RBE/CS549: Project 4
Deep and Un-Deep Visual Inertial Odometry

Prasanna Natu
M.S. Robotics Engineering
Worcester Polytechnic Institute
Worcester, MA
pvnatu@wpi.edu

Peter Dentch
M.S. Robotics Engineering
Worcester Polytechnic Institute
Worcester, MA
pdentch@wpi.edu

***Abstract*—This project aims to implement Vision-aided Odometry utilizing the Multi-state Constraint Kalman Filter (MSCKF) approach. The motivation behind adopting MSCKF for Visual Inertial Odometry lies in addressing the challenges faced by autonomous vehicles in employing high-quality sensors and efficient processors. By implementing this approach, it is possible to achieve accurate state estimation and localization without relying on costly and heavy sensor systems. In this project, a filter-based method is developed, fusing data from two sensors: a stereo camera and an IMU. The MSCKF is employed to estimate the state of the robot, while sensor fusion of the IMU and stereo camera are used to determine its localization. In the Phase 2 we try deep Learning appraoches to overcome the drawback and get a generalised model for Visual Inertial Odometry.**

***Index Terms*—Visual Inertial Odometry, Multi-state Constraint Kalman Filter, MSCKF, sensor fusion, stereo camera, inertial measurement unit, IMU.**

I. Phase I

A. INITIALIZING GRAVITY AND BIAS

The 6-DOF IMU sensor utilized in this project is prone to biases that must be rectified in each reading. The 6-DOF represents three Degrees of Freedom for rotation (gyroscope) and three Degrees of Freedom for acceleration (accelerometer). This correction process can be considered for the calibration of the IMU sensor, where the biases in both rotation and acceleration are calculated and subsequently subtracted from every subsequent IMU reading.

This calibration is performed by keeping the rotor stationary and collecting approximately 100 - 200 readings, followed by calculating the mean of these readings. Ideally, the gyroscope reading should be [0, 0, 0]; however, due to noise and biases, small fluctuations in the gyroscope readings are observed.

To correct for these fluctuations, the mean of the collected readings is subtracted from the subsequent IMU readings while the rotor is stationary, as described earlier. Ideally, the accelerometer reading should be [0, 0, -g] in the world frame. Nevertheless, noise and biases in the low-cost IMU sensor cause fluctuations.

The overview of the traditional method which we implemented is represented below:

$$\mathbf{x}_I = \left(\, {}^I_G\mathbf{q}^\top \quad \mathbf{b}_g^\top \quad {}^G\mathbf{v}_I^\top \quad \mathbf{b}_a^\top \quad {}^G\mathbf{p}_I^\top \quad {}^I_C\mathbf{q}^\top \quad {}^I\mathbf{p}_C^\top \, \right)^\top \tag{1}$$

Fig. 1. State Vector

$$\tilde{\mathbf{X}}_k = \left[\; \tilde{\mathbf{X}}\mathrm{IMU}_k{}^T \quad \delta\theta C_1{}^T \quad {}^G\tilde{\mathbf{p}}C_1{}^T \quad \ldots \quad \delta\theta C_N{}^T \quad \mathbf{p}_{C_N} \; \right]^T \tag{2}$$

Fig. 2. Error State Vector

B. BATCH IMU PROCESSING

IMU batch processing is carried out to process IMU messages until the next set of images from the stereo camera becomes available. Prior to this, it is crucial to define the state vector to estimate subsequent states. The state vector comprises states in both the camera and IMU.

As depicted in Figure 1, the state vector includes the quaternion q, which describes the rotation from the global to the IMU frame. The variables bg and ba represent the biases in the gyroscope and accelerometer, respectively. The position and velocity of the body frame in the inertial (world) frame are denoted by pi and vi. The transformations between the IMU and camera frames are represented by q_{Ic} and p_{Ic}.

For N camera poses, the state vector adds a new state in the buffer, with the first element being the states associated with the IMU sensor.

The objective of the batch IMU processing function is to predict the subsequent state and update the state information using the process model for a given time step, based on the IMU messages. The state information is updated after processing the IMU data and concludes after a specified time duration.

