Project 4 : Visual-Intertial Odomentry (VIO)

Abhijeet Sanjay Rathi
M.S. Robotics
Worcester Polytechnic Institute
Email: asrathi@wpi.edu

Anuj Jagetia
M.S. Robotics
Worcester Polytechnic Institute
Email: ajagetia@wpi.edu

I. Introduction

Using a Multi-state Constraint Kalman Filter (MSCKF), a filter-based stereo Vision-aided odometry is implemented in Phase 1. The method used in this project combines sensor data from an IMU and a stereo camera. We have applied eight MSCKF functions. Our goal is to precisely ascertain the robot's location and state using the information gathered from these two sensors. Additionally, we have assessed the S-MSCKF output for the EuRoC dataset in relation to ground truth.

II. Intializing Gravity and Bias

When the robot is positioned statically, the first 200 messages from the IMU are used to initialize the gyroscope bias and gravity. The average of these 200 gyroscope readings serves as the initial value for the gyroscope bias bg. The initial value of the gravity g is [0, 0, gnorm], where gnorm represents the average of the first 200 accelerometer readings. The quaternion from -g to gnorm is used to initialize the orientation.

III. Batch IMU Processing

Upon receiving a new feature, we want to process all of the IMU messages that were received before the time stamp of the new feature. We use the process model to update our state estimation for each IMU message received before the feature time stamp and stored in the IMU message buffer.

IV. Process Model

Using a fourth order Runge-Kutta integration technique, the function "process model" propagates the system state and covariance. It begins by gathering pertinent data from the existing state of the system, including the orientation, velocity, position, and accelerometer and gyroscope biases from the IMU (Inertial Measurement Unit) state. The time step is then determined using the given time and IMU state timestamp.

$$\begin{aligned} {}^{I}_{G}\dot{\overline{q}} &= \frac{1}{2}\Omega(\omega(t))\,{}^{I}_{G}\overline{q}(t), \\ \dot{b_g}(t) &= n_{wg}(t), \\ {}^{G}\dot{v}_I(t) &= {}^{G}a(t), \\ \dot{b_a}(t) &= n_{wa}(t), \\ {}^{G}(\dot{p})_I(t) &= {}^{G}v_I(t), \end{aligned} \tag{1}$$

Where ${}^{I}_{G}\overline{q}(t)$ is the unit quaternion describing the rotation from global frame G to IMU frame I. $\omega(t) = [\omega_x, \omega_y, \omega_z]^T$ is the rotational velocity in IMU frame, and

$$\lfloor \omega_\times \rfloor = \begin{bmatrix} -\lfloor \omega_\times \rfloor & \omega \\ \omega^T & 0 \end{bmatrix} \tag{2}$$

$$\lfloor \omega_\times \rfloor = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & -\omega_x & 0 \end{bmatrix} \tag{3}$$

Gyroscope measurement equation:

$$\omega_m = \omega + b_g + n_g \tag{4}$$

Accelerometer measurement equation:

$$a_m = {}^{I}_{G}R({}^{G}a - {}^{G}g + 2\lfloor \omega_{G\times} \rfloor\, {}^{G}v_I + 2\lfloor \omega_{G\times} \rfloor\, {}^{2G}p_I) + b_a + n_a \tag{5}$$

where ${}^{I}_{G}R$ is the rotation matrix calculated from quaternion ${}^{I}_{G}\overline{q}$ Then apply expectation operator on equation 1 we obtain the quations for propagating IMU state estimates:

$$\begin{aligned} {}^{I}_{G}\dot{\hat{\overline{q}}} &= \frac{1}{2}\Omega(\hat{\omega}^{I}_{G})\hat{\overline{q}}, \\ \dot{\hat{b}}_g &= 0_{3\times 1}, \\ {}^{G}\dot{\hat{v}}_I &= C^T_{\hat{q}}\hat{a} - 2\lfloor \omega_{G\times} \rfloor\, {}^{G}\hat{v}_I + \lfloor \omega_{G\times} \rfloor\, {}^{2G}\hat{p}_I + {}^{G}g, \\ \dot{\hat{b}}_a &= 0_{3\times 1}, \\ {}^{G}\dot{\hat{p}}_I &= {}^{G}\hat{v}_I \end{aligned} \tag{6}$$

The code computes discrete transition (F) and noise (G) matrices, using a 3rd order matrix exponential method with a small time step. It predicts the system state using 4th order Runge-Kutta integration. The transition matrix Phi is adjusted for the null space. State covariance matrix (Q) is updated with Phi, G, and continuous noise covariance. Covariance between IMU and camera states is also updated. The IMU state is updated with current orientation, position, and velocity for the next iteration.

