RBE 549 Computer Vision
Project 4 - Phase 2
Visual and Inertial Odometry
Deep Learning Approach

Taruneswar Ramuu*
Robotics Engineering Department
Worcester Polytechnic Institute
Worcester, MA, USA
Email: tramuu@wpi.edu

Soumik Saswat Patnaik*
Robotics Engineering Department
Worcester Polytechnic Institute
Worcester, MA, USA
Email: sspatnaik@wpi.edu

***Abstract*—This paper presents a novel approach to odometry estimation using synthetic data for computer vision applications in robotics. We generate a controlled dataset combining RGB images and six-degree-of-freedom inertial measurement unit (IMU) data, synthesized in Blender. This approach circumvents the limitations of traditional real-world datasets by allowing precise control over experimental variables. We evaluate three neural network architectures designed to process vision-only, IMU-only, and integrated sensor data, demonstrating the utility of synthetic environments in enhancing the accuracy of odometry estimations in robotics.**

***Index Terms*—Visual Inertial Odometry, Sensor Fusion,**

INTRODUCTION

Classical odometry estimation techniques in robotics have traditionally relied on feature detection and tracking, methods that require meticulous calibration and are sensitive to environmental variances. These challenges have prompted an exploration into the potential of deep learning approaches, which can potentially bypass some of the limitations inherent in classical methods by using deep feature matching technologies like SuperPoint or SuperGlue. While these advanced techniques offer robust alternatives for processing visual data, their application to inertial data remains complex due to the fundamentally different nature of the data involved.

The integration of visual and inertial sensors (VI-fusion) using deep learning is still an emerging field, with limited studies successfully combining these technologies. This paper seeks to bridge this gap by proposing three innovative deep learning architectures designed to predict relative camera poses from synthetic datasets created in Blender. These architectures include a CNN-LSTM network for image input, an LSTM network for IMU data, and a hybrid model that integrates both data sources. Each model outputs a translation vector (x, y, z) and a rotation quaternion ([x, y, z, w]), tailored to mimic real-world dynamics in a controlled environment. Through this research, we aim to demonstrate the effectiveness of deep learning models in overcoming the challenges posed by classical odometry techniques, providing a comparative analysis of their performance using various loss functions.

I. SYNTHETIC DATA GENERATION - DATASET

A. Trajectory Generation - MATLAB

The dataset for this study was meticulously crafted to simulate realistic aerial robot trajectories. Utilizing MATLAB, we generated 3D trajectory data reflecting typical quadrotor movements. The IMU sensor function simulated the readings of an actual IMU sensor, incorporating additional noise to enhance realism. The sensor operated at a frequency of 1000Hz, and trajectory data was segmented into corresponding timesteps. This data was exported to CSV files for subsequent processing in Blender using Python scripts.

B. Data Processing and Setup

Before rendering, the CSV files containing trajectory and IMU data were processed using Python scripts. This step ensured that the data could be accurately interpreted by Blender's rendering engine, maintaining consistency across all generated datasets.

C. Image Rendering - Blender

For visual data generation, a base plane with a varied image texture was created in Blender, representing the ground over which the simulated aerial vehicle would navigate. The trajectory and orientation data, prepared in the previous step, were encoded within a Blender script. This script automated the rendering process, generating frames for each timestep across the entire trajectory. Each frame was rendered as *.png* image at 320x240p resolution to ensure fast rendering. Ten sets of renders were produced for ten distinct simulated trajectories; nine sets were used for training our models, and one set was reserved for testing purposes. Side-by-side image plots included below illustrate the alignment between the simulated trajectories and their rendered counterparts, showcasing the fidelity and consistency of our synthetic dataset.

