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

Shambhuraj Mane
MS Robotics Engineering
samane@wpi.edu

Swati Shirke
MS Robotics Engineering
svshirke@wpi.edu

Jesulona Akinyele
MS Robotics Engineering
jfakinyele@wpi.edu

***Abstract*—This paper addresses the challenges of Visual-Inertial Odometry (VIO) through deep learning techniques. We present a comprehensive study on integrating camera and inertial measurement unit (IMU) data for robust motion estimation. We explore three distinct phases: Visual Odometry (VO), Inertial Odometry (IO), and a fusion of both in Visual-Inertial Odometry (VIO). Each phase is approached with end-to-end deep learning models designed to enhance the system's ability to estimate poses in complex environments. Our research delves into innovative neural network architectures and learning strategies to effectively handle the spatial-temporal nature of VIO data. This includes experimenting with recurrent neural networks, and introducing methods to improve data efficiency through unsupervised and semi-supervised techniques.**

Index Terms* — *Blender, Synthetic Data Generation, Visual Inertial Odometry, Visual Odometry, Inertial Odometry

I. Introduction

Odometry, the process of estimating motion through sensory input, plays a pivotal role in robotics, navigation systems, and various autonomous applications by providing real-time estimates of a device's position and orientation. By continuously tracking motion through sensor input, odometry allows systems to update their internal representation of the environment, make informed decisions, and adapt to changing conditions. Moreover, accurate odometry is essential for tasks such as path planning, obstacle avoidance, and object manipulation, where precise knowledge of the device's position and motion is paramount.

To address this requirement many techniques has been studied by the researchers. From which Visual Odometry (VO), Inertial Odometry (IO) and Visual Inertial Odometry (VIO) are extensively explored. VIO estimates the agent's self-motion using information collected from cameras and inertial measurement unit (IMU) sensors and leverage the complementary strengths of each sensor modality. Visual sensors offer detailed scene perception and feature-rich environment representation, enabling accurate localization and mapping. Meanwhile, IMU sensors provide precise motion dynamics measurements, facilitating motion prediction and correction.

In this paper, We implemented all three techniques and compared the results with respect to ground truth. Since all the three techniques are achieved in an end-to-end manner, it does not need any module in the classic VIO pipeline (even camera calibration). The main contribution is :

- We demonstrate that the odometry problem can be addressed in an end-to-end fashion based on DL, in all three techniques VO, IO and VIO.
- We developed a custom synthetic data using quadrotor simulation in Blender.

II. Related Work

Early work on the VO, IO and VIO is reviewed in this section, discussing various algorithms and their differences from others. There are mainly three types of algorithms in terms of the technique and framework adopted: Classical Methods, Data-Driven Methods and Temporal Adaptive Inference.

A. Classical Methods

Early approaches to VIO primarily focused on sensor fusion techniques, such as Extended Kalman Filters (EKF) and Unscented Kalman Filters (UKF), to integrate visual and inertial measurements for motion estimation. Notable works include the seminal work by Mourikis and Roumeliotis [1], which introduced a tightly-coupled VIO framework using an EKF for state estimation. Subsequent research efforts extended classical VIO methods to address challenges such as sensor calibration, feature tracking, and drift mitigation [2].

B. Data-Driven Methods

In recent years, there has been a growing interest in leveraging data-driven approaches for VIO, driven by the success of deep learning in various computer vision tasks. Researchers have explored convolutional neural networks (CNNs), recurrent neural networks (RNNs), and hybrid architectures to learn motion patterns directly from sensor data. For instance, Clark et al. [3] proposed a deep learning-based VIO system trained on large-scale datasets, achieving superior performance in challenging environments. Similarly, Zhu et al. [4] introduced a novel VIO framework using recurrent neural networks for temporal adaptive inference, enabling robust motion estimation in dynamic scenes.

C. Temporal Adaptive Inference

Temporal adaptive inference has emerged as a key concept in data-driven VIO methods, enabling dynamic adjustment of motion estimates based on changing environmental conditions and sensor characteristics. LSTM networks, known for their ability to model sequential data and long-term dependencies, play a central role in enabling temporal adaptive inference in VIO systems. By incorporating feedback mechanisms and adaptive learning algorithms, temporal adaptive inference enhances the robustness and accuracy of VIO systems, particularly in scenarios with varying motion dynamics or sensor noise levels.

III. Organization of the paper

Further, the paper is structured into methodology, results and discussion, conclusion and future work. Section IV discusses the methodology in detail. It consists of synthetic data generation, network architecture, training process and loss functions. Further, section V discusses the results, section VI discusses the conclusion and section VII discusses future work.

IV. Methodology

Objective of this project is to train neural network models to perform pose estimation using camera and or IMU data. Here, we are training 3 different networks as follows:

- Visual-Inertial Odometry: In this phase, we have designed and trained a neural network to perform pose estimation using both camera and IMU data. The beauty of an Inertial Measurement Unit (IMU) lies in its effectiveness with fast movements and sudden changes (jerks), where traditional cameras may struggle, yet it tends to experience drift over time, a challenge where cameras excel. This complementary relationship sets the stage for an intriguing multi-modal fusion problem, allowing for the accurate estimation of the camera's pose to enhance depth tracking.
- Visual Odometry: In this phase, we have designed and trained a neural network to estimate pose using only images.
- Inertial Odometry: ln this phase, we have designed and trained a neural network perform pose estimation using IMU data only.

The end to end pipeline subsumes following stages.

A. Synthetic data generation

In this phase, we have generated synthetic data using quadrotor simulation in Blender software. To perform 6-DOF quadrotor simulation, we used this framework[1]. This framework uses linearized model of a quadrotor.

Initially, we created waypoints for different trajectories using parametric equations of various shapes such circle, oval, figure eight shape. We input these waypoints to the quadrotor simulation to generate quadrotor state data for a given trajectory. Quadrotor state is $z = [x, \alpha, v, \omega] \in R^{12}$ where $x, \alpha, v,$ andω are the position, orientation, linear velocity, and angular velocity vectors respectively. Position and orientation information together forms ground truth of the training data.

