Deep-UnDeep Visual Inertial Odometry: Phase 2

Rutwik Kulkarni
Department of Robotics Engineering
Worcester Polytechnic Institute
Email: rkulkarni1@wpi.edu

Ankit Mittal
Department of Robotics Engineering
Worcester Polytechnic Institute
Email: amittal@wpi.edu

***Abstract*— This work presents a deep learning-based approach to compute visual, inertial and visual-inertial odometry for a Micro Aerial Vehicle (MAV) using a custom and synthetic dataset generated from simulation tools. We explore architectures that combine convolutional neural networks (CNN) and long short-term memory (LSTM) networks to learn spatial and temporal features from the visual and inertial data. Convolutional and LSTM networks are trained independently on the visual and inertial data, and their outputs are fused to estimate the MAV's pose. The effectiveness of our approach is evaluated on the custom dataset, demonstrating the potential of deep learning techniques in enhancing visual inertial odometry for MAVs under simulated conditions.**

I. Introduction

Visual-Inertial Odometry (VIO) is crucial for autonomous navigation in GPS-denied environments, particularly for Micro Aerial Vehicles (MAVs). VIO combines visual data and inertial measurements to estimate device trajectories. Traditionally, VIO relies heavily on precise feature detection and tracking [1] [2], a process that can falter in dynamic or complex settings. While recent deep learning advancements like [3] and [4] have improved feature-based visual odometry, integrating these techniques with inertial data remains a challenge due to the abstract nature of inertial measurements.

This paper introduces a novel deep learning framework for VIO tailored specifically for MAVs. Our framework develops a unique architecture and loss function designed to effectively fuse visual and inertial data. The approach aims to predict the relative pose between two camera frames by utilizing both image frames and interceding inertial measurements. By adapting and innovating upon existing deep learning methods, this research seeks to enhance the precision and reliability of VIO systems for MAVs, contributing significantly to the advancement of autonomous navigation technologies.

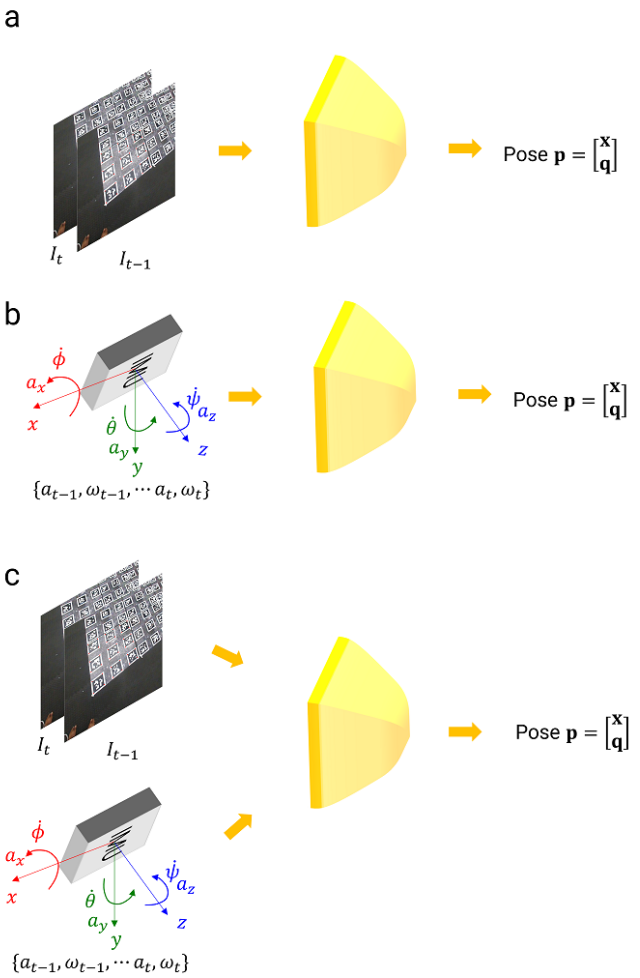

Fig. 1: Problem Statement

II. Problem Statement and Methodology

Figure 1 illustrates the predictive outputs of our networks, detailing how incremental positions and orientations are derived. After acquiring these increments, we employ dead reckoning to assess the trajectory tracking performance of our system. This process provides a comprehensive evaluation of how accurately the MAV follows its intended path over time.

