RBE 549 Project 4 Phase 1: Classical Visual Inertial Odometry

Aabha Tamhankar
Masters in Robotics Engineering
Worcester Polytechnic Institute
astamhankar@wpi.edu

Miheer Diwan
Masters in Robotics Engineering
Worcester Polytechnic Institute
msdiwan@wpi.edu

***Abstract*—MSCKF (Multi-State Constraint Kalman Filter) is an EKF-based tightly-coupled visual-inertial odometry algorithm. S-MSCKF is MSCKF's stereo version, its results on tested datasets are comparable to state-of-art methods including OKVIS, ROVIO, and VINS-MONO. This project is a Python reimplementation of S-MSCKF, the code is directly translated from the official C++ implementation.**

I. Introduction

Due to advances in robotics in the field of aerial vehicles, state estimation is crucial for gaining pose and achieving stability of the robot while in flight. The combination of visual information from cameras and measurements from an Inertial Measurement Unit (IMU) is referred to as Visual Inertial Odometry (VIO)[1]. This project presents an implementation of VIO using Multi-State Constraint Kalman Filter (MSCKF), which compensates for the high cost of sensors and processing required for other implementations of VIO. In this method, MSCKF is used to determine state and localization of the robot using sensor fusion of IMU and a stereo camera.

A. Dataset

The Machine Hall 01 easy dataset was used for implementing this project. The data was collected on board a Micro Aerial Vehicle (MAV). The datasets contain stereo images, synchronized IMU measurements, and accurate motion and structure ground truth. It is a subset of the EuRoC dataset and the ground truth is provided by a sub-mm accurate Vicon Motion capture system.

II. Estimator Description

The goal of the estimator is to track the 3D pose of the IMU-affixed frame I with respect to a global frame of reference G. The MSCKF contains an algorithm with various functions which are illustrated in Fig.1.

Each of the functions has been explained in detail through the next few sections.

A. Initializing Gravity and Bias

The IMU sensor is a 6DOF sensor, that is, 3 DOF for the gyroscope which measures rotation, and 3 DOF for the accelerometer which measures acceleration. However, being mechanical systems, they face uncertainties and biases, which need to be compensated for further calculations.
Theoretically, the gyroscope should read [0, 0, 0], and the accelerometer due to the involvement of gravity should read [0, 0, -g]. However, as mentioned above, we see deflection in the measured values. These biases can be removed by taking multiple readings of both, and taking their meanwhile keeping the rotor stationary. This mean value can then be subtracted from the next IMU readings. In this way, the IMU is calibrated before the rest of the operations

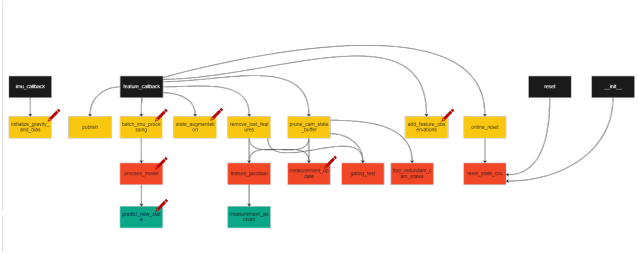

Fig. 1: Function Structure of MSCKF

B. Batch IMU Processing

For the further steps of performing VIO like executing the process model and updating the states as the vehicle is in flight, it is important to first define the state vector and its error state vector. The state vector consists of the states in the camera as well as IMU, which is given as

$$\hat{\mathbf{X}}_k = \begin{bmatrix} \hat{\mathbf{X}}_{\text{IMU}_k}^T & {}_{G}^{C_1}\hat{q}^T & {}^G\hat{\mathbf{p}}_{C_1}^T & \cdots & {}_{G}^{C_N}\hat{q}^T & {}^G\hat{\mathbf{p}}_{C_N}^T \end{bmatrix}^T$$

where q and p are the estimates of the camera attitude and position for N camera states.
The state error vector is given in

$$\tilde{\mathbf{X}}_k = \begin{bmatrix} \tilde{\mathbf{X}}_{\text{IMU}_k}^T & \boldsymbol{\delta\theta}_{C_1}^T & {}^G\tilde{\mathbf{p}}_{C_1}^T & \cdots & \boldsymbol{\delta\theta}_{C_N}^T & {}^G\tilde{\mathbf{p}}_{C_N}^T \end{bmatrix}^T$$

The function helps to predict the next state and update the state information using the process model for a given time step in IMU messages. This state information is updated after every N state for the data.