To generate visual data, we chose an image with lot of feature points and spawned it in blender as plane on ground. Also, in the Blender framework, a camera is attached to the quadrotor facing towards ground. We captured images in Blender at a frequency of 100Hz.

To generate IMU data, we extracted linear and angular velocity data from the quadrotor state. We are generating data for 6DOF IMU, which consist of linear acceleration and angular velocity. Further, we calculated linear acceleration from linear velocity of quadrotor state. In order to make data more realistic, we added Gaussian noise to data using IMU simulation model of MATLAB[2]. Frequency of quadrotor simulation and IMU data is 1000Hz. Generated data set consist of pose (ground truth), IMU data and Visual data, which is used for training neural network for pose estimation.

We created total 6 trajectories using parametric equations which includes circle, infinity sign, ellipse, spiral, cycloid, sine wave. Figure 1 and Figure 2 show example visualizations of one of our paths. For each shape, we created 2 trajectories by varying parameters. For each trajectory we generated 500 images, 500 pose readings and 5000 IMU readings. Overall, we created 6000 images, 6000 pose readings and 60000 IMU readings. Further, we split this data for training and evaluation in a 70-30 ratio respectively.

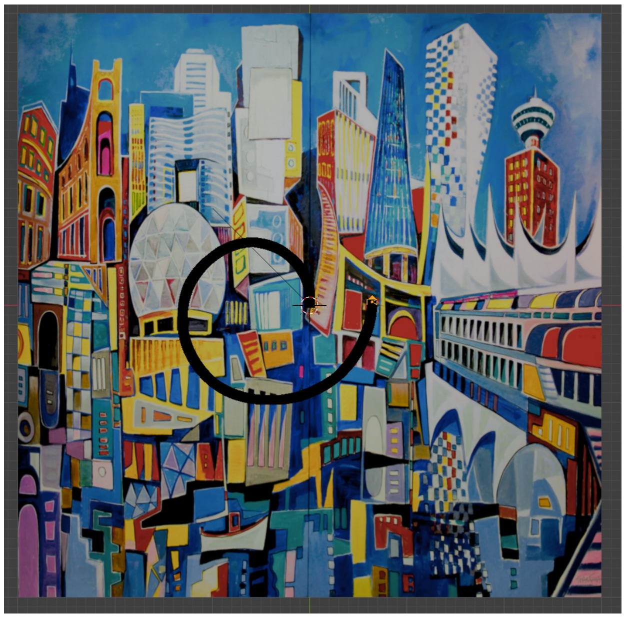

Fig. 1. Example Blender Image Path: Spiral Trajectory

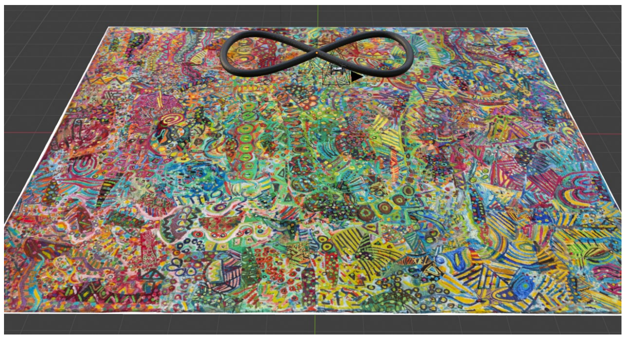

Fig. 2. Example Blender Image Path: Infinity Sign Trajectory

B. Network Architecture

We have created 3 different networks for Visual, Inertial and Visual-Inertial odometry as mentioned below.

- **Visual Odomerty**:
  The Visual Encoder network comprises a series of convolutional layers followed by an LSTM layer and a linear layer for visual odometry tasks. The initial convolutional layers progressively extract hierarchical features from the input images, starting with 32 channels and gradually increasing to 512 channels. Each convolutional layer is paired with batch normalization, ReLU activation, and max-pooling to enhance feature representation and reduce spatial dimensions. Subsequently, the LSTM layer processes the sequential feature maps generated by the convolutional layers, capturing temporal dependencies in the visual data. Finally, the linear layer maps the LSTM output to a six-dimensional vector representing the predicted pose. This architecture enables the network to effectively learn spatial and temporal patterns from visual input data, facilitating accurate localization and mapping in visual odometry applications.

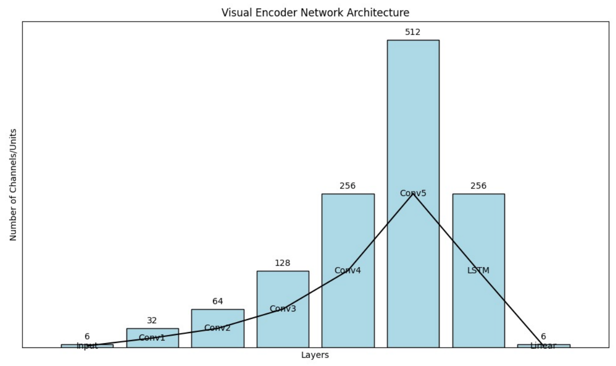

Fig. 3. VO Network Architecture

- **Inertial Odometry**:
  In this network, we are using three convolutional layers with kernel sizes (5, 5, 3) and output channels (32, 64, 128) respectively. Each convolutional layer is followed by batch normalization, ReLU activation, max-pooling (kernel size 2), and dropout (p=0.1). Further, an LSTM layer with two LSTM cells and input size 128, hidden size 64 is added. We applied dropout with p=0.1 to inputs and recurrent connections. In the linear layer, input size is 64 and output size is 32, followed by ReLU activation and dropout (p=0.1). Another linear layer is added with input size 32 and output size 6 for the final pose prediction.