C. Process Model

The process model predicts the IMU state using a motion model derived from error states, as shown in Figure 2. The error in quaternion is a quaternion operation, represented as:

$$\delta q = q \otimes q_0^{-1} \tag{3}$$

The $\hat{\omega}$ and $\hat{a}$ are given as follows:

$$\hat{\omega} = \omega_m - b_g \tag{4}$$

$$C^t\dot{\mathrm{q}} = \tfrac{1}{2}\Omega(\dot{\omega})_G^l\hat{\mathrm{q}}, \quad \dot{\mathrm{b}}_g = 03 \times 1,,$$

$$G_{\hat{\mathrm{v}}}^{\hat{l}} = C\left(G^l\hat{\mathrm{q}}\right)^\top \hat{\mathbf{a}} + {}^G\mathrm{g},$$

$$\dot{b}a = 03 \times 1, \quad G_{\dot{\mathrm{p}}j} = {}^a\dot{\mathrm{v}},$$

$$C\dot{\mathrm{q}}\mathrm{q} = 0_{3\times1}, \quad {}^I\dot{\mathrm{p}}C = 03 \times 1$$

Fig. 3. IMU dynamics

$$\Phi_k = \Phi\left(t_{k+1}, t_k\right) = \exp\left(\int_{t_k}^{t_k+1} \mathrm{F}(\tau)d\tau\right)$$

$$Q_k = \int_{t_k}^{t_{k+1}} \Phi\left(t_{k+1}, \tau\right)\mathrm{GQG}\Phi\left(t_{k+1}, \tau\right)^\top d\tau$$

Fig. 4. Covariance Matrix

$$\hat{a} = a_m - b_g \tag{5}$$

The other errors are additive errors that simply add to the previous quantity. These error states are employed to determine the robot's process model. Angular velocity and linear acceleration are derived, as illustrated in Figure 3, where Ω is the quaternion derivative and is expressed as:

$$\Omega(\omega) = \begin{bmatrix} \omega & \hat{\omega} \\ \omega^T & 0 \end{bmatrix} \tag{6}$$

Here, $\hat{\omega}$ is a skew-symmetric matrix of the ω vector. The linearized continuous error dynamics of the IMU error state are defined as follows:

$$\dot{\tilde{X}}_I = F\tilde{X}_I + Gn_I \tag{7}$$

The term n_I denotes the Gaussian noise of the accelerometer and gyro readings. To propagate the IMU measurement in discrete time, the 4th-order Runge Kutta method is applied.

The F matrix in the above equation (discrete-time equation) is utilized to derive the discrete-time state transition matrix, and the G matrix is employed to obtain the discrete-time noise covariance matrix. ϕ_K is approximated using a Taylor expansion up to the 3rd order of F, while Q_k is a discrete-time state covariance matrix obtained by continuous-time methods of state covariance Q and G matrix. The observability constraint is applied by modifying the transition matrix. The state transition matrix is corrected by making it symmetric in this step.