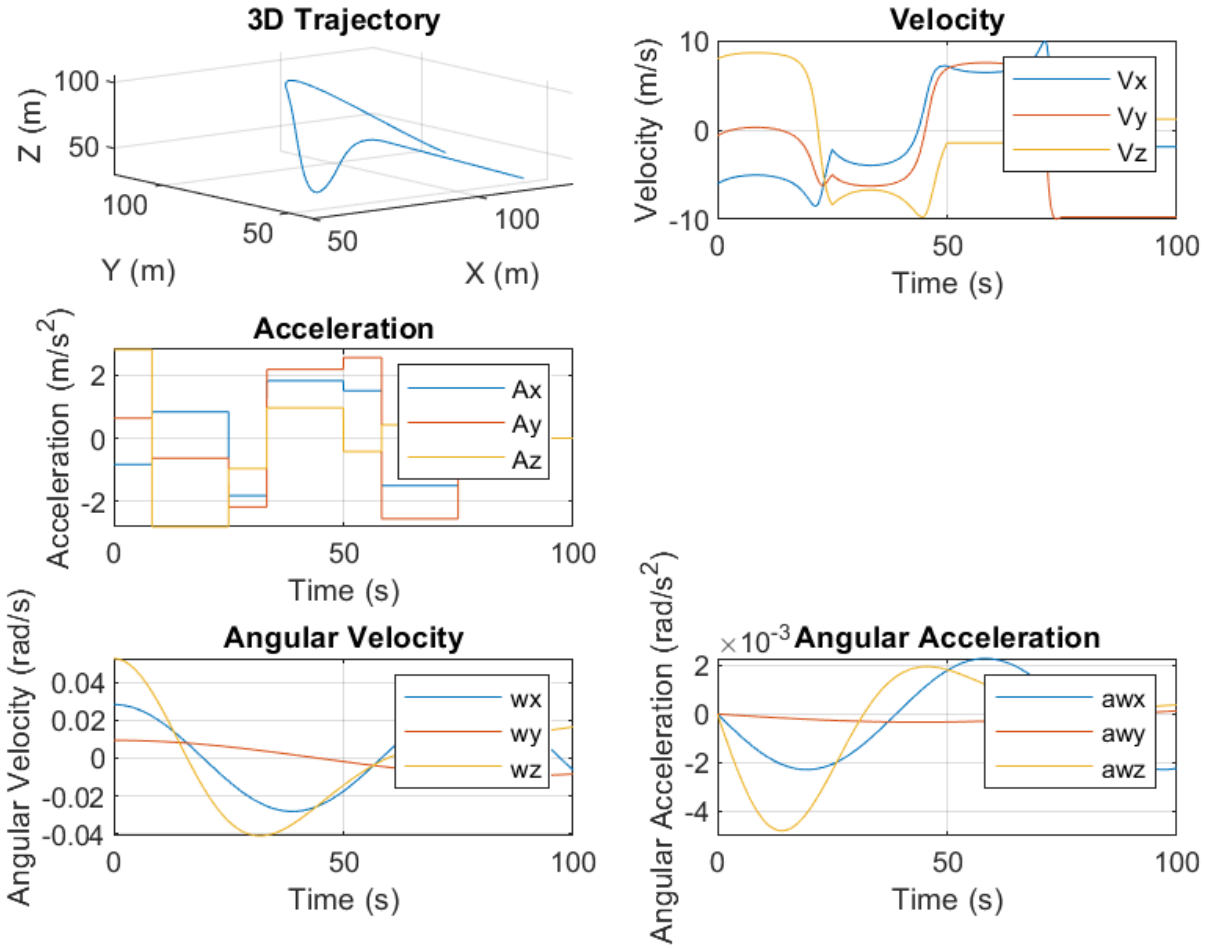
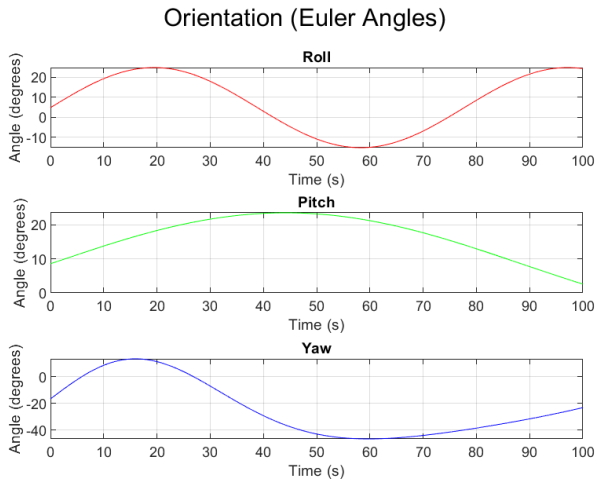
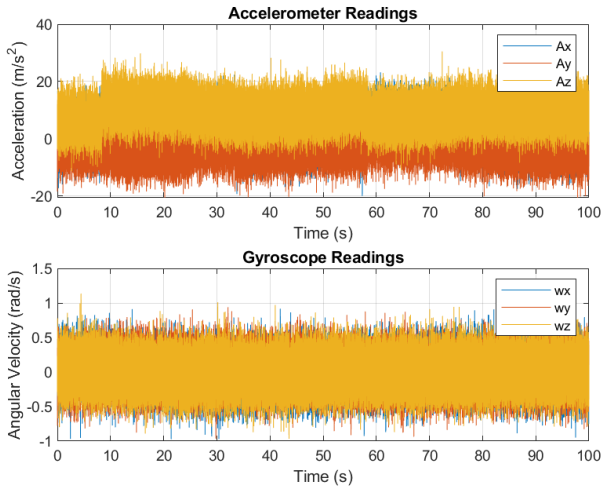