$$\dot{\tilde{X}}_{IMU} = F\tilde{X} + Gn_{IMU} \tag{7}$$

Where F and G are:

$$F = \begin{bmatrix} \lfloor \omega_\times \rfloor & -I_3 & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times1} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ -C(\hat{{}^I_G q})^T \lfloor \hat{a}_\times \rfloor & 0_{3\times3} & 0_{3\times3} & -C(\hat{{}^I_G q})^T & 0_{3\times1} \\ 0_{3\times3} & 0_{3\times3} & I_3 & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \end{bmatrix} \tag{8}$$

$$G = \begin{bmatrix} -I_3 & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & I_3 & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & -C(\hat{{}^I_G q})^T & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & I_3 \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \end{bmatrix} \tag{9}$$

V. Predict New State

The "predict new state" function utilizes an Extended Kalman Filter (EKF) for prediction. It estimates IMU orientation, velocity, and position by integrating gyroscope, accelerometer, and IMU measurements with a fourth-order Runge-Kutta method and adaptive time step. Intermediate variables (k1, k2, k3, k4) are computed to update IMU orientation, velocity, and position.

VI. State Augmentation

The "state augmentation" function adds a new camera state to the server, updates the covariance matrix, and ensures symmetry. It calculates IMU-to-camera rotation and translation, updates camera state, and adjusts the covariance matrix accordingly, crucial for consistency between IMU and camera states in the INS implementation.

$$^G p_C = {}^G p_I + C({}^C_G q)^T {}^I p_C \tag{10}$$

$${}^C_G q = {}^C_I q \otimes {}^I_G q \tag{11}$$

where ${}^G p_I$ is the position of the IMU, ${}^C_I \mathbf{q}$ and ${}^I_G \mathbf{q}$ 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.

$$J = [J_1 \quad 0_{6\times6N}] \tag{12}$$

$$J_1 = \begin{bmatrix} C({}^I_C q) & 0_{3\times9} & 0_{3\times3} & I_3 & 0_{3\times3} \\ \lfloor C({}^I_G q)^T {}^I p_{C\times} \rfloor & 0_{3\times9} & I_3 & 0_{3\times3} & I_3 \end{bmatrix} \tag{13}$$

$$P_{k|k} = \begin{bmatrix} I_{21+6N} \\ J \end{bmatrix} P_{K|K} \begin{bmatrix} I_{21+6N} \\ J \end{bmatrix}^\mathsf{T} \tag{14}$$

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.

VII. Adding Feature Observation

The "add feature observations" function updates the map server in visual-inertial odometry by adding new features, updating existing ones, and calculating the tracking rate.

$$Z_j^i = \begin{bmatrix} u_j^{i,1} \\ v_j^{i,1} \\ u_j^{i,2} \\ v_j^{i,2} \end{bmatrix} \tag{15}$$

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

VIII. Measurement State

The measurement update function computes the Kalman gain K using measurement matrix H and residual matrix r. It then updates the IMU state, camera state, and state covariance matrix P. If the number of features exceeds the number of state components, QR decomposition is used for matrix H to obtain Q and TH.

The measurement matrix H consists of block rows $H(j)$, where $j = 1, \ldots, L$ 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:

$$H = \begin{bmatrix} Q_1 & Q_2 \end{bmatrix} \begin{bmatrix} T_H \\ 0 \end{bmatrix} \tag{16}$$

The residual r_n is calculated by projecting the original residual r onto the column space of Q_1:

$$r_n = Q^T r = T_H \tilde{X} + n_n$$

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

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

$$R_n = \sigma_{\text{im}}^2 I_{q\times q}$$

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

The Kalman gain K is typically computed using:

$$K = PT\,{}^T_H (T_H P T\,{}^T_H + R_n)^{-1}$$

Once Kalman Filter (K) is obtained, it is used to compute the correction for the state:

$$\Delta X = K r_n$$

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

$$P_{k+1|k+1} = (I - KH) P_{k|k} (I - KH)^T + K R_n K^T$$

IX. Future Research

Research problems for classical Visual-Inertial Odometry (VIO) approaches:

Feature Tracking: Improve robustness in tracking features across different environments.

Sensor Fusion: Enhance accuracy in integrating camera and

IMU data, including calibration and synchronization.
Dynamic Environment Handling: Develop methods to handle moving objects in the scene.
Scale Ambiguity and Drift: Mitigate scale ambiguity and drift accumulation over time.
Real-time Optimization: Optimize computational efficiency for real-time performance.
Adapting to Conditions: Adapt algorithms to challenging conditions like low-light or adverse weather.
Long-term Localization: Ensure accurate localization and mapping over extended periods.
Robustness to Sensor Failures: Enhance reliability in the face of sensor failures or degradation.