III. Dataset Generation

Generating a reliable dataset is important for training deep learning models to estimate odometry from visual and inertial data in Micro Aerial Vehicles (MAVs). Due to the limited availability and complexity of existing datasets, we chose to create our own synthetic dataset using Blender for better control and clearer evaluation.

A. Simulation Environment Setup

We used Blender to simulate scenarios with an aerial robot that has a down-facing camera and a 6-DoF IMU (3-axis accelerometer and 3-axis gyroscope). The robot moves over a flat surface, capturing RGB images and IMU data. The environment is a large, flat area with textured images from the internet, providing plenty of data for feature tracking.

This setup replicates the conditions an MAV might encounter but does not include elements like motion blur, depth of field, or changing lighting, to keep the simulation simple and focused on basic dynamics.

B. IMU Data Generation

To generate realistic IMU data, we used Matlab's IMU Model. This third-party software simulates the output of an IMU sensor, including noise and other realistic properties. The simulator records angular rates and linear accelerations at a frequency of 1000 Hz. These raw data points are then processed to mimic how actual IMU sensors would capture movements, ensuring that our model trains on data that closely resembles real-world sensor output.

C. Camera Setup and Image Acquisition

We set the camera's intrinsic matrix (K matrix) to a fixed value to make the calibration process simpler, and we did not simulate lens distortion. Images were captured at a rate of 100 Hz using a down-facing camera. This setup aligns the visual data with the higher-frequency IMU data, although at a slower rate, replicating the common differences in data capture speeds between cameras and IMUs in real Micro Aerial Vehicles (MAVs).

D. Motion Simulation and Data Extraction

We used a physics simulator from the "Hands-On Aerial Robotics" course to simulate drone movements. These movements were based on the control equations used by the PX4 controller for quadrotors. We created the drone's motion profile using a custom-made random motion profile generator and tested the model on two different flight paths: a square and a spiral. We recorded the drone's positions, velocities, and orientations in the global frame (NED) and converted these to the local sensor frame to get the incremental poses, velocities, and accelerations needed for training our model.

E. Data Synchronization and Transformation

To align the visual and inertial datasets, we addressed the differing data acquisition rates: the camera captures images at 100 Hz, while the IMU collects data at 1000 Hz. We downsampled the IMU readings to 100 Hz to synchronize the datasets, ensuring accurate data correlation for reliable odometry estimation in our VIO network. This synchronization enhances the consistency of our training and validation phases.

By creating and using a custom synthetic dataset, we make sure that the model is trained in controlled, yet realistic conditions. This allows for a detailed evaluation of its performance in simulated MAV navigation tasks.