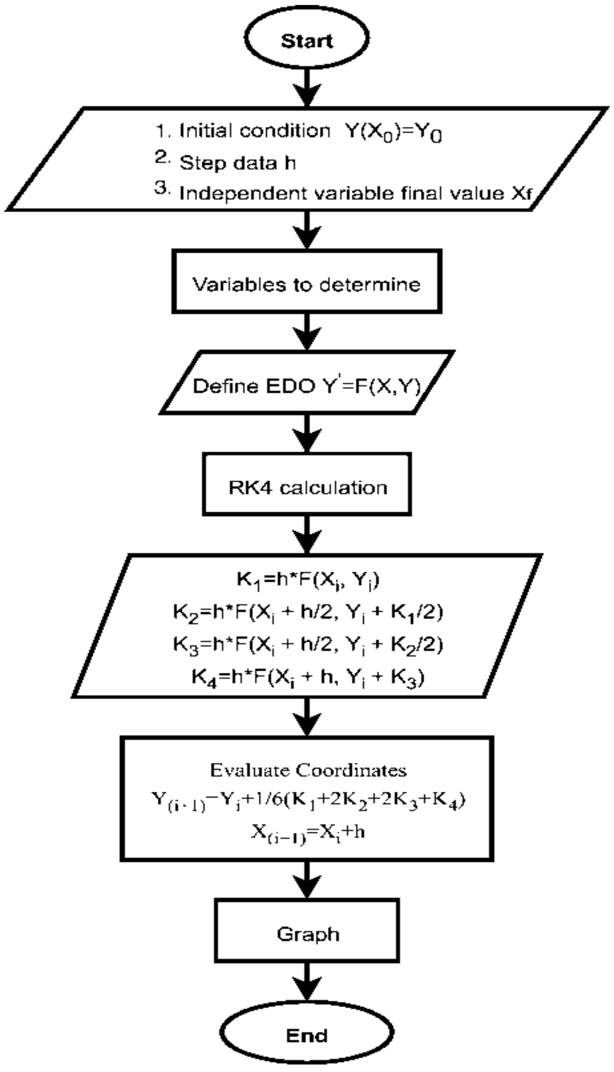


Fig. 1. Runge Kutta

D. 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.

The augmented covariance matrix is given by the following equation in Section VI, and the J matrix is given by the equation in Section VI.

E. Adding Feature Observation

Here, we check if the feature observed is in the map server dictionary or not. If it is not there, we add a new key to the map server dictionary.

F. Measurement Model and Update

A single feature f_j is observed by the stereo cameras with the pose. In the same instance, the stereo cameras have different poses, for the left and the right cameras respectively. Although the state vector only contains the pose of the left camera, the pose of the right camera can be easily obtained using the extrinsic parameters from the calibration.

The dimension is then reduced to $\mathbb{R}^3$ assuming the stereo images are properly rectified. However, by representing the same in $\mathbb{R}^4$, we can skip the rectification, and the camera poses are given by:

The position of the feature in the world frame is calculated using Gaussian-Newton least square minimization. The residual of measurement can be approximated by the following equation:

G. State Augmentation

The global frame feature pose is determined using the camera pose, which causes the uncertainty of p_j in the global frame to be related to the camera states. By projecting the residual in equation (4) onto the null space V of HJ, this correlation is removed.

1) Determine Cam0 pose.
2) Determine Cam1 pose.
3) Identify 3D feature position in the world frame and its observation using stereo cameras.
4) Transform the feature position from the world frame to the cam0 and cam1 frame.
5) Adjust the measurement Jacobian to maintain observability constraint.
6) Calculate the residual.

II. Updating Procedure

The updating process is executed in the following manner:

1) Verify if H and r are empty.
2) Perform decomposition on the final Jacobian matrix to minimize computational complexity.
3) Determine the Kalman gain.
4) Calculate the state error.
5) Update the IMU state.
6) Modify the camera states.
7) Refresh state covariance.
8) Ensure covariance symmetry.

III. Results

The outcomes of our implementation are presented below.

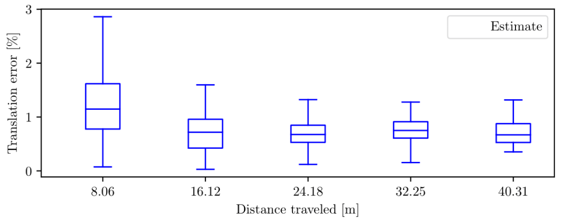


Fig. 2. Classical approach translation error percent across distance

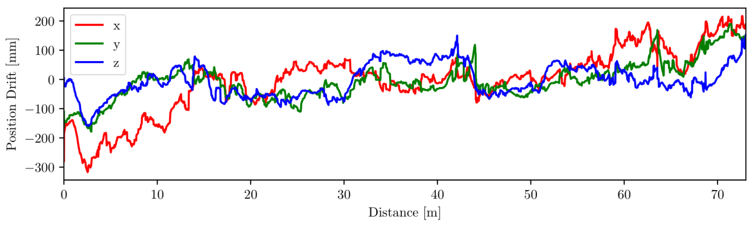
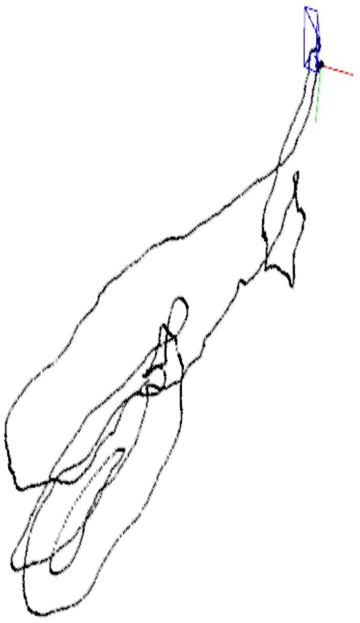


