RBE/CS549 Computer Vision

Project 4 - VIO

Anagha Dangle - Mihir Kulkarni
ardangle@wpi.edu - mmkulkarni@wpi.edu

***Abstract*—The task of this project is to perform visual-inertial odometry, which is a fusion of camera and imu data to bring about robust odometry for a robot. The complementary characteristics of a camera and an imu are used for the same. We use the Multi-State Constraint Kalman Filter (S-MSCKF) approach which is a state-of-the-art method to perform VIO. It used a stereo camera and an IMU. It is the favored method for MAV platforms since it can function well in GPS-denied conditions and requires a much smaller and lighter sensor package than lidar-based techniques.**

***Index Terms*—Visual-Inertial Odometry, MSCKF, IMU, Stereo**

I. Functions implemented

The following functions are implemented - $initializegravityandbias$ (estimates gravity and bias using initial few measurements of IMU), $batchimuprocessing$ (Processes the messages in the $imumsgbuffer$, executes the process model and updates the state), $processmodel$ (Dynamics of the IMU error state), $predictnewstate$ (Handles the transition and covariance matrices), $stateaugmentation$ (Adds the new camera to state and updates), $addfeatureobservations$ (Adds the image features to the state), $measurementupdate$ (Updates the state using the measurement model) and $predictnewstate$ (Propogates the state using 4th order Runge-Kutta).

A. Initialize Gravity and Bias

In Visual Inertial Odometry, the IMU state is defined as.

$$\mathbf{X}_{\text{IMU}} = \begin{bmatrix} {}^{T}_{G}\bar{q}^{T} & {\mathbf{b}_g}^{T} & {G_{\mathbf{v}_f}}T & {\mathbf{b}_a}^{T} & G^{T}_{\mathbf{p}_I} \end{bmatrix}^{T} \quad (1)$$

The vectors b_g and b_a are the biases of the measured angular velocity and linear acceleration from the IMU. The white Gaussian noise vectors $n_w g$ and $n_w a$, are used to drive the random walk processes which model the IMU biases. In the IMU frame, gravity is also defined. In order for the estimation to be consistent with the inertial frame, initialize the initial orientation as well. It also makes use of the rotation quaternion from v_0 to v_1.

B. Batch IMU Processing

The time evolution of the IMU state is described by:

$$I_G\dot{\bar{q}}(t) = \tfrac{1}{2}\Omega\big(\omega(t)\big)\,{}^{I}_{G}\bar{q}(t), \quad \hat{\mathbf{b}}_g(t) = \mathbf{n}_{wg}(t) \quad (2)$$

$${}^{G}\hat{\mathbf{v}}_I(t) = {}^{G}_{\mathbf{a}}(t), \quad {}^{\mathbf{b}}{}_a(t) = \mathbf{n}_{wa}(t), \quad G^{*}_{\mathbf{p}I}(t) = G_{\mathbf{V}I}(t) \quad (3)$$

In this function, we continuously evolve the process model with increments in time. The process model function is called continuously. this is done until a certain timestamp Is reached. After a finite time, the state is updated and the msg buffer is cleared. The process model is explained below.

C. Process Model

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

$$\tilde{\mathbf{X}}_{\text{IMU}} = \mathbf{F}\tilde{\mathbf{X}}_{\text{IMU}} + \mathbf{G}\mathbf{n}_{\text{IMU}} \quad (4)$$

where

$$\mathbf{n}_{\text{IMU}} = \begin{bmatrix} \mathbf{n}_g^T & \mathbf{n}_{wag}^T & \mathbf{n}_a^T & \mathbf{n}_{wa}^T \end{bmatrix}^T \quad (5)$$

is the system noise. The covariance matrix of $n_I MU$, $Q_I MU$, depends on the IMU noise characteristics and is computed off-line during sensor calibration. Finally, the matrices F and G that appear in Eq. (10) are given by:

$$\mathbf{F} = \begin{bmatrix} -\lfloor\hat{\boldsymbol{\omega}}\times\rfloor & -\mathbf{I}_3 & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ -\mathbf{C}^T_{\hat{q}}\lfloor\hat{\mathbf{a}}\times\rfloor & \mathbf{0}_{3\times3} & -2\lfloor\boldsymbol{\omega}_G\times\rfloor & -\mathbf{C}^T_{\hat{q}} & -\lfloor\boldsymbol{\omega}_G\times\rfloor^2 \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{I}_3 & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \end{bmatrix}$$