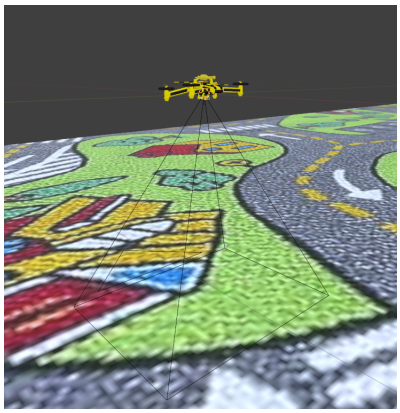

Fig. 2: Simulation Environment Setup

Fig. 3: Image at t

Fig. 4: Image at t+1

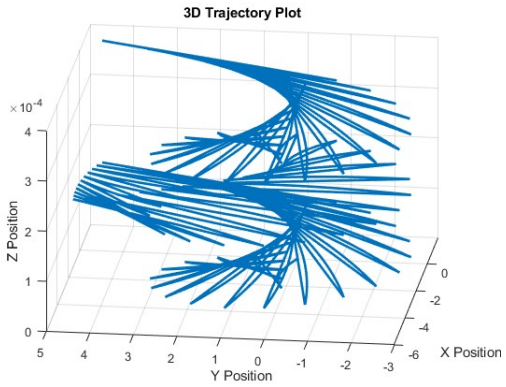

Fig. 5: Training Trajectory for Train Dataset

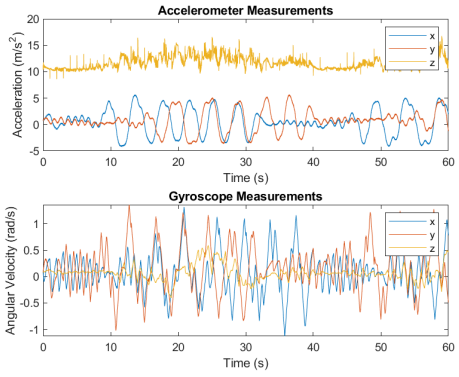

Fig. 6: Noisy IMU Readings mimicking real IMU readings

F. Visual Odometry Network

1) Network Architecture: We designed our network based on the paper [5] . Refer image no 7 in which it begins with two sequential video frames marked as T and T+1, capturing distinct moments. These frames are merged side by side, resulting in a 6-channel image since each original frame has three channels (assumed to be RGB). This composite image is then sequentially fed through a convolutional neural network (CNN), with each layer specifically designed to gradually compress the spatial dimensions while expanding the depth, indicated by the number of filters in each layer—scaling down from an initial 640x480x3 to a final feature map of 10x8x1024. The CNN's output is then processed by a Long Short-Term Memory (LSTM) network comprised of two series of cells with 1000 units each, showcasing its capability to capture the complex temporal patterns between the frames. This integration of spatial and temporal processing is central to the model's function of estimating the movement and orientation between the two captured frames. Culminating the process, the model outputs a 6x1 vector that represents the final relative pose, consisting of Euler angles for orientation and a 3-dimensional vector for position, detailing the motion from frame T to T+1.

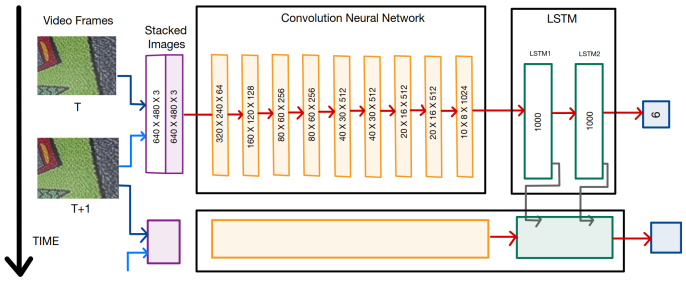

Fig. 7: Deep VO Architecture

Sr. No.	Hyperparameter	Value
1	Epochs	50
2	Learning Rate	1e-4
3	Batch Size	8
4	Optimizer	Adagrad
4	Sequence Length	5

TABLE I: Deep VO hyper-parameter

2) Training and Validation: For training and testing purposes, we produced a total of 8 video sequences that featured various backgrounds and trajectories. Out of these, 2 sequences were set aside exclusively for testing. The network was trained using sequences of 5 consecutive frames, with a batch size of 8. We set the learning rate to 0.0001 and employed the Adagrad optimizer for the training process.

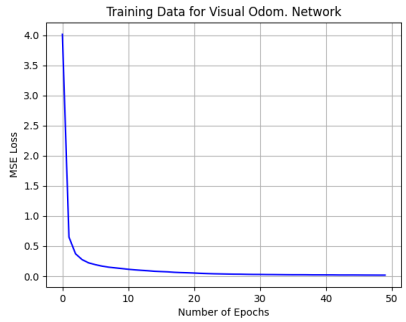

Fig. 8: Training Loss for VO

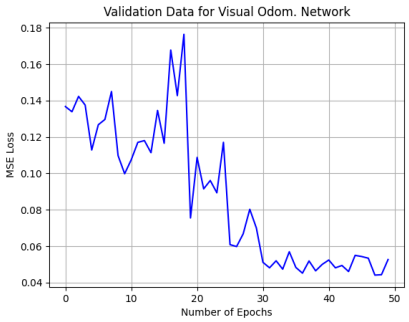

Fig. 9: Validation Loss for VO

G. Inertial Odometry Network

1) Network Architecture: For IO based deep learning network we utilized a simple LSTM-based network. The LSTM's ability to capture sequential dependencies makes it well-suited for the odometry estimation problem, which requires understanding the temporal motion model.