Fig. 3. Classical approach translation error across distance



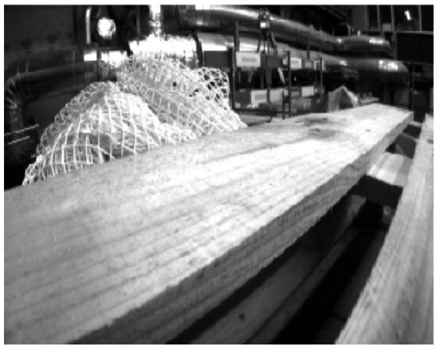


Fig. 4. Output

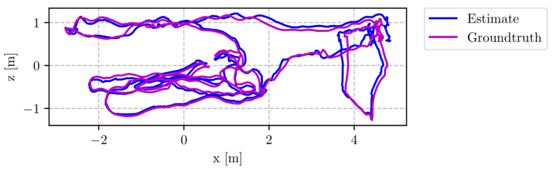


Fig. 5. Classical approach trajectory side view

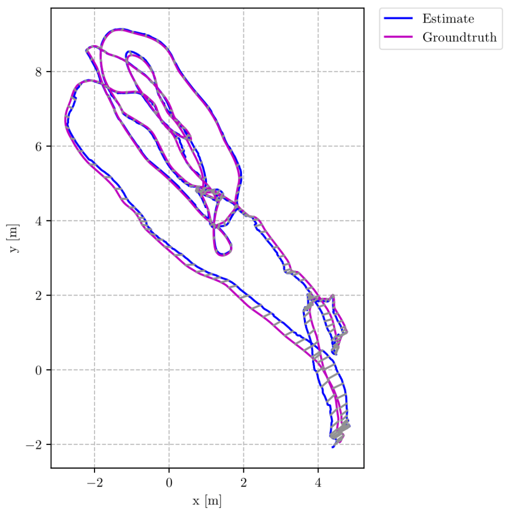


Fig. 6. Classical approach trajectory top view

IV. Phase 2

A. Need of Deep Learning approach

The MSCKF is working brilliantly provided the data from sensors are good and is not drifting over time. However, that is not the case usually overtime the IMU drifts and If the feature is not good for the Camera the output affects badly. In this part, we try to first apply the deep learning approach to the VIO so that we can overcome this drawback

B. Introduction

The approach towards VIO was to start simple and get Visual and Inertial Odometry and then combined those results using simple weighted or another neural network to get results. This can use various approaches and can be varied depending on the size of the dataset and computing power.

C. Dataset

For our implementation, we used EuRoC MAV Dataset's subset, Machine Hall 01,02,03,04,05 The dataset we used is comparatively small and hence the chance of overfitting will be always high in such cases.

We had to convert all the data into relative data, and using transformation we successfully got it.

As these sensors are completely different from each other and have different ways of recording the data they work at different frequencies:
IMU frequency: 200Hz
Camera frame rate: 20 Hz
Ground truth rate:100 Hz
As a result, to overcome the problem of different frequencies we only consider the data of common timestamps across all data.

D. Loss

As we knew the L1 and L2 norms are most commonly used loss and are commonly used In this approach, we combined both the losses together so that depending on the value of β it can be easily scaled

$$L_{\delta,\beta}(y, f(x)) = \begin{cases} \frac{1}{2}(y - f(x))^2 & \text{if } |y - f(x)| \leq \delta \\ \beta\left(|y - f(x)| - \frac{\delta}{2}\right) & \text{otherwise} \end{cases} \quad (8)$$

Here, y is the true label, $f(x)$ is the predicted value, δ is the threshold parameter that determines where the loss function transitions from quadratic to linear, and β is a scaling factor that determines the relative weight of the linear and quadratic portions of the loss function.