X. Results

We plot the error between Ground Truth and Estimate Trajectory with the rpg trajectory evaluation toolbox. The absolute median trajectory error (ATE) is 0.07072745620375322 m and the root mean square translation error (RMSE) is 0.08225543715390592 m

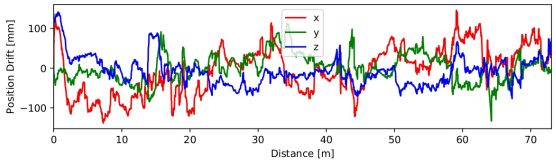

Fig. 1. *Translation Error*

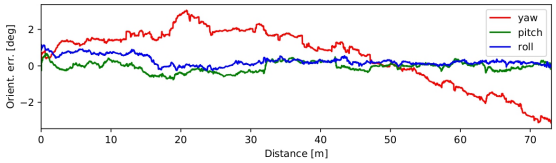

Fig. 2. *Rotation Error*

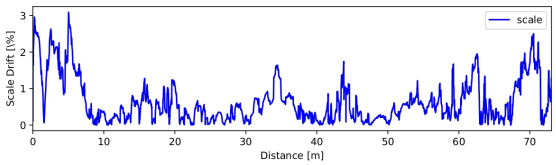

Fig. 3. *Scale Drift*

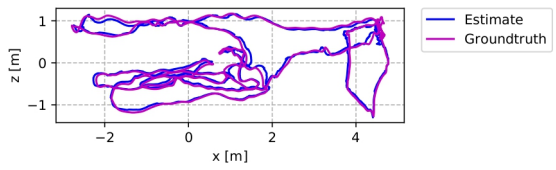

Fig. 4. *Trajectory Side View*

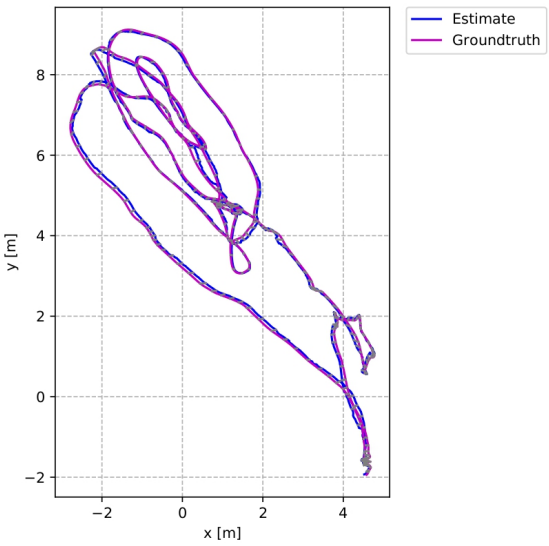

Fig. 5. *Trajectory Top View*

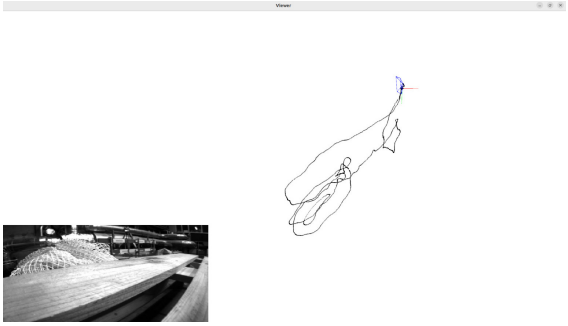

Fig. 6. *Output Trajectory*

References

[1] K. Sun, K. Mohta, B. Pfrommer, M. Watterson, S. Liu, Y. Mulgaonkar, C.J. Taylor, V. Kumar, *Robust Stereo Visual Inertial Odometry for Fast Autonomous Flight* 2018.
[2] A.I. Mourikis, S.I. Roumeliotis, *A Multi-State Constraint Kalman Filter for Vision-aided Inertial Navigation* .
[3] Z. Zhang, D. Scaramuzza, *A Tutorial on Quantitative Trajectory Evaluation for Visual(-Inertial) Odometry* 2018.

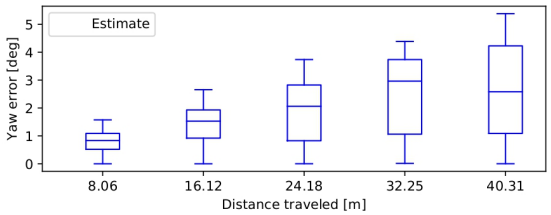

Fig. 7. *Relative Yaw Error*

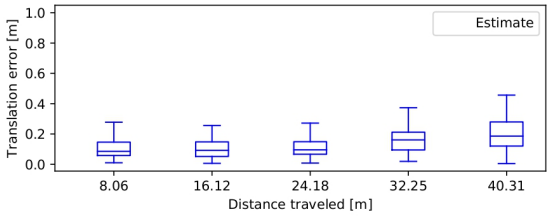

Fig. 8. *Relative Translation Error*