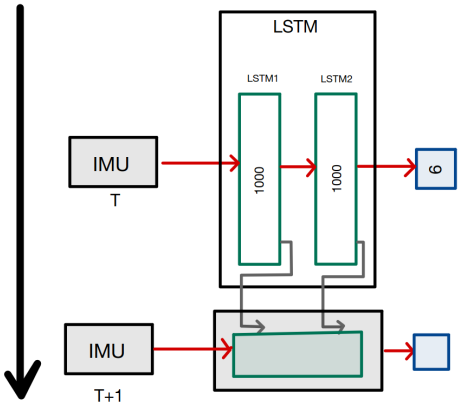

Fig. 10: Deep IO Architecture

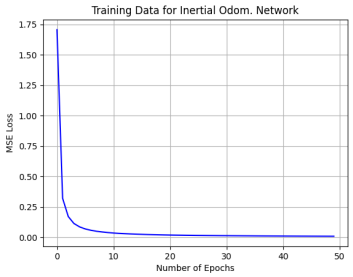

Fig. 11: Training Loss for IO

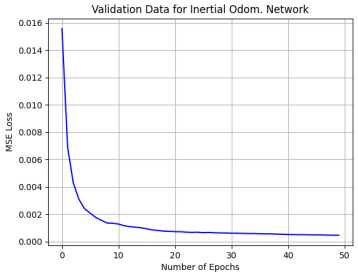

Fig. 12: Validation Loss for IO

Sr. No.	Hyperparameter	Value
1	Epochs	50
2	Learning Rate	1e-4
3	Batch Size	8
4	Optimizer	Adagrad

TABLE II: Deep IO hyper-parameter

2) Training and Validation: We generated data at 1000 Hz across 8 different video sequences, each comprising 8000 frames, to train the Deep IO network. As with the Deep VO, 2 of these sequences were held back for testing. For details on the hyperparameters used during training, Refer Table II.

H. Visual-Inertial Odometry Network

1) Network Architecture: Regarding Deep VIO architecture use utilized same architecture as Deep VO. But to intregate the imu values in the network wr concated the IMU data with the CNN output before feeding it into LSTM. We utilized same CNN parameter for DEEP VIO as used in deep vo. For detail refer 13

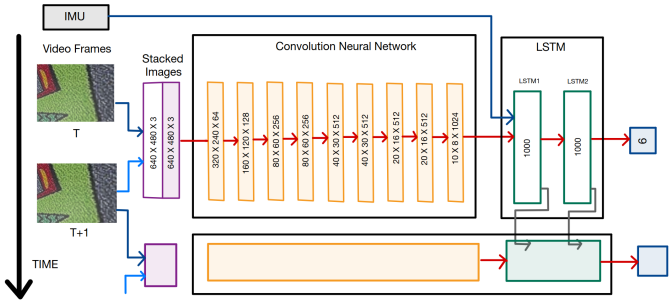

Fig. 13: Deep IO Architecture

2) Training and Validation: For training purposes, we employed the same dataset as used for the Deep VIO and Deep IO models. To synchronize the datasets, we matched the camera's frame rate of 100 Hz with the IMU's 1000 Hz by integrating the IMU data over the last 10 frames before inputting it into the network.