E. Visual Odometry

Visual Odometry using Deep learning can be extremely simple and works well with only the CNN network we used multiple CNN layers to estimate odometry and we even get good results for the provided dataset. The model is as below:

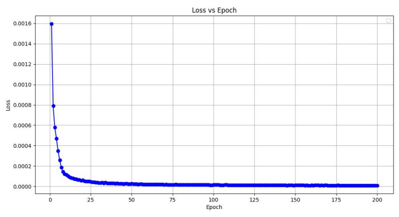
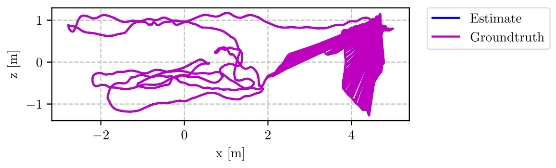
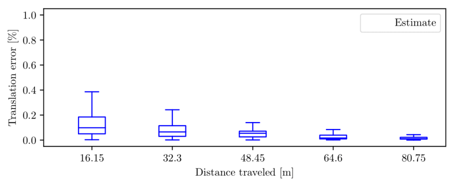


Fig. 7. Loss vs Epoch for VO





F. Inertial Odometry

For Inertial Odometry as we talked we used the same concept of common for data generation and used the data for training the data on Temporal Convolution Network. We started with the LSTM network, however upon the time and

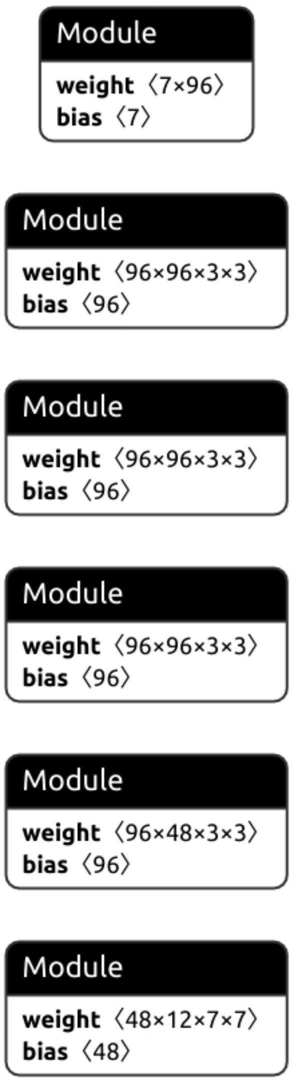
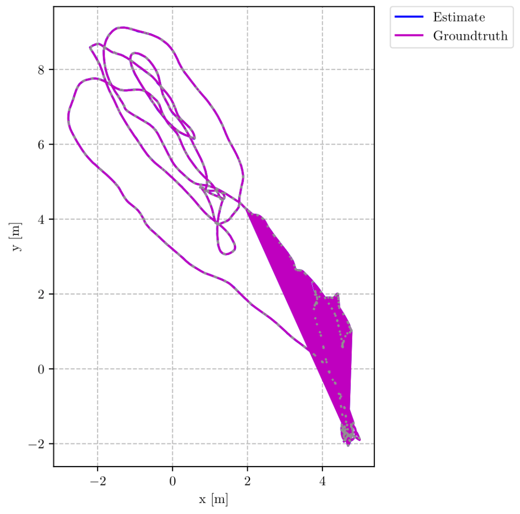
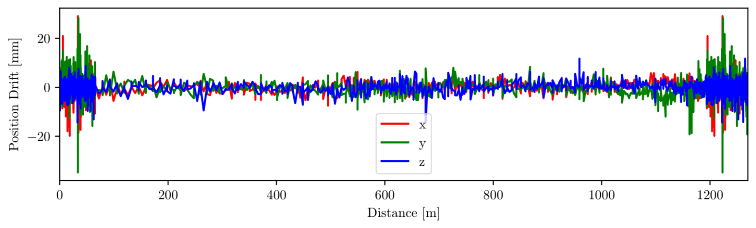


Fig. 10. Network for VO





data it needs to train we switched to Temporal Convolution Networks. The temporal convolution network also worked well for the dataset and in less than 20 epochs get overfit and gave us different results. The models we used are pretty big hence not included in the report. The loss vs epoch graph for the following is shown below:

G. *Visual Inertial Odometry*

Visual Inertial Odometry is simply combining the Visual and inertial network with a linear layer at the end that can be used to get good more optimised results. The network was huge and it took much longer than only Visual or only Inertial network. The models we used are pretty big and hence not included in the report. We thought of two approaches for Visual image Odometry first is simply the weighted average of the function.