Fig. 1: *Trajectory Dynamics*



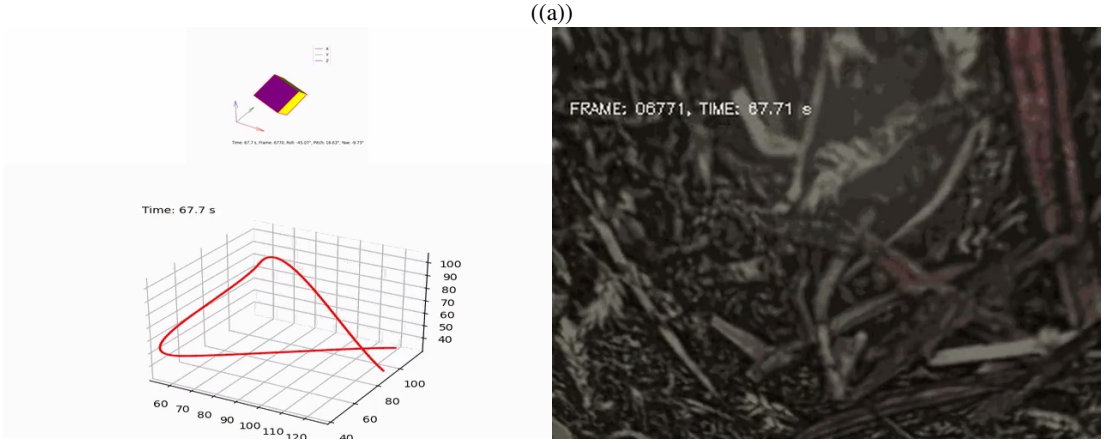
((a)) *Trajectory Orientation*



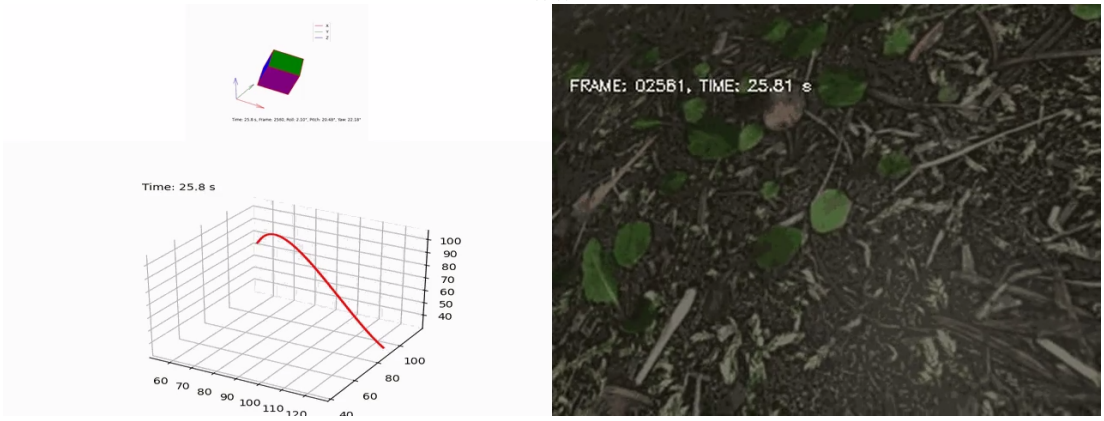
((b)) *IMU Simulation*

Fig. 2: Trajectory Simulation Characteristics - 1

((a))