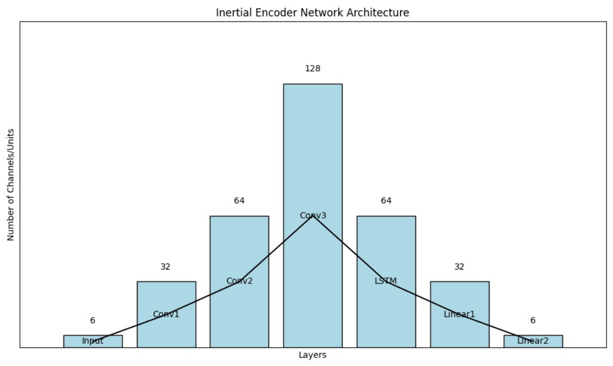

Fig. 4. IO Network Architecture

- **Visual Inertial Odometry**:
  The Visual-Inertial Encoder network comprises two main branches: the visual encoder and the inertial encoder followed by a fusion layer. The visual encoder consists of six layers: the input layer with six channels, followed by four convolutional layers with 64, 128, 256, and 512 channels respectively, a linear layer with 256 channels, and a final linear layer with six channels. Similarly, the inertial encoder consists of seven layers: the input layer with six channels, followed by three convolutional layers with 32, 64, and 128 channels respectively, an LSTM layer with 64 hidden units, and two linear layers with 32 and six channels respectively. The fusion layer then combines the output of both encoders, resulting in a layer with 64 channels.

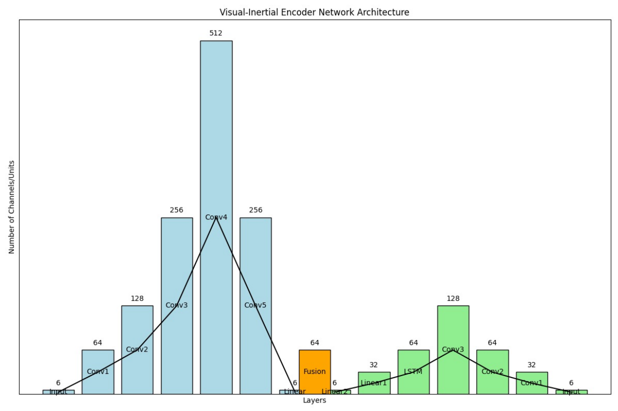

Fig. 5. VIO Network Architecture

C. Training Process

We used the network architecture mentioned above and trained it with our data set for 500 epochs. Also, we used a pre-trained flownet model from this repository[3]. Further, we terminate training if validation loss falls below 0.1. Initially, we kept a higher learning rate which is 0.0001 and further we decreased it to 0.000001 after 200 epochs. An initial higher learning rate ensures faster convergence and further lowering it avoids the overshoot problem. We trained the following 3 types of networks.

- Visual-Inertial Odometry: In this case, we feed images, pose data and IMU data to the network for pose estimation. In this case, we trained 2 networks with Adam and AdamW optimizers.
- Visual Odometry: In this case, we feed only image data to the network. Also, we trained 2 network models with Adam and AdamW optimizer.
- Inertial Odometry: In this case, we provided the network with only IMU data as input for pose estimation and trained the network 2 optimizer settings Adam and AdamW same as above 2 cases.

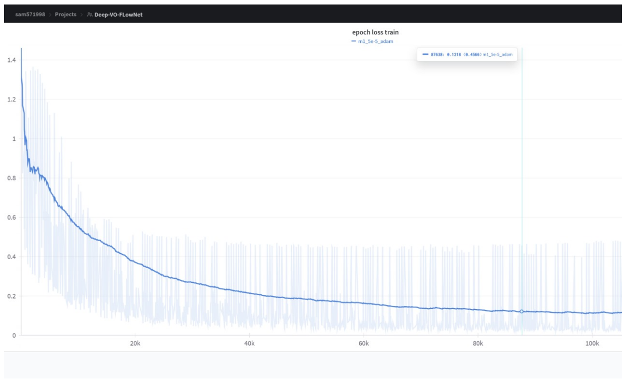

Fig. 6. VO Flownet Training Loss

VO Flownet Training Loss: The depicted training loss curves for Visual Odometry with FlowNet show that over several itterations the loss converges to approximately 0.1218

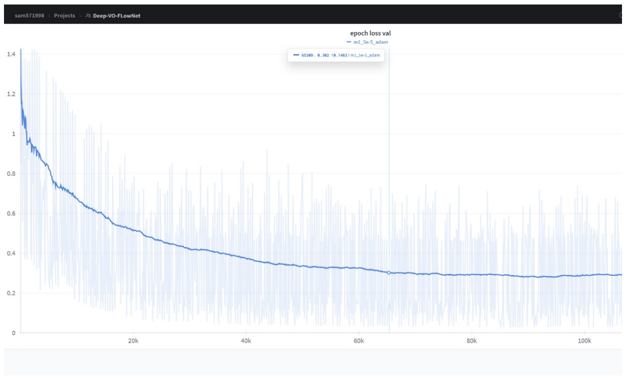

Fig. 7. VO Flownet Validation Loss

VO Flownet Validation Loss: The validation loss curves for Visual Odometry with FlowNet show a similar trend to the training and converges at approximately 0.302.

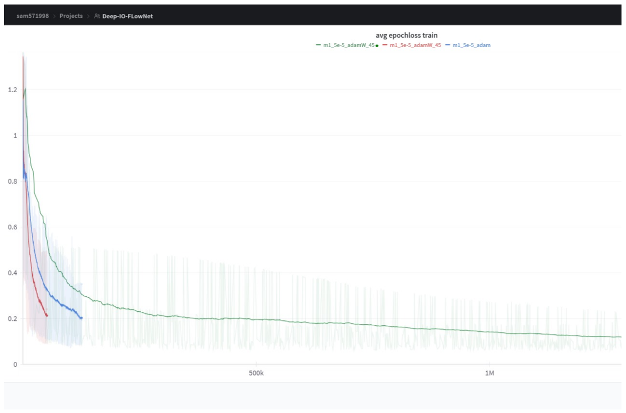

Fig. 8. IO Flownet Training Loss

IO FLownet Training Loss: The training loss graph for Inertial Odometry underscores a significant disparity in the loss descent between the two optimizers, with ”adam45” reducing loss more effectively than the standard Adam. This points to the efficacy of ”adam45” in capturing and learning from the dynamics of IMU data.