Fig. 1

$$\mathbf{G} = \begin{bmatrix} -\mathbf{I}_3 & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{I}_3 & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & -\mathbf{C}^T_{\hat{q}} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{I}_3 \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \end{bmatrix}$$

Fig. 2

If dt is within 0.01s, our 3rd-order approximation of the matrix exponential can be considered precise enough. Use Runge-Kutta of the fourth order to propagate the state. We use a 4th-order Runge-Kutta numerical integration of the following equation to propagate the estimated IMU state in order to handle discontinuous time measurements from the IMU. The discrete-time state transition matrix of the Equation and the discrete-time noise covariance matrix must first be constructed in order to convey the state's uncertainty. Then we propagate the state covariance matrix and make the covariance symmetric. Finally, we update the state corresponding to the null space.

D. Predict new state

We use the following equations to propagate to new IMU state: k1 = f(tn, yn)
k2 = f(tn+dt/2, yn+k1*dt/2)
k3 = f(tn+dt/2, yn+k2*dt/2)
k4 = f(tn+dt, yn+k3*dt)
yn+1 = yn + dt/6*(k1+2*k2+2*k3+k4)

E. State Augmentation

When a new image is captured, the camera pose estimate is calculated from the IMU pose estimate as follows:

$$ {}^{G}\hat{\mathbf{p}}_C = {}^{G}\hat{\mathbf{p}}_I + \mathbf{C}_q^{T\backslash}\hat{\mathbf{p}}_C \tag{6} $$

The quaternion expressing the rotation between the IMU and camera frames is q, and the position of the camera frame's origin with respect to I is IpC, both of which are known. This new camera pose estimate is appended to the state vector. The state covariance matrix is then updated. The Jacobian is given as follows:

$$ \mathrm{J} = \begin{bmatrix} \mathbf{C}\left(\frac{C}{I}\bar{q}\right) & \mathbf{0}_{3\times9} & \mathbf{0}_{3\times6} & \mathbf{0}_{3\times6N} \\ \left[\, C_{\bar{q}}^{T} I\mathbf{p}_C \times\right] & \mathbf{0}_{3\times9} & \mathbf{1}_3 & \mathbf{0}_{3\times6N} \end{bmatrix} \tag{7} $$

F. Add feature observations

A feature message contains the image's timestamp, cam0 and cam1 frames, and cam0 and cam1 messages. The map server is used to determine whether the detected feature is a new or existing feature before adding it. Following the addition of the feature to the map server, the tracking rate is updated.

G. Measurement update

H_{Xi}^{j} and H_{fi}^{j} are the Jacobians of the measurement z (j) I with respect to the state and feature position, respectively, in the preceding expression, and Gpefj is the error in the position estimate of fj. Jacobians are provided by: The final Jacobian matrix is then decomposed using the QR decomposition method to reduce computational complexity. The Kalman gain for state covariance is computed first, followed by the state error. Finally, the IMU state, camera state, and state covariance are all updated.

$$ \mathbf{H}_{\mathbf{X}_i}^{(j)} = \begin{bmatrix} 0_{2\times15} & 0_{2\times6} & \cdots & P_i^{(j)} | C_i^c \hat{\mathbf{X}}_{f_j} \times | & -\mathbf{J}_i^{(j)} \mathbf{C}(\mathrm{C}_i^c \hat{q}) & \cdots \end{bmatrix} \tag{8} $$

$$ \mathbf{H}_{ji}^{(j)} = \mathbf{J}_i^{(j)} \mathbf{C}(\mathbf{\Lambda}_G^{(C)} \hat{q}) \tag{9} $$

II. Error Calculation

Th RMSE ATE Error to be estimated is done using rpg Repository. This repository implements common used trajectory evaluation methods for visual-inertial odometry. The plots are illustrated in the figures. SE(3) alignment is done.

The procedure to run the toolbox involved convering the groundtruth text files to their specified file formats. Also the estimated IMU states are added in CSV file which is later convert to .txt. The .yaml file is edited to specify SE3 alignment. The README.md has specified all the steps to be followed .