((b))



((c))

((d))

Fig. 3: Trajectory Simulation vs Rendered - 1

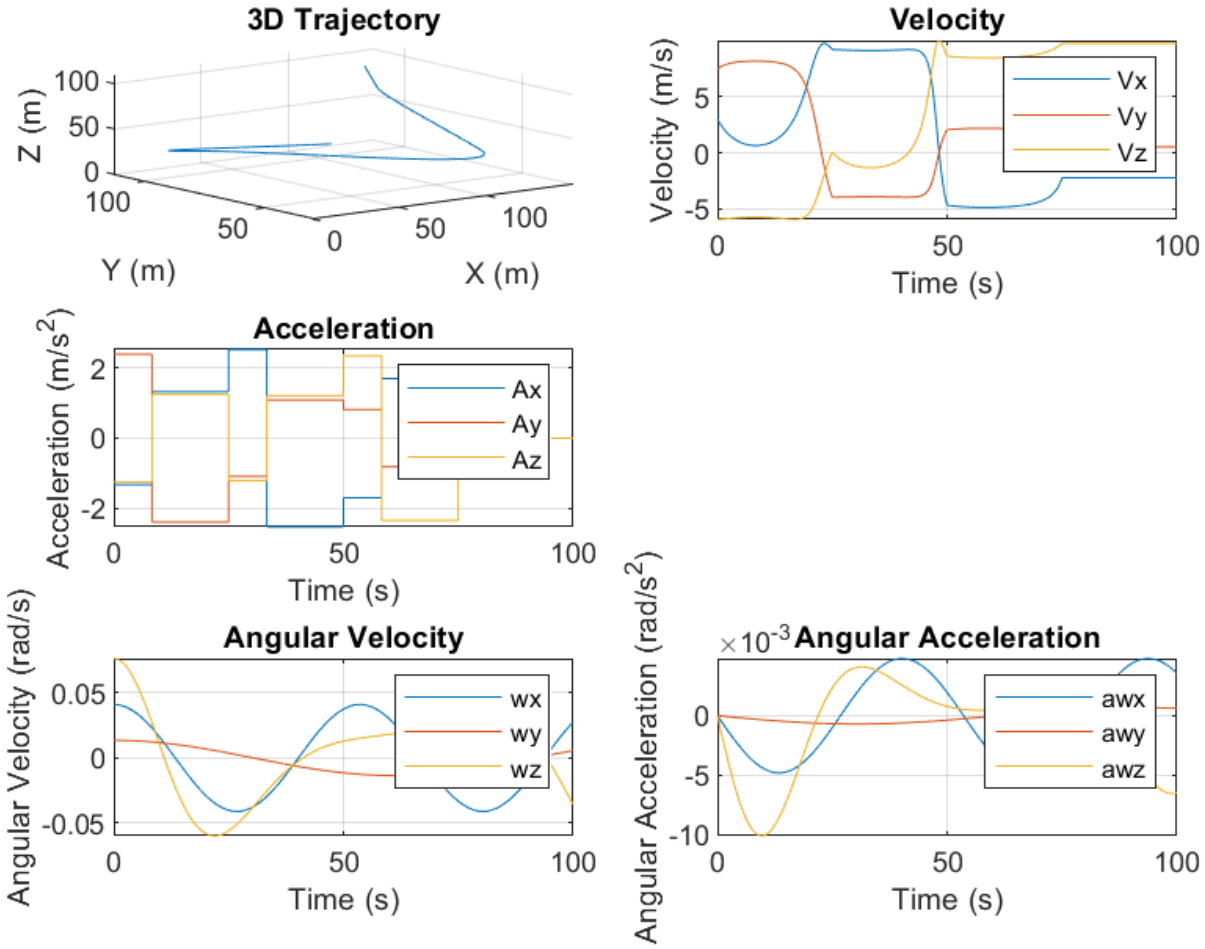
