology described in the paper and reference code, we adjust the transition matrix using the null-orientation and orientation to rotate the gravity vector and update the Φ matrix exponential. Once Φ is computed, We employ the following partitioning for the covariance:

$$P_{k|k} = \begin{bmatrix} P_{IIk|k} & P_{ICk|k} \\ P_{ICk|k}^T & P_{CCk|k} \end{bmatrix}$$

where $P_{IIk|k}$ is the 21×21 covariance matrix of the evolving IMU state, $P_{CCk|k}$ is the $6N \times 6N$ covariance matrix of the

camera pose estimation, and $P_{ICk|k}$ is the correlation between the errors in the IMU state and the camera pose estimates.

The covariance matrix of the propagated state can then be given by:

$$P_{k+1|k} = \begin{bmatrix} P_{IIk+1|k} & \Phi_k P_{ICk|k} \\ P_{ICk|k}^T \Phi_k^T & P_{CCk|k} \end{bmatrix} \quad (17)$$

where the propagated covariance of the IMU state is:

$$P_{IIk+1|k} = \Phi_k P_{IIk|k} \Phi_k^T + Q_k \quad (18)$$

The discrete-time noise covariance matrix Q_k can be computed as:

$$Q_k = \int_{t_k}^{t_{k+1}} \Phi(t_{k+1}, \tau) G Q G \Phi(t_{k+1}, \tau)^T \, d\tau \approx \Phi_k G Q G \Phi_k^T \Delta t$$

Finally, we enforce the symmetry of the covariance matrix with $P_{k+1|k} = \frac{P_{k+1|k} + P_{k+1|k}^T}{2}$, and refresh the null-space values for orientation, position, and velocity of the IMU state.

D. Predict New State

Given that the states represent errors relative to the previous state, it is necessary to integrate across all previous states in order to predict the new state. The continuous dynamics of the estimated IMU state, are linearized in Section II-C. To accommodate the discrete-time measurements of the IMU, state propagation is performed using a numerical integration function, specifically the 4th order Runge-Kutta method (RK4).

$$\begin{aligned} \frac{I}{2} {}^I_G\dot{\bar{q}}(t) &= \begin{bmatrix} -\boldsymbol{\omega}(t)_\times & \boldsymbol{\omega}(t) \\ -\boldsymbol{\omega}^T(t) & 0 \end{bmatrix} {}^I_G\bar{q} \\ &=: \frac{1}{2} \Omega(\boldsymbol{\omega}(t)) \, {}^I_G\bar{q} \end{aligned}$$

$$\begin{aligned} {}^G\dot{P}_I(t) &= {}^G v_I(t) \\ {}^G\dot{v}_I(t) &= \frac{I}{G} \boldsymbol{R}^T \boldsymbol{a}(t) \\ \dot{\boldsymbol{b}}_g(t) &= \boldsymbol{n}_{wg} \\ \dot{\boldsymbol{b}}_a(t) &= \boldsymbol{n}_{wa} \end{aligned}$$

The orientation, pose, and velocity are computed alongside $\boldsymbol{\omega}_\times$. Utilizing the above equations, we calculate $\delta\bar{q} * \delta t$ which defines the new state orientation. It is also essential to compute the fourth-order Runge-Kutta (RK4) for velocity and pose.

$${}^I_G\bar{q}_I = \delta\bar{q} \otimes {}^{\hat{I}}_G\bar{q}$$

$$\delta\bar{q} = \begin{bmatrix} \mathbf{k} \sin\left(\frac{1}{2}\boldsymbol{\theta}\right) \\ \cos\left(\frac{1}{2}\boldsymbol{\theta}\right) \end{bmatrix} \approx \begin{bmatrix} \frac{1}{2}\boldsymbol{\theta} \\ 1 \end{bmatrix}$$

E. State Augmentation

When new images are received, the state should be augmented with the new camera state. The pose of the new camera state can be computed from the latest IMU state:

$$\begin{aligned} {}^C_G\hat{q} &= {}^C_I\hat{q} \otimes {}^I_G\hat{q} \\ {}^C_G\hat{p} &= {}^G\hat{p}_c + C({}^I_G\hat{q})^T \, {}^I\hat{p}_c \end{aligned}$$

where ${}^C_G\hat{q}$ is the quaternion of the new camera with respect to the world frame, ${}^C_I\hat{q}$ is the quaternion of the new camera with respect to the IMU frame, ${}^I_G\hat{q}$ is the quaternion of the IMU with respect to the world frame, ${}^C_G\hat{p}$ is the position of the new camera with respect to the world frame, $\hat{\mathbf{p}}_G$ is the position of the IMU with respect to the world frame, and ${}^I\hat{p}_c$ is the position of the new camera with respect to the IMU frame.

