RBE/CS549: Project 4 - Classical Visual-Inertial Odometry

Sanya Gulati
sgulati@wpi.edu

Siyuan Huang
shuang4@wpi.edu

Yijia Wu
ywu21@wpi.edu

I. Introduction

In this project, we implemented the seven functions of stereo Multi-State Constraint Kalman Filter (MSCKF)[1]. As this paper is an extension of the original MSCKF paper[2], we have taken it as a reference.

II. Initialize gravity and bias

The gravity and gyroscope bias are initialized with the first 200 messages from IMU when the robot is placed statically. The gyroscope bias b_g is initialized as the average of these 200 gyroscope readings. The gravity g is initialized as $[0, 0, g_{norm}]$, and g_norm is the norm of the first 200 accelerometer readings. The orientation is initialized as the quaternion from $-g$ to g_{norm}.

III. Batch IMU processing

When we receive a new feature, we want to first process all the IMU messages received prior to the new feature's time stamp. For every IMU message within the IMU message buffer that's received prior to the feature time stamp, we update our state estimation using the process model.

IV. Process model

We model the IMU system model as a continuous-time in 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 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

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

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

The gyroscope and accelerometer measurements, ω_m and a_m can be written as

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

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

where ${}^I_G R$ is the rotation matrix calculated from quaternion ${}^I_G\bar{q}$ Then apply expectation operator on equation 1 we obtain the equations for propagating IMU state estimates:

$$\begin{aligned}
{}^I_G\dot{\hat{\bar{q}}}(t) &= \frac{1}{2}\Omega(\hat{\omega}^I_G)\hat{\bar{q}},\\
\dot{\hat{b}}_g &= 0_{3\times1},\\
{}^G\dot{\hat{v}}_I &= C^T_{\hat{q}}\hat{a} - 2\lfloor\omega_{G\times}\rfloor{}^G\hat{v}_I + \lfloor\omega_{G\times}\rfloor^{2G}\hat{p}_I +{}^G g,\\
\dot{\hat{b}}_a &= 0_{3\times1},\\
{}^G\dot{\hat{p}}_I &={}^G \hat{v}_I
\end{aligned} \tag{6}$$

The linearized continuous-time model for IMU error-state is:

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

Where F and G are

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

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

V. Predict new state

Every time we received a new IMU measurement, we use 4^{th} order Runge-Kutta numerical integration to propagate our estimation for the next state.

VI. State augmentation

In the state_augmentation function, we update the camera position ${}^G p_C$ and orientation ${}^C_G q$ with the last IMU state and augment the state covariance matrix P.

The pose of the camera can be computed as,

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

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

$$ J = \begin{bmatrix} J_1 & O_{6\times 6N} \end{bmatrix} \tag{12}$$

$$ J_1 = \begin{bmatrix} C({}^I_C q) & 0_{3\times 9} & 0_{3\times 3} & I_3 & 0_{3\times 3} \\ \lfloor C({}^I_G q)^T \; {}^I p_C \times \rfloor & 0_{3\times 9} & I_3 & 0_{3\times 3} & I_3 \end{bmatrix} \tag{13}$$

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

VII. Add feature observations

This function can update the detected feature in the latest frame to the feature map. Each feature will be added to the feature map with its feature ID and current state ID. The feature that was collected with state ID i and feature ID j is represented as

$$ Z_i^j = \begin{bmatrix} u^j_{i,1} & v^j_{i,1} & u^j_{i,2} & v^j_{i,2} \end{bmatrix}^T \tag{15}$$

1 represents the left camera, and 2 represents the right camera.

VIII. Measurement update

The measurement_update function takes measurement matrix H and residual matrix r as input to compute the Kalman gain K, and then updates the IMU state X_{IMU}, camera state, and state covariance matrix P.

The measurement matrix H is a matrix with block rows $H^(j)$, $j = 1...L$. L is the number of all detected features. If the number of row (feature) is larger than the number of state X components (21), we employ QR decomposition for the matrix H. With the reduced mode of numpy.linalg.qr function, we can directly get Q and T_H.

$$ H = \begin{bmatrix} Q & Q_2 \end{bmatrix} \begin{bmatrix} T_H \\ O \end{bmatrix} \tag{16}$$

The matrix Q can then be used to compute residual r_n as

$$ r_n = Q^T r = T_H \widetilde{X} + n_n \tag{17}$$

R_n is the covariance matrix of the noise vector n_n. σ^2_{im} is the noise of observation. q is the number of rows of matrix Q.

$$ R_n = \sigma^2_{im} I_{q\times q} \tag{18}$$

The Kalman gain K can be computed with the following equation

$$ K = PT_H^T (T_H P T_H^T + R_n)^(-1) \tag{19}$$

However, because the matrix inverse computation is unstable, we can change it as solving the $Ax = b$ problem, x is K^T, A is $S = T_H P T_H^T + R_n$, and b is $T_H P$. A is S because $S = S^T$, both matrix P and R_n are symmetric.

$$ SK^T = T_H P \tag{20}$$

After getting the Kalman gain K, we can use it to compute the correction for state ΔX as

$$ \Delta X = K r_n \tag{21}$$

Finally, the state covariance matrix P can be updated with

$$ P_{k+1|k+1} = (I_{k\times k} - K T_H) P_{k|k} \tag{22}$$

IX. Trajectory error evaluation

In this section, we plot the error between Ground Truth and Estimate Trajectory with the rpg_trajectory_evaluation toolbox[3]. The absolute median trajectory error (ATE) is 0.06910338087405558 m and the root mean square translation error (RMSE) is 0.081715762810781 m.