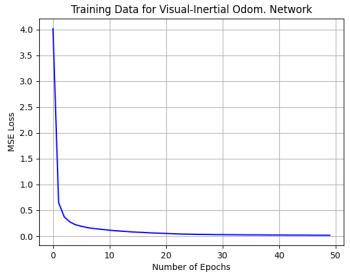

Fig. 14: Training Loss for VIO

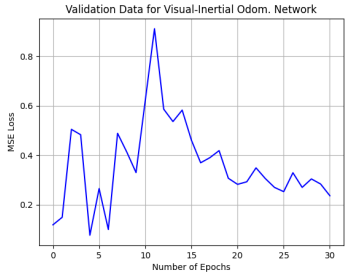

Fig. 15: Validation Loss for VIO

Sr. No.	Hyperparameter	Value
1	Epochs	10
2	Learning Rate	1e-4
3	Batch Size	8
4	Optimizer	Adagrad

TABLE III: Hyperparameter Settings for Unsupervised Homography Net

I. Loss Function

The loss function employed in our model is a combination of the Mean Square Error (MSE) for both positions, denoted as $\mathbf{p}$, and orientations, denoted as ϕ. It is defined as follows:

$$\theta^* = \arg\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} \sum_{k=1}^{t} \left(\|\hat{\mathbf{p}}_k - \mathbf{p}_k\|_2^2 + \kappa \|\hat{\phi}_k - \phi_k\|_2^2 \right) \tag{1}$$

Here, $\|\cdot\|_2$ represents the Euclidean norm, κ is a scaling factor, set to 100 in our experiments, to balance the contributions of position and orientation errors in the loss function, and N is the total number of samples in the dataset. In our model, the orientation ϕ is represented using Euler angles, providing a more interpretable framework as opposed to quaternions.

IV. Results

A. Trajectory Tracking Visual Odometry Network only

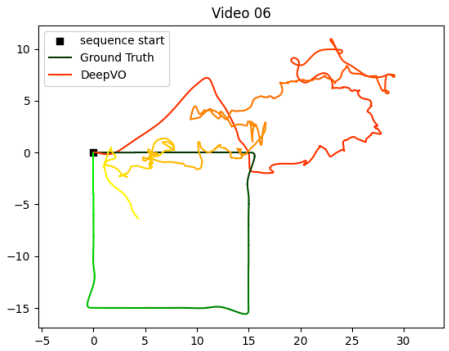

Fig. 16: Trajectory tracking for VO Network

The Absolute Trajectory Error (ATE) after dead-reckoning from the Visual Odometry Network is given in the table.

TABLE IV: Absolute Trajectory Error Statistics

Parameter	RMSE	Std Dev
Rotation	6488.96	4.597
Translation	79.5	1.030

B. Trajectory Tracking Inertial Odometry Network only

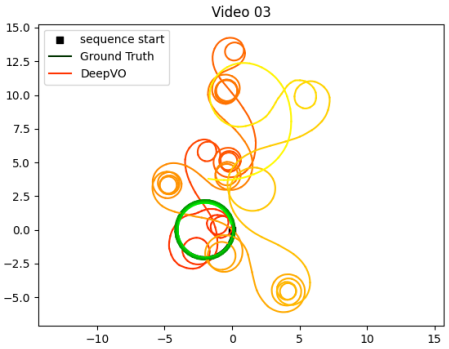

Fig. 17: Spiral Trajectory tracking for IO Network

The Absolute Trajectory Error (ATE) after dead-reckoning from the Inertial Odometry Network is given in the table.

TABLE V: Absolute Trajectory Error Statistics

Parameter	RMSE	Std Dev
Rotation	13889.96	6.597
Translation	102.06	2.030

C. Trajectory Tracking Visual-Inertial Odometry Network

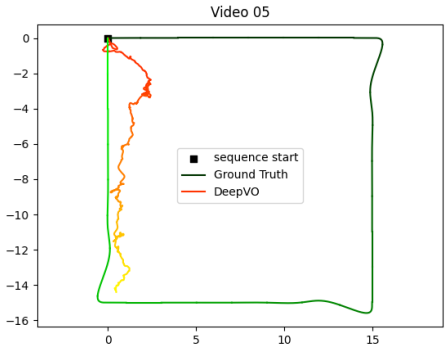

Fig. 18: Trajectory tracking for VIO Network

The Absolute Trajectory Error (ATE) after dead-reckoning from the Visual-Inertial Odometry Network is given in the table.