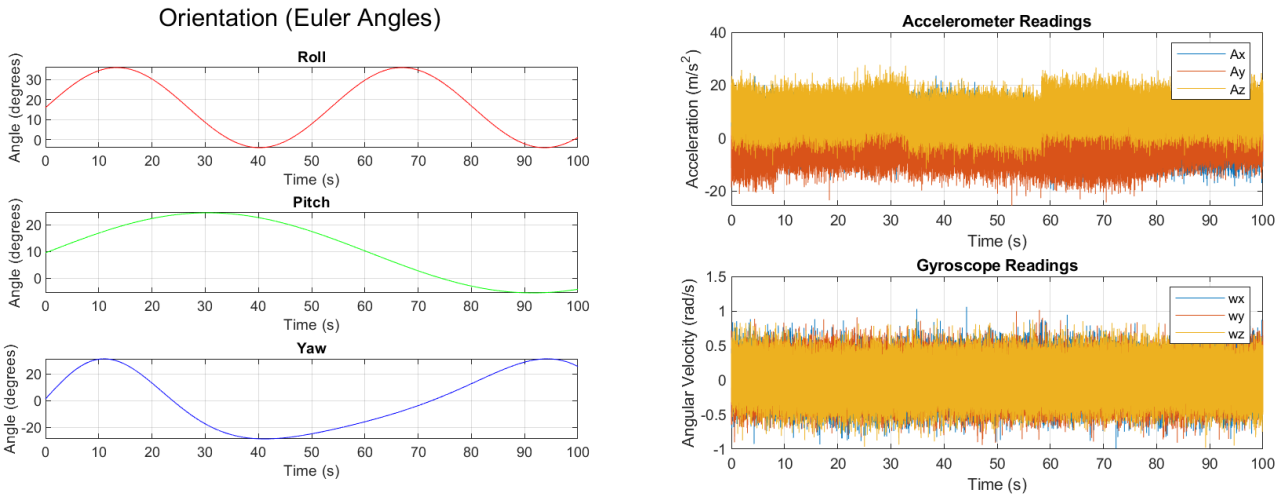
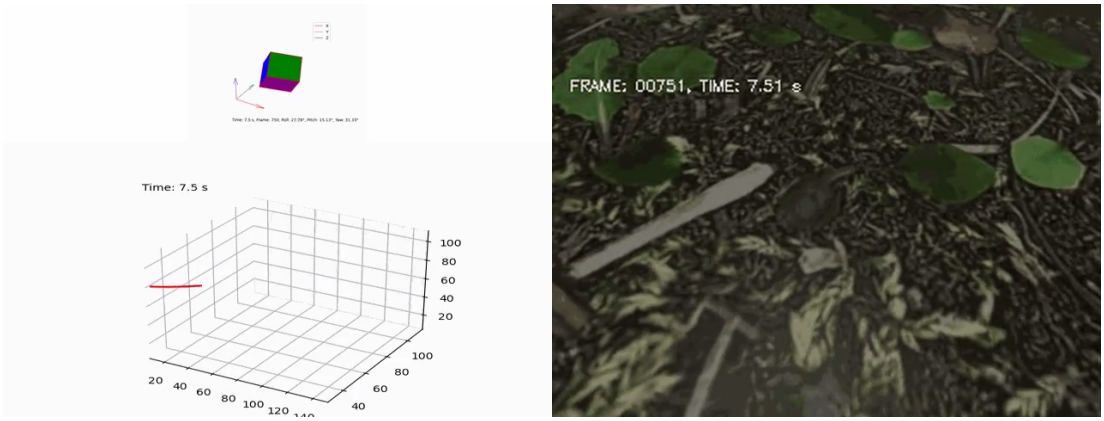