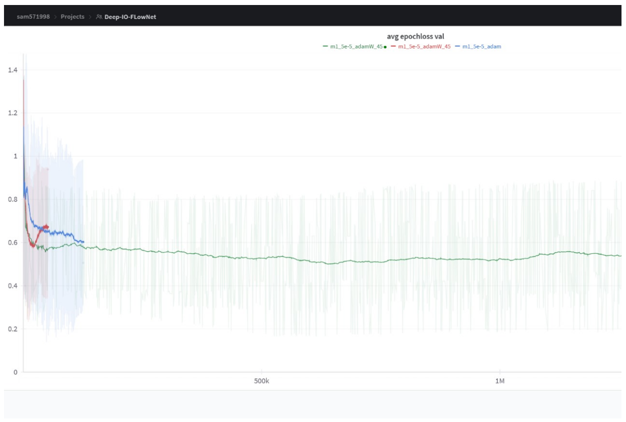

Fig. 9. IO Flownet Validation Loss

IO FLownet Validation Loss: The validation loss graph for Inertial Odometry shows a similar trend to that of the training loss, with ”adam45” converging faster than the standard ”adam”.

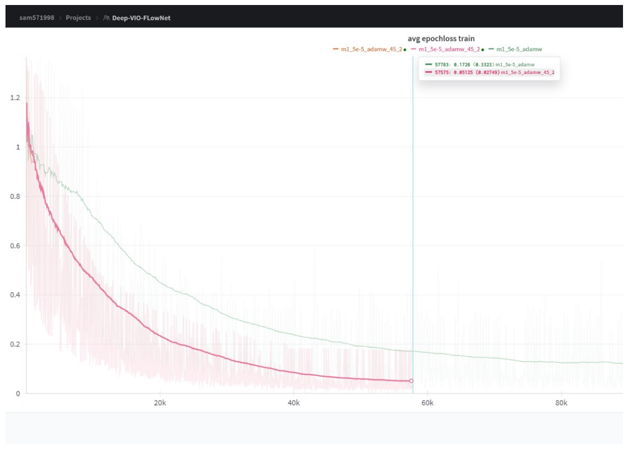

Fig. 10. VIO Flownet Training Loss

VIO FLownet Training Loss: The standard Adam optimizer exhibits a classic convergence pattern, yet the alternative ”adam45” variant achieves a noticeably sharper decline in loss, suggesting more efficient learning dynamics.

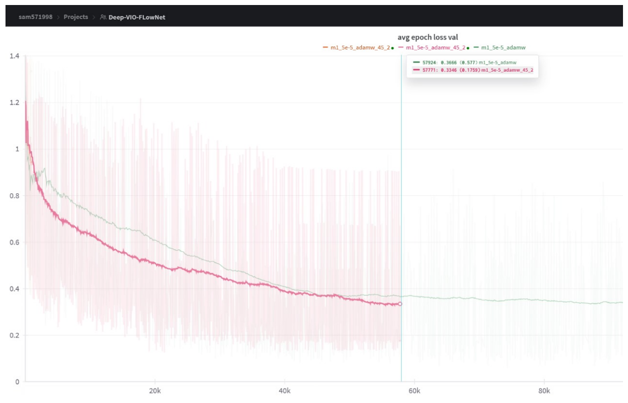

Fig. 11. VIO Flownet Validation Loss

VIO FLownet Validation Loss: Although both curves converge towards similar values, the path taken by the ”adam45” optimizer suggests a potential for a more reliable performance on unseen data.

D. Loss Function

- Position Loss: We use the Mean Squared Error (MSE) loss for position estimation. Taking the square root of the MSE ensures that the loss is in the same scale as the actual position values.
- Orientation Loss: We use the Cosine Embedding Loss for orientation estimation. Before computing the loss, we normalize both the predicted and target quaternions. The Cosine Embedding Loss is suitable for measuring angular differences between normalized quaternions.

These loss functions encourage accurate estimation of both position and orientation, taking into account the geometric nature of the problem and the importance of each component in the overall VIO performance.

Weighted Combination: We combine the position and orientation losses using weighted coefficients (0.4 for position and 0.6 for orientation, as an example). Adjusting these coefficients allows us to prioritize one aspect of the estimation over the other, depending on the specific requirements of the VIO task.

V. Results

To visualize and analyse the results, we used the RPG toolbox from this repository [4]. We tested trained networks on new trajectories to perform pose estimation. Figure 12 to Figure 23 show testing loss for IO, VO and VIO models. After visualizing results in the RPG toolbox, we found that position estimation gives satisfactory results for VIO, VO and IO networks, however for orientation, results are not as expected for all 3 modules. We can infer that the prediction of the orientation is getting fitted perpendicular to the ground truth because of cosine embedding loss. The negation of cosine embedding loss was not considered correctly while training. Figures 18 to 23 show output visualized in the RPG toolbox for VIO, VO and IO networks.