Therefore, as shown in the reference [2], the augmented covariance matrix is:

$$P_{k|k} = \begin{bmatrix} I_{21+6N} \\ J \end{bmatrix} P_{k|k} \begin{bmatrix} I_{21+6N} \\ J \end{bmatrix}^T$$

$$J = \begin{bmatrix} J_I & 0_{6 \times 6N} \end{bmatrix}$$

$$J_I = \begin{bmatrix} C({}^I_G\hat{q}) & 0_{3\times9} & 0_{3\times3} & I_3 & 0_{3\times3} \\ -C({}^I_G\hat{q})^T [{}^I\hat{p}_c]_\times & 0_{3\times9} & I_3 & 0_{3\times3} & I_3 \end{bmatrix}$$

where N is the number of the camera states.

Eventually, the covariance is fixed to be symmetric:

$$P_{k+1|k} = \frac{P_{Ik+1|k} + P_{Ik+1|k}^T}{2}$$

F. Add Feature Observations

Upon receiving a new feature message, we add the features to the map server. We iterate over all the features (stereo points from the images) contained in the message. For each feature, we verify its presence in the map server. If found, we update the observations for the corresponding IMU message with a matching ID. Then, the tracking rate is updated based on the equation:

$$\text{trackingRate} = \frac{\text{trackedFeatures}}{\text{stateFeatures} + 0.00001}$$

G. Measurement Update

The measurement update is a pivotal element of the Kalman filter. It is responsible for computing the Kalman gain and updating the state estimates, such as position, velocity, and orientation. Initially, the Jacobian matrices H and r are decomposed using QR decomposition (employing the Numpy function in "reduced" mode to enhance sparsity), thereby minimizing computational load. The Kalman gain is computed using the formula $K = \text{linearSolve}(S, HP)^T$, with $S = HPH^T + \text{obs-Cov} \cdot I$. This approach aligns with the Kalman gain formulation presented in equation 29 of the seminal MSCKF paper.

$$\mathbf{K} = \mathbf{P}\mathbf{T}^T\mathbf{H}^T \left(\mathbf{H}\mathbf{P}\mathbf{T}^T\mathbf{H}^T + \mathbf{R}_n\right)^{-1}$$

The correction to the state is computed using equation, expressed as $\Delta X = K \cdot r$. The ΔIMU comprises the first 21 elements of the ΔX. The quaternion representing the small-angle change for the IMU ($IMU - DQ$) is derived from the first three state components of ΔIMU using the small-angle quaternion formula. The orientation is updated by the quaternion multiplication of the previous orientation with the new $IMU - DQ$. The remaining IMU state components, including GyroBias, Velocity, AccelerometerBias, and Position, are updated by adding the corresponding ΔIMU components. Subsequently, the IMU rotation and translation components relating to the IMU-to-Camera frames are updated using the small-angle quaternion derived from the ΔIMU components.

After updating the IMU state components, we proceed to update the camera states. This is done by iterating through each camera state and applying the specific rotation and translation components of ΔX that correlate to each camera state. The updates are computed using the small-angle quaternion form and quaternion multiplication. Finally, to ensure the symmetry of the state covariance matrix, we enforce it by setting

$$P_{k+1|k} = \frac{P_{Ik+1|k} + P_{Ik+1|k}^T}{2}$$

II. Trajectory Error Evaluation

The estimated and ground truth trajectories are plotted in both x-y and x-z as given below. The close alignment of these trajectories signifies a highly accurate estimation.