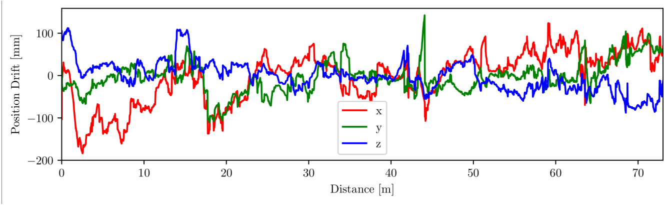

Fig. 1: Translation Error

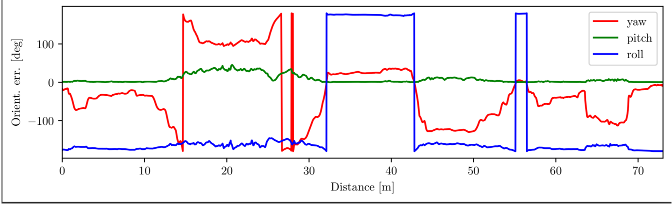

Fig. 2: Rotation Error

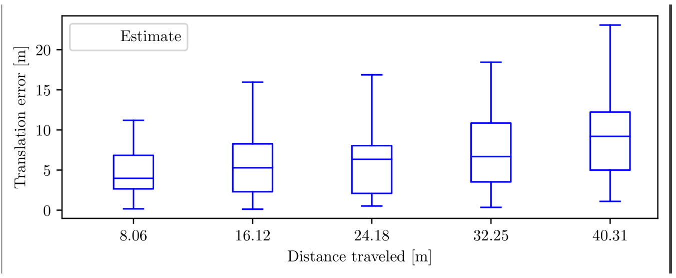

Fig. 3: Relative Translation Error

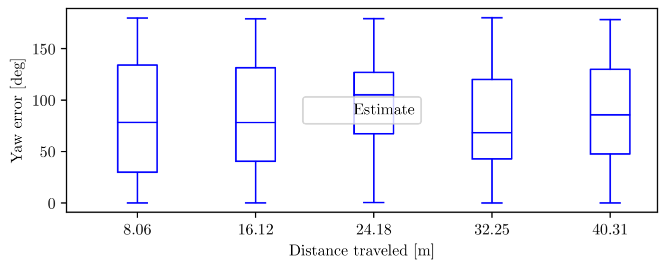

Fig. 4: Relative Yaw Error

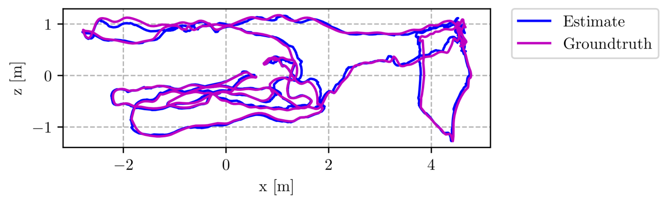

Fig. 5: Trajectory Side View

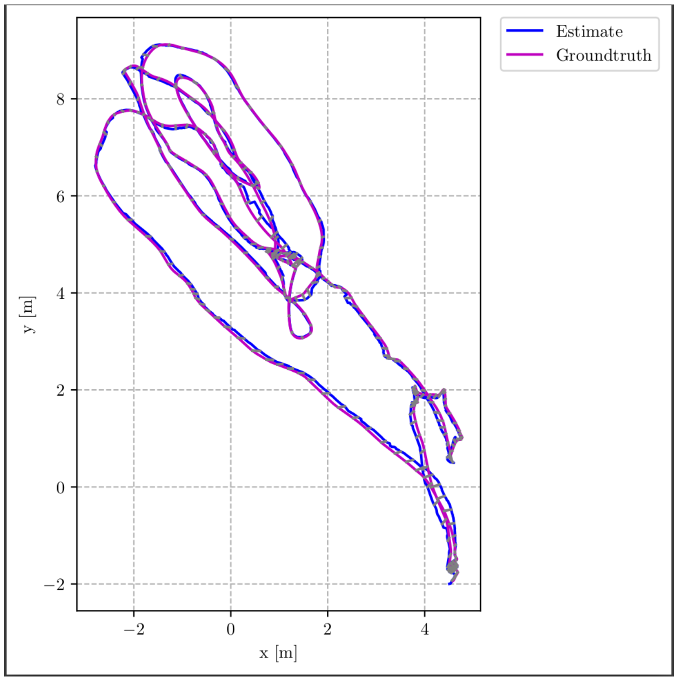

Fig. 6: Trajectory Top View

References

[1] K. Sun et al., "Robust Stereo Visual Inertial Odometry for Fast Autonomous Flight." arXiv, 2017. doi: 10.48550/ARXIV.1712.00036.

[2] A. I. Mourikis and S. I. Roumeliotis, "A Multi-State Constraint Kalman Filter for Vision-aided Inertial Navigation," Proceedings 2007 IEEE International Conference on Robotics and Automation. IEEE, Apr. 2007. doi: 10.1109/robot.2007.364024.

[3] Z. Zhang and D. Scaramuzza, "A Tutorial on Quantitative Trajectory Evaluation for Visual(-Inertial) Odometry," 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, Oct. 2018. doi: 10.1109/iros.2018.8593941.