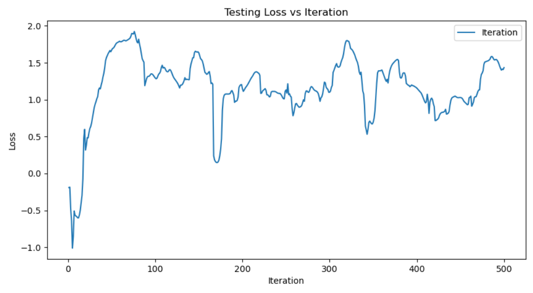

Fig. 12. IO Flownet Testing Loss

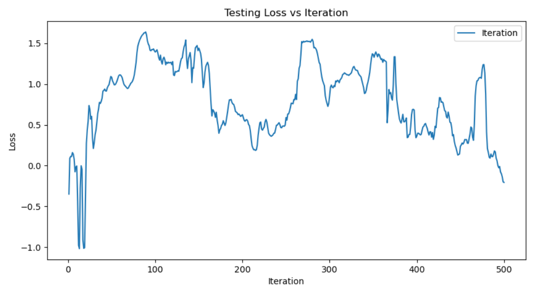

Fig. 13. VO Flownet Testing Loss with Adam

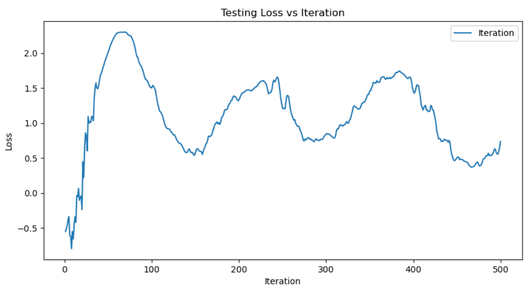

Fig. 14. VO Flownet Testing Loss with AdamW

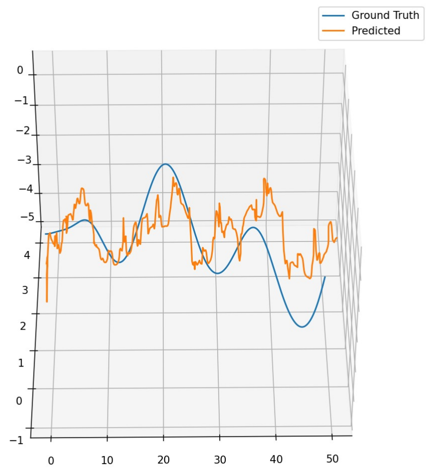

Fig. 17. IO Position Output

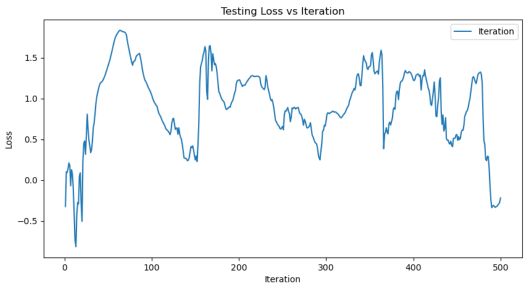

Fig. 15. VIO Flownet Testing Loss with Adam

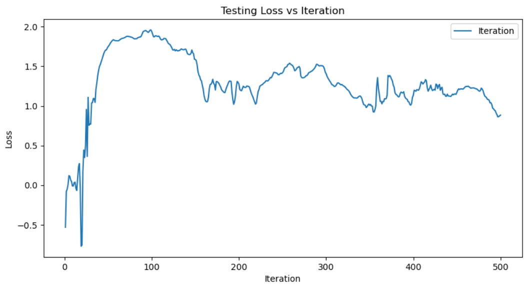

Fig. 16. VIO Flownet Testing Loss with AdamW

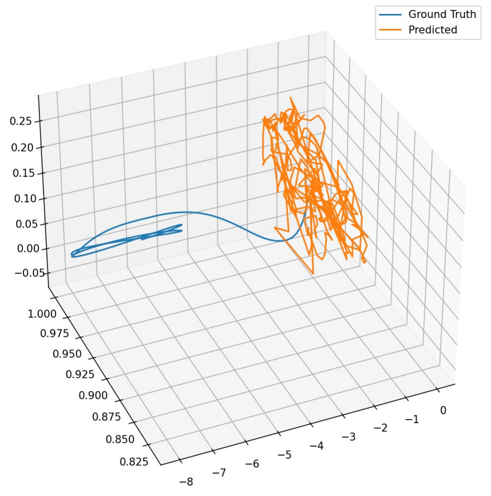

Fig. 18. IO Orientation Output

The outputs of the trained models are shown below.