III. Results

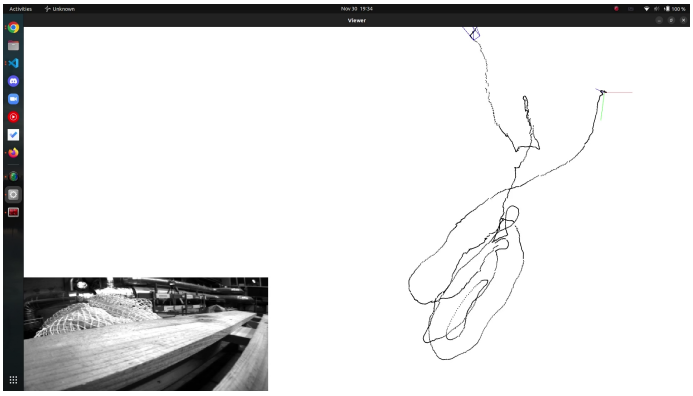

Fig. 3: Final Output

Fig. 4: Initial Camera Pose

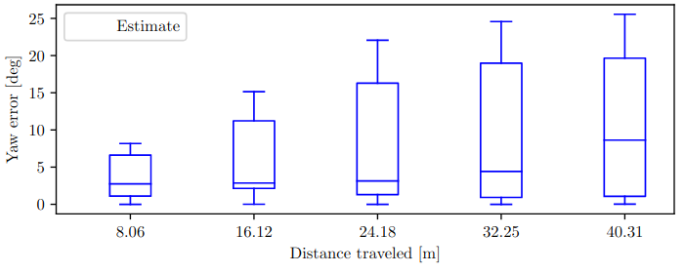

Fig. 5: Relative Yaw error

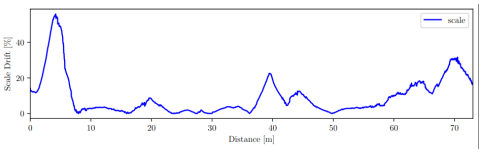

Fig. 6: Scale Error

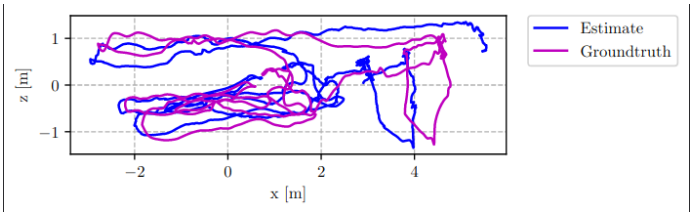

Fig. 7: Side trajectory plot

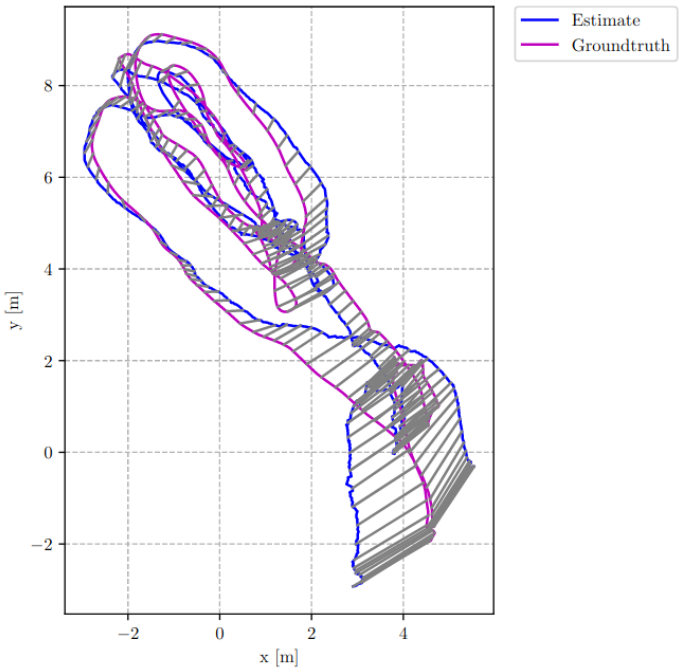

Fig. 8: Top trajectory plot

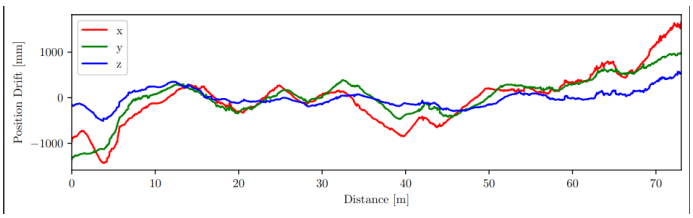

Fig. 9: Translation error