C. Process Model

The filter propagation equations are derived by discretization of the continuous-time IMU system model. The time evolution of IMU state dynamics is given by

$$
\begin{aligned}
{}^{I}_{G}\dot{\hat{\mathbf{q}}} &= \frac{1}{2}\Omega(\hat{\boldsymbol{\omega}})\,{}^{I}_{G}\hat{\mathbf{q}}, \quad \dot{\hat{\mathbf{b}}}_g = \mathbf{0}_{3\times 1}, \\
{}^{G}\dot{\hat{\mathbf{v}}} &= C\left({}^{I}_{G}\hat{\mathbf{q}}\right)^{\top}\hat{\mathbf{a}} + {}^{G}\mathbf{g}, \\
\dot{\hat{b}}_a &= \mathbf{0}_{3\times 1}, \quad {}^{G}\dot{\hat{\mathbf{p}}}_I = {}^{G}\hat{\mathbf{v}}, \\
{}^{I}_{C}\dot{\hat{\mathbf{q}}} &= \mathbf{0}_{3\times 1}, \quad {}^{I}\dot{\hat{\mathbf{p}}}_C = \mathbf{0}_{3\times 1}
\end{aligned}
$$

where $\hat{\omega}$ and $\hat{a}$ are are the IMU measurements for angular velocity and acceleration respectively with biases removed, that is, $\hat{\omega} = \omega_m - b_g$ and $\hat{a} =_m - b_g$.

The quarternion detivative Ω is given by

$$
\Omega\left(\hat{\boldsymbol{\omega}}\right) = \begin{pmatrix} -[\hat{\boldsymbol{\omega}}_{\times}] & \boldsymbol{\omega} \\ -\boldsymbol{\omega}^{\top} & 0 \end{pmatrix}
$$

where $\hat{\omega}$ is a skew-symmetric matrix of its vector. The error dyanamics of IMU error state can be given by

$$
\dot{\tilde{X}}_I = F\tilde{X}_I + Gn_I
$$

The F matrix is used to derive a discrete-time state time state transition matrix. The G matrix is used to obtain the discrete time noise covariance matrix. ϕ_K is approximated by Taylor's expansion till 3rd order of F, and Q_K is a discrete-time state covariance matrix obtained by continuous time methods of state covariance Q and G matrix. The state matrix is used by converting it into a symmetric matrix.

D. New State Prediction

This function handles the transition and covariance matrices. We start by computing the norm of the gyroscope value and extracting the current orientation, velocity, and position from the IMU. We then propagate the state using the 4^{th} order Runge-Kutta method for equation (1) in 2. The new orientation, velocity, and position values for the next state are calculated, and the state of the IMU is updated. The pose of the new camera state can be computed as,

$$
{}^{C}_{G}\hat{\mathbf{q}} = {}^{C}_{I}\hat{\mathbf{q}} \otimes {}^{I}_{G}\hat{\mathbf{q}}, \quad {}^{G}\hat{\mathbf{p}}_C = {}^{G}\hat{\mathbf{p}}_C + C\left({}^{I}_{G}\hat{\mathbf{q}}\right)^{\top}{}^{I}\hat{\mathbf{p}}_C
$$

The augmented covariance matrix, J and J_I are shown below

$$
\mathbf{P}_{k|k} = \begin{pmatrix} \mathbf{I}_{21+6N} \\ \mathbf{J} \end{pmatrix} \mathbf{P}_{k|k} \begin{pmatrix} \mathbf{I}_{21+6N} \\ \mathbf{J} \end{pmatrix}^{\top}
$$

$$
\mathbf{J} = \begin{pmatrix} \mathbf{J}_I & \mathbf{0}_{6\times 6N} \end{pmatrix}
$$

$$
\mathbf{J}_I = \begin{pmatrix} C\left({}^{I}_{G}\hat{\mathbf{q}}\right) & \mathbf{0}_{3\times 9} & \mathbf{0}_{3\times 3} & \mathbf{I}_3 & \mathbf{0}_{3\times 3} \\ -C\left({}^{I}_{G}\hat{\mathbf{q}}\right)^{\top}\lfloor {}^{I}\hat{\mathbf{p}}_{C\times}\rfloor & \mathbf{0}_{3\times 9} & \mathbf{I}_3 & \mathbf{0}_{3\times 3} & \mathbf{I}_3 \end{pmatrix}
$$

E. State Augmentation

The state augmentation function adds the new camera to the state and updates it. We start extracting the IMU state, the rotation from the IMU to the camera (previous state: cam0), and the translation from the camera (previous state: cam0) to the IMU. The camera state at that particular instant of time is also calculated and stored in a variable called 'time'. We then update the covariance matrix of the state with the help of the J matrix shown above, resize it and fill in the augmented state covariance. We also make the state covariance matrix symmetric by adding a constraint.

F. Adding Feature Observations

This function is used to get the current IMU state ID and the number of current features. All the features in the 'feature_msg' are then added to the 'self.map_server'. With the help of the old features and the new tracked features, we calculate and update the tracking rate.

G. Measurement Update

In the Measurement Update function, we compute the residuals by stacking multiple observations of the same feature.

$$
\mathbf{r}^j = \mathbf{H}^j_{\mathbf{x}}\tilde{\mathbf{x}} + \mathbf{H}^j_f\,{}^{G}\tilde{\mathbf{p}}_j + \mathbf{n}^j
$$

First, we check if the H and r matrices are empty. If not, we decompose the final Jacobian matrix to reduce the computational complexity. We use the outputs from this step to update the Kalman gains and compute the error of the state. After this, we finally update the IMU state, the camera states, and the state covariance and fix the covariance to be symmetric with a small constraint.