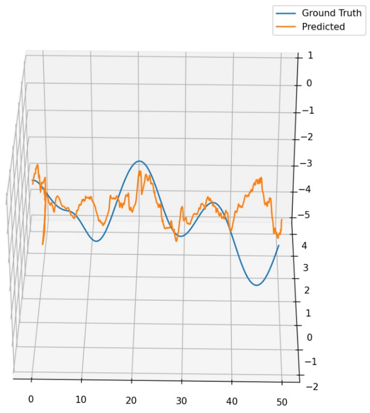

Fig. 19. VO Position Output

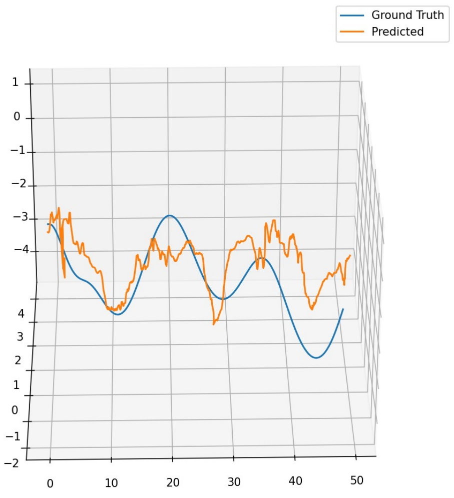

Fig. 21. VIO Position Output

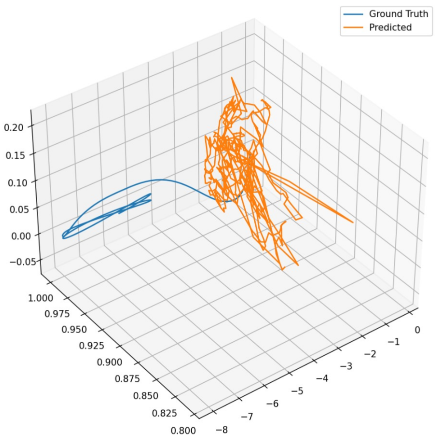

Fig. 20. VO Orientation Output

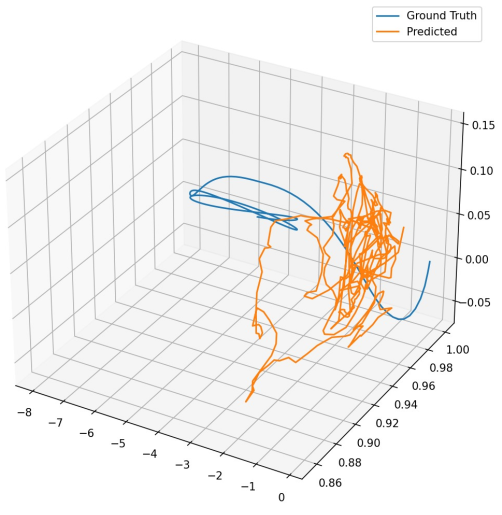

Fig. 22. VIO Orientation Output 3D view

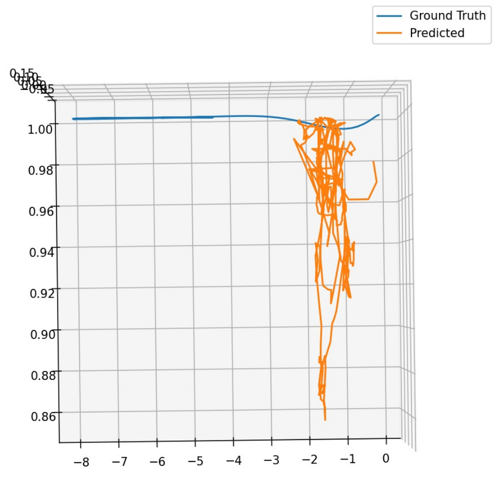

Fig. 23. VIO Orientation Output top view

VI. RPG Toolbox Results

Below we see the results generated using the RPG Toolbox

A. VO

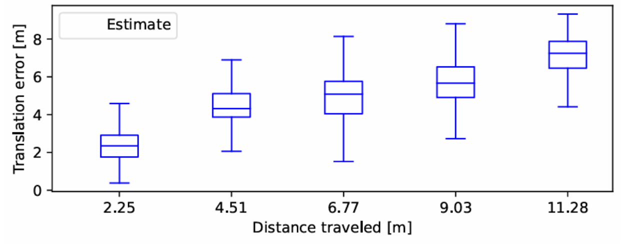

Fig. 24. VO Relative Translation Error

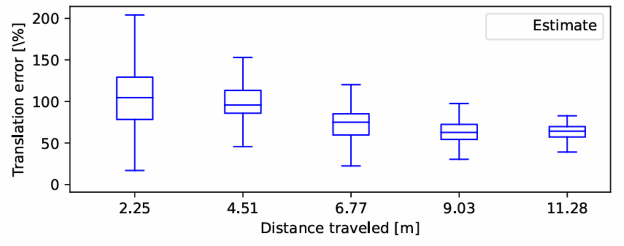

Fig. 25. VO Relative Translation Error (Percentage)

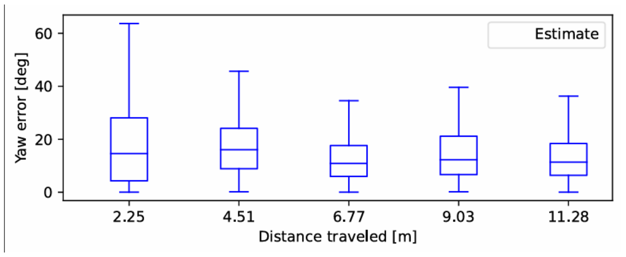

Fig. 26. VO Relative Translation Error

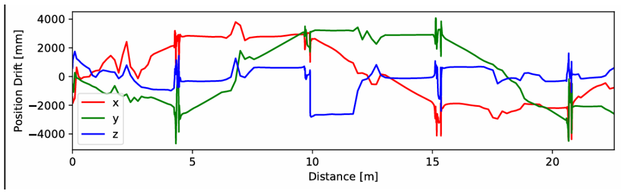

Fig. 27. VO Translation Error

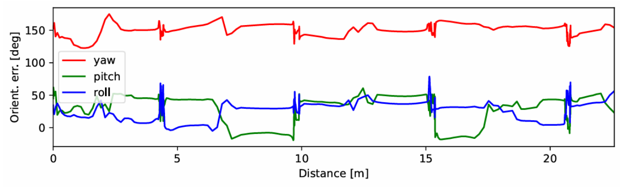

Fig. 28. VO Rotation Error

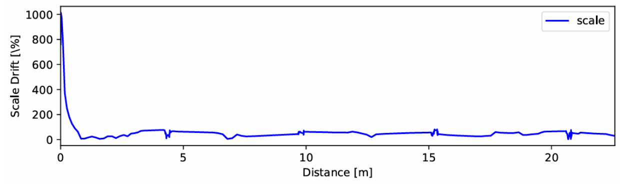

Fig. 29. VO Scale Error

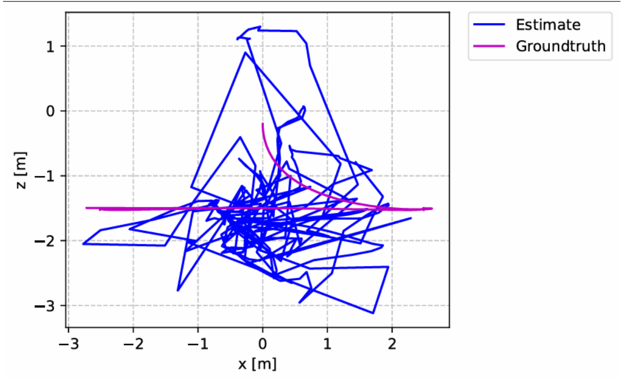

Fig. 30. VO Trajectory (Side)