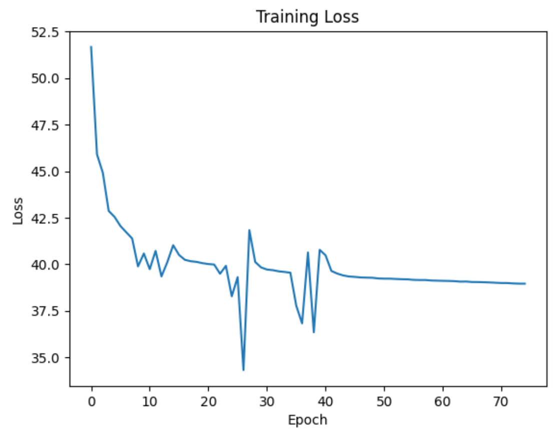


Fig. 13. Loss vs Epoch for VIO

V. Conclusion:

In this Project, we got an overview of sensor fusion and the ground reality of the complexity and drawback it has. we tried to completely implement the classical approach using the Multi-Scaled Kalmann filter in Phase 1, while in Phase 2 we tried to overcome the issue of Phase 1 using different approaches and networks. In sensor fusion getting time-sync data is an essential step as it decides your outputs. The so data interpolation is an essential step that we missed. For our current approach, we reduce our timestamp by taking common which indeed affected the overall training and testing outputs. In the Inertial part, we didn't reduce the IMU biases from the measurement leading to an error in trajectories. We successfully implemented all three networks and losses were decreasing. We got considerably good trajectories for our outputs for the trained network We realised that for such a network we need varied data-set as the network easily overfits in no time. The loss convergence doesn't guarantee the better output

References

[1] RBE 549: Robot Perception (2023). Project 4: Visual Inertial Odometry. Retrieved from https://rbe549.github.io/spring2023/proj/p4/

[2] Sun, K., Mohta, K., Pfrommer, B., Watterson, M., Liu, S., Mulgaonkar, Y., Taylor, C. J., Kumar, V. (YEAR). Robust Stereo Visual Inertial Odometry for Fast Autonomous Flight.

[3] Mourikis, A. I., Roumeliotis, S. I. (YEAR). A Multi-State Constraint Kalman Filter for Vision-aided Inertial Navigation.

[4] Caballero, J., Ledig, C., Aitken, A., Acosta, A., Totz, J., Wang, Z., Shi, W. (2017). Real-Time Video Super-Resolution with Spatio-Temporal Networks and Motion Compensation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (pp. 0-0).

[5] Zhan, H., Weerasekera, C. S., Bian, J. -W., Reid, I. (2020). Visual Odometry Revisited: What Should Be Learnt? In 2020 IEEE International Conference on Robotics and Automation (ICRA) (pp. 4203-4210). doi:10.1109/ICRA40945.2020.9197374

[6] Cioffi, G., Bauersfeld, L., Kaufmann, E., Scaramuzza, D. (2023). Learned Inertial Odometry for Autonomous Drone Racing. In IEEE Robotics and Automation Letters (RA-L).

[7] Clark, R., Wang, S., Wen, H., Markham, A., Trigoni, N. (YEAR). VINet: Visual-Inertial Odometry as a Sequence-to-Sequence Learning Problem. In Department of Computer Science, University of Oxford, United Kingdom; Department of Computer Science, University of Warwick, United Kingdom.

[8] Kendall, A., Grimes, M., Cipolla, R. (YEAR). PoseNet: A Convolutional Network for Real-Time 6-DOF Camera Relocalization.