III. Results

The resulting trajectory was plotted against a video of the aerial vehicle in flight using the given in Machine Hall 01 easy dataset which is a subset of the EuRoC dataset. The trajectory was generated using OpenGL and Pangolin as demonstrated in [3], and can be seen in Fig.2. A video of the trajectory was also recorded and saved in the file "Output.mp4".

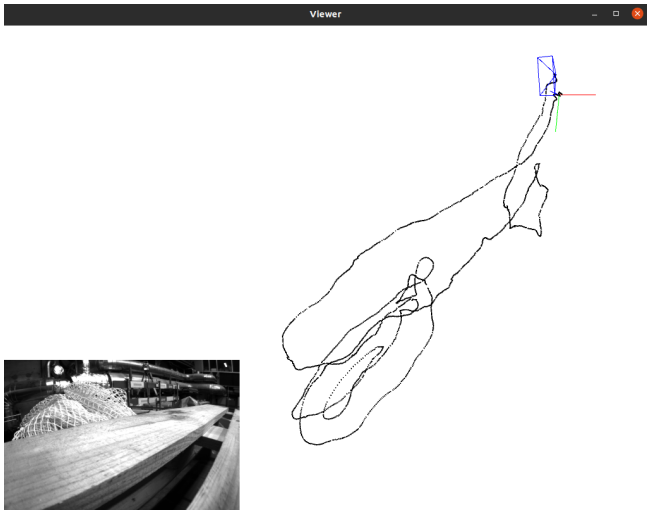

Fig. 2: Output of S-MSCKF for EuRoC MH01Easy Using Pangolin

This trajectory can be easily compared with the ground truth and estimated errors can be calculated. The ”data.csv” file from the given dataset contains the transformations and quaternions for each timestamp in the flight. This file can be used as the data for the ground truth trajectory. A similar ”estimated.txt” file is generated through the given starter code, which is all the estimated values of transformations and quaternions. Using these two files in the format required by [4], the plots for estimated against ground-truth can be generated. The compared trajectories are illustrated in Fig.3 and Fig.4. This method also provides rotations and translation errors in the form of graphs.

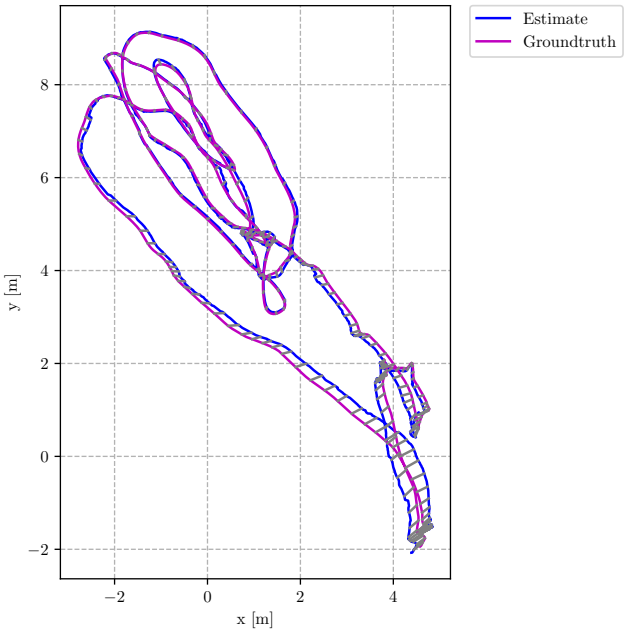

Fig. 3: Evaluation of output of S-MSCKF with respect to ground truth for EuRoC MH01Easy

REFERENCES

[1] Ke Sun, Kartik Mohta, Bernd Pfrommer, Michael Watterson, Sikang Liu, Yash Mulgaonkar, Camillo J. Taylor, and Vijay Kumar, ”Robust Stereo Visual Inertial Odometry for Fast Autonomous Flight.”
[2] Anastasios I. Mourikis and Stergios I. Roumeliotis, ”A Multi-State Constraint Kalman Filter for Vision-aided Inertial Navigation”
[3] https://github.com/uoip/stereo$_m sckf$
[4] https://github.com/uzh-rpg/rpg$_t rajectory_e valuation$

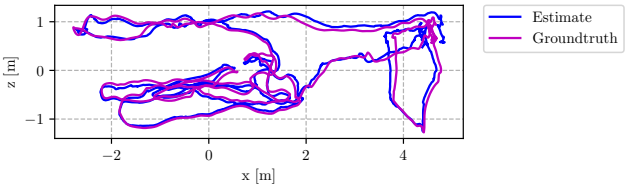

Fig. 4: Evaluation of output of S-MSCKF with respect to ground truth for EuRoC MH01Easy