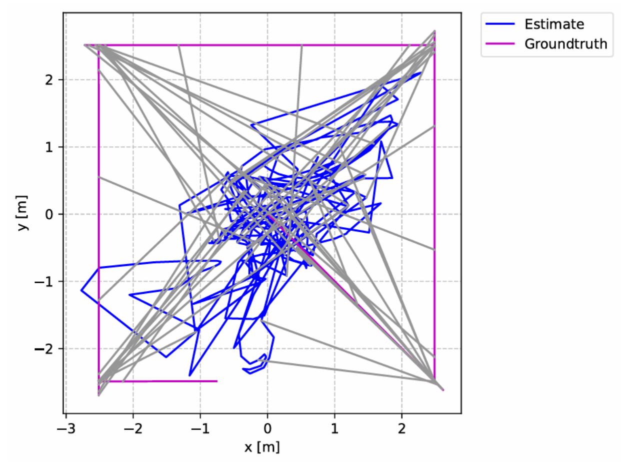

Fig. 31. VO Trajectory (Top)

B. VIO

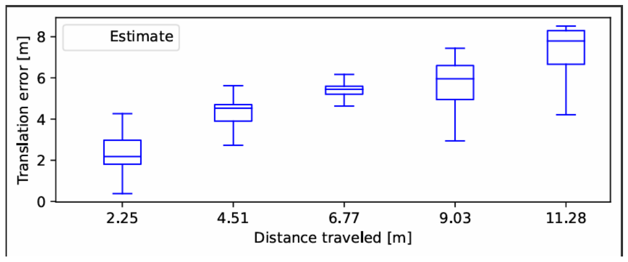

Fig. 32. VIO Relative Translation Error

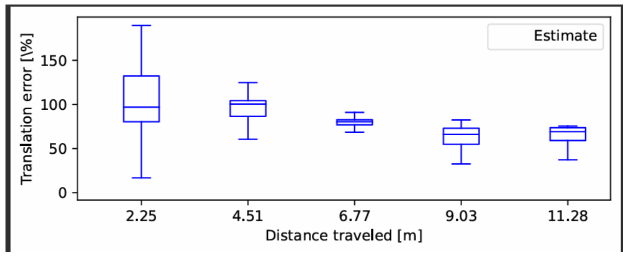

Fig. 33. VIO Relative Translation Error (Percentage)

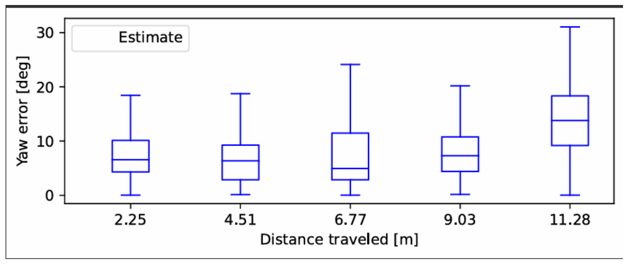

Fig. 34. VIO Relative Translation Error

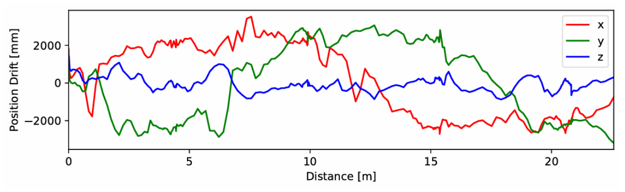

Fig. 35. VIO Translation Error

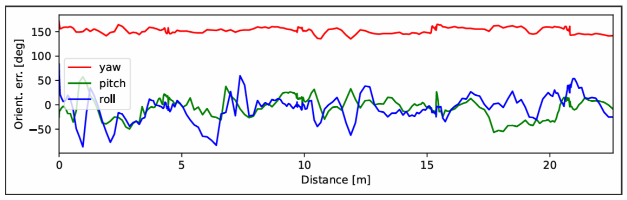

Fig. 36. VIO Rotation Error

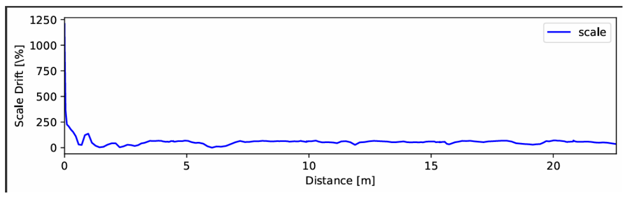

Fig. 37. VIO Scale Error

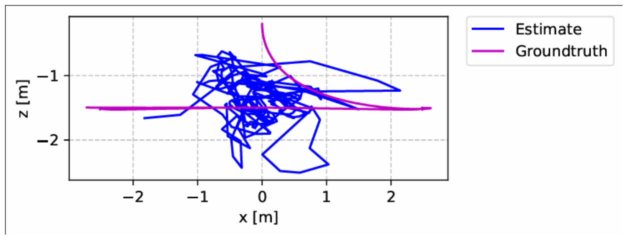

Fig. 38. VIO Trajectory (Side)

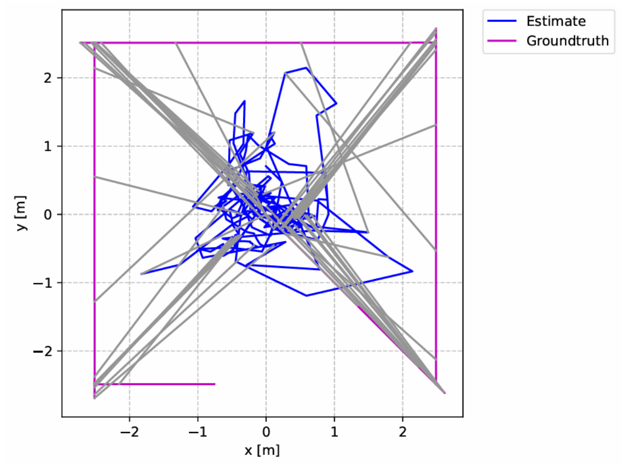

Fig. 39. VIO Trajectory (Top)

VII. Conclusion

We used various pre-trained neural network models for object detection. We processed these outputs with classical Computer Vision techniques to achieved expected results. This includes, color detection using HSV and RGB value, correction of 3D bounding boxes using projection matrix, curve fitting of lanes, contour detection.