TABLE VI: Absolute Trajectory Error Statistics

Parameter	RMSE	Std Dev
Rotation	6768.96	4.107
Translation	35.06	0.930

V. Discussions

A. Insights into the problem with our approach

During training, we observed a decrease in both training and validation losses(especially in VO), yet the results were not accurate and contained significant errors initially. This may be attributed to the position and orientation values being very close to each other from one frame to the next, with a precision of approximately 0.001. We suspect that the model struggled to learn and predict values with such high precision. Additionally, due to the use of LSTM, which carries forward information from previous frames, a few inaccurate predictions early in the sequence could adversely affect future predictions. As a result, once the network started producing incorrect outputs, the deviations in predicted values began to accumulate, leading to further errors in subsequent frames.

B. VO v/s IO v/s VIO

Our results indicate that each odometry method has its strengths and weaknesses depending on the specific navigation context. Visual Odometry (VO) performs well in visually rich environments but struggles in texture-less regions or under rapid movements. Conversely, Inertial Odometry (IO) offers consistent performance regardless of visual features but is prone to drift over time. The combined Visual-Inertial Odometry (VIO) approach leverages the strengths of both to significantly reduce drift and improve trajectory estimation in diverse environments. While our results might not be precise in every scenario, careful observation of the trajectories reveals subtleties in the data that highlight inherent challenges

in visual-inertial navigation. These observations suggest that the problem of integrating visual and inertial data effectively under varying dynamic conditions remains complex, and our method provides a framework that begins to address these intricacies. Further refinement and testing are required to enhance the robustness and accuracy of the system across all types of environments.

C. Potential Research Avenues

1) Combining Traditional methods with Deep Learning Methods: Combining Bayesian filters and deep learning offers a promising approach to enhance visual-inertial odometry. This fusion aims to leverage deep learning's feature extraction capabilities and Bayesian filters' ability to handle uncertainties. Challenges include efficient integration for real-time processing while improving accuracy in dynamic environments. This research could advance autonomous navigation systems by providing more reliable odometry estimates.

2) Using Event cameras: Exploring the integration of event cameras in visual-inertial odometry offers a promising avenue for improving system robustness and accuracy. Event cameras provide high temporal resolution and low latency, making them suitable for dynamic environments with rapid motion and lighting changes. This research has significant potential to enhance the performance of odometry systems in real-world applications.

VI. Acknowledgment

We express our sincere gratitude to Prof. Nitin Sanket, Teaching Assistant Yijia Wu, and Grader Aabha Tamhankar for their constant support and technical guidance.

References

[1] A. I. Mourikis and S. I. Roumeliotis, "A multi-state constraint kalman filter for vision-aided inertial navigation," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*, 2007, pp. 3565–3572. 1

[2] K. Sun, K. Mohta, B. Pfrommer, M. Watterson, S. Liu, Y. Mulgaonkar, C. J. Taylor, and V. R. Kumar, "Robust stereo visual inertial odometry for fast autonomous flight," *IEEE Robotics and Automation Letters*, vol. 3, pp. 965–972, 2017. [Online]. Available: https://api.semanticscholar.org/CorpusID:3725704 1

[3] D. DeTone, T. Malisiewicz, and A. Rabinovich, "Superpoint: Self-supervised interest point detection and description," *CoRR*, vol. abs/1712.07629, 2017. [Online]. Available: http://arxiv.org/abs/1712.07629 1

[4] P. Sarlin, D. DeTone, T. Malisiewicz, and A. Rabinovich, "Superglue: Learning feature matching with graph neural networks," *CoRR*, vol. abs/1911.11763, 2019. [Online]. Available: http://arxiv.org/abs/1911.11763 1

[5] S. Wang, R. Clark, H. Wen, and N. Trigoni, "Deepvo: Towards end-to-end visual odometry with deep recurrent convolutional neural networks," *CoRR*, vol. abs/1709.08429, 2017. [Online]. Available: http://arxiv.org/abs/1709.08429 3