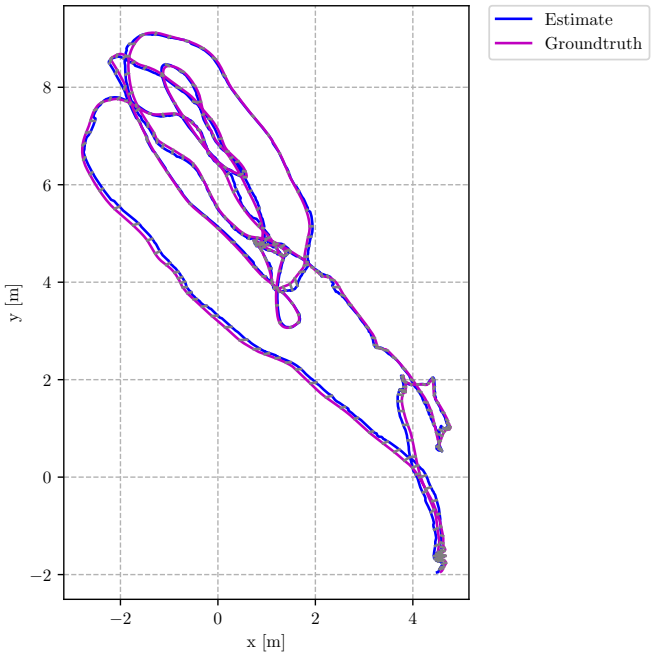

Fig. 1: Estimated Trajectory Pose v/s Ground Truth Trajectory (Top View)

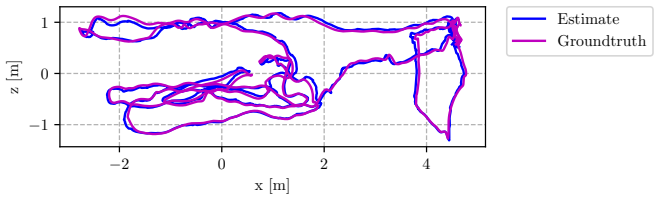

Fig. 2: Estimated Trajectory Pose v/s Ground Truth Trajectory (Side View)

We compute the Absolute Trajectory Error (ATE) statistics using [4] and they are presented in TABLE 1. The analysis reveals that our estimated trajectory exhibits low ATE RMSE (0.075 meters) and minimal mean and median errors. Additionally, the small standard deviation indicates consistent tracking performance throughout the experiment.

TABLE I: Absolute Trajectory Error Statistics

Parameter	Mean	Median	RMSE	Std Dev
Rotation	1.764	1.782	1.863	0.597
Scale	0.748	0.502	1.049	0.736
Translation	0.068	0.069	0.075	0.030

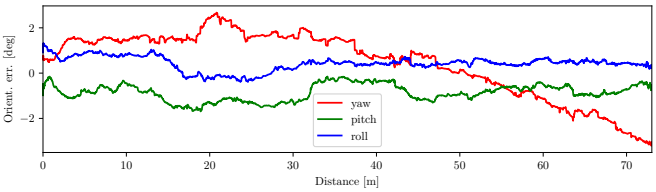

Fig. 3: Rotation Error

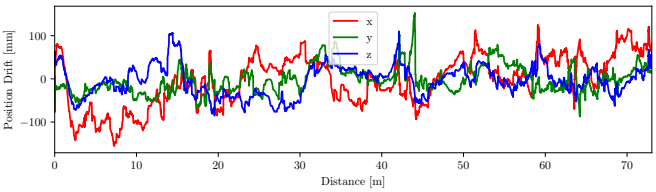

Fig. 4: Translation Error

III. Acknowledgment

The author would like to thank Prof. Nitin Sanket, Teaching Assistant, and Grader of this course RBE549- Computer Vision.

References

[1] A. I. Mourikis and S. I. Roumeliotis, "A multi-state constraint kalman filter for vision-aided inertial navigation," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*, 2007, pp. 3565–3572.

[2] K. Sun, K. Mohta, B. Pfrommer, M. Watterson, S. Liu, Y. Mulgaonkar, C. J. Taylor, and V. R. Kumar, "Robust stereo visual inertial odometry for fast autonomous flight," *IEEE Robotics and Automation Letters*, vol. 3, pp. 965–972, 2017. [Online]. Available: https://api.semanticscholar.org/CorpusID:3725704

[3] ——, "Vio github repository," *IEEE Robotics and Automation Letters*, 2017. [Online]. Available: https://github.com/KumarRobotics/msckf_vio
[4] Z. Zhang and D. Scaramuzza, "A tutorial on quantitative trajectory evaluation for visual(-inertial) odometry," in *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*, 2018.