VIII. Future Work

Deep learning-based approaches to Visual-Inertial Odometry (VIO) present novel opportunities and challenges. These methods can potentially outperform classical VIO techniques by leveraging the ability of deep neural networks to extract and process complex features from raw data.

A. Learning Under Uncertainty

Current deep learning models often operate under the assumption of certainty in their predictions, which is seldom the case in real-world scenarios, particularly in VIO. Incorporating models that can explicitly learn and reason about uncertainty is crucial. Techniques such as **Bayesian deep learning** offer a framework for modeling uncertainty by treating the network weights as distributions rather than fixed values. This approach allows the network not only to make predictions but also to provide confidence estimates, which are crucial for safety-critical applications like autonomous driving. Implementing Bayesian methods in deep VIO models could involve using variational inference to approximate posterior distributions or employing Monte Carlo Dropout to estimate uncertainty.

B. Data Efficiency and Semi-supervised Learning

A significant limitation of current deep learning methods is their reliance on extensive labeled datasets, which are expensive and time-consuming to produce. Addressing this issue, research could pivot towards **unsupervised, semi-supervised, and self-supervised learning methods**. These techniques reduce the dependency on labeled data by utilizing unlabeled data effectively. One promising direction is the development of systems that can simultaneously collect data and generate labels in a self-supervised manner, thus continuously learning and adapting from new environments without extensive manual annotation. This approach not only enhances data efficiency but also accelerates the adaptability of VIO systems to diverse conditions.

C. Network Architecture Innovations

Exploring novel architectures such as **recurrent neural networks (RNNs)**, **3D convolutions**, and **transformer models** can provide significant advancements. RNNs can handle data sequences effectively, making them suitable for temporal data in VIO. Similarly, 3D convolutions can process spatial-temporal video data directly, potentially improving the feature extraction process for motion analysis. Transformers, known for their superior performance in sequence modeling tasks, could be adapted to manage the sequential sensor data, offering improvements in both accuracy and computational efficiency.

References

[1] R. Clark, S. Wang, and H. Wen, ”VINet: Visual-inertial odometry as a sequence-to-sequence learning problem,” *arXiv preprint arXiv:1708.03852*, 2017.

[2] Y. Zhu, A. Garg, S. Huang, and S. Saripalli, ”Learn to Fuse: A Deep Learning Approach to Visual-Inertial Camera Fusion,” in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 6744-6751, 2017.

[3] A. Kendall, M. Grimes, and R. Cipolla, ”Posenet: A convolutional network for real-time 6-dof camera relocalization,” *arXiv preprint arXiv:1505.07427*, 2015.

[4] A. I. Mourikis and S. I. Roumeliotis, ”A multi-state constraint Kalman filter for vision-aided inertial navigation,” *IEEE Transactions on Robotics and Automation*, vol. 23, no. 1, pp. 108-120, 2007.

[5] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. T. Furgale, ”Keyframe-based visual–inertial odometry using nonlinear optimization,” *The International Journal of Robotics Research*, vol. 34, no. 3, pp. 314-334, 2015.

[6] M. He, C. Zhu, Q. Huang, B. Ren, and J. Liu, “A review of monocular visual odometry,” Vis Comput, vol. 36, no. 5, pp. 1053–1065, May 2020, doi: 10.1007/s00371-019-01714-6.

[7] R. Khorrambakht, C. X. Lu, H. Damirchi, Z. Chen, and Z. Li, “Deep Inertial Odometry with Accurate IMU Preintegration.” arXiv, Jan. 18, 2021. Accessed: Apr. 28, 2024. [Online]. Available: http://arxiv.org/abs/2101.07061

[8] Z. Tu, C. Chen, X. Pan, Z. Chen, and C. Chu, “Deep Learning Based Visual-Inertial Odometry withExternal Attention Mechanism,” in Advances in Guidance, Navigation and Control, L. Yan, H. Duan, and Y. Deng, Eds., Singapore: Springer Nature, 2023, pp. 5890–5897. doi: 10.1007/978-981-19-6613-2_569.

[9] S. Wang, R. Clark, H. Wen, and N. Trigoni, “DeepVO: Towards end-to-end visual odometry with deep Recurrent Convolutional Neural Networks,” in 2017 IEEE International Conference on Robotics and Automation (ICRA), May 2017, pp. 2043–2050. doi: 10.1109/ICRA.2017.7989236.

[10] M. Yang, Y. Chen, and H.-S. Kim, “Efficient Deep Visual and Inertial Odometry with Adaptive Visual Modality Selection.” arXiv, Oct. 19, 2022. Accessed: Apr. 28, 2024. [Online]. Available: http://arxiv.org/abs/2205.06187

[11] M. O. A. Aqel, M. H. Marhaban, M. I. Saripan, and N. Bt. Ismail, “Review of visual odometry: types, approaches, challenges, and applications,” SpringerPlus, vol. 5, no. 1, p. 1897, Oct. 2016, doi: 10.1186/s40064-016-3573-7.

[12] https://drive.google.com/file/d/1EhSywBhsXSyUUijVkeRaEO-5auc8paVO/view

[13] https://www.mathworks.com/help/nav/ref/imusensor-system-object.html

[14] https://github.com/mingyuyng/Visual-Selective-VIO/tree/main

[15] https://github.com/uzh-rpg/rpg-trajectory-evaluation