Fig. 4: *Trajectory Dynamics*

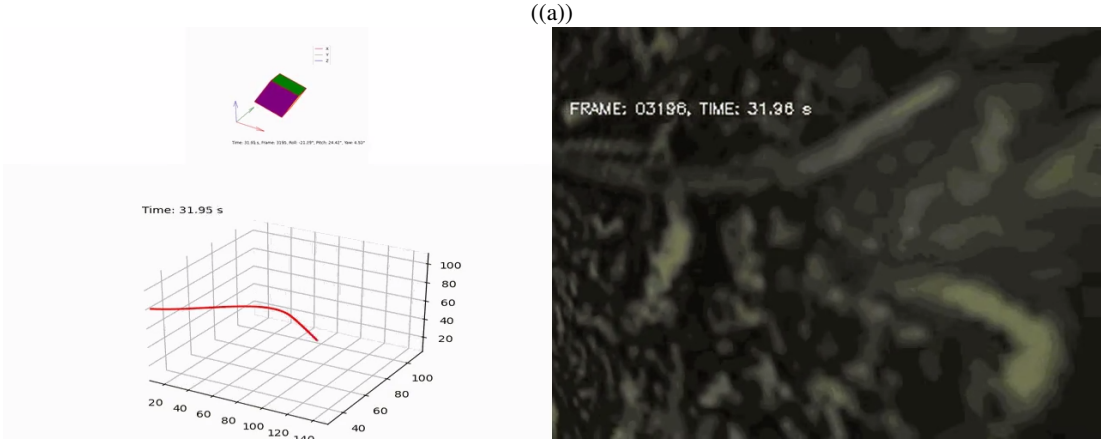


((a)) *Trajectory Orientation* ((b)) *IMU Simulation*

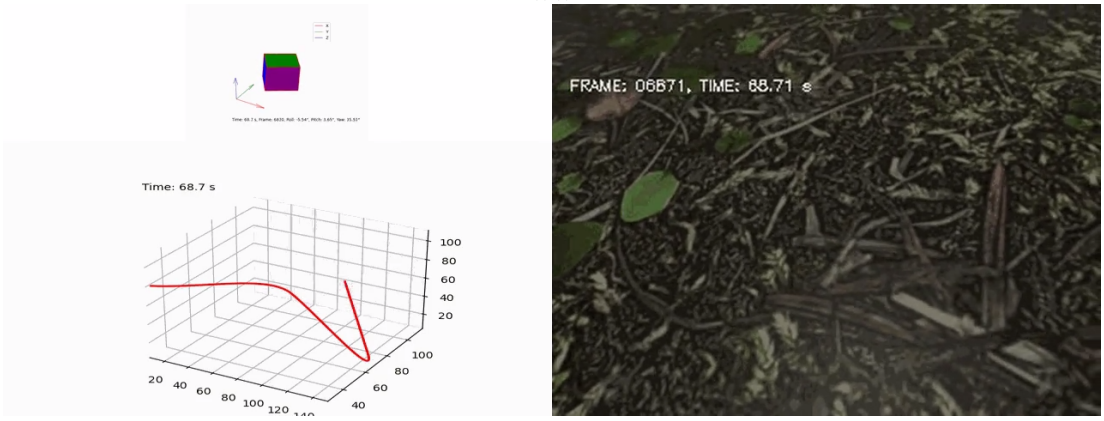
Fig. 5: Trajectory Simulation Characteristics - 2



((a))



((b))



((c))

((d))

Fig. 6: Trajectory Simulation vs Rendered - 2

II. Visual Odometry

A. Network Pipeline

The model architecture employed for visual odometry robustly combines Convolutional Neural Networks (CNNs) with Long Short-Term Memory (LSTM) networks. The detailed process flow includes:

- *Feature Extraction:* The initial layers are CNNs designed to identify spatial patterns within consecutive grayscale image frames. These layers are enhanced with batch normalization and ReLU activation functions, improving the efficiency of feature extraction.
- *Feature Refinement:* Following CNN processing, a fully connected layer refines these features to prepare them for temporal analysis.
- *Temporal Processing:* An LSTM layer takes over, processing the sequential feature vectors. This layer is crucial for capturing the temporal dependencies between frames by maintaining hidden states over time, which is essential for contextual understanding and temporal dynamics.
- *Prediction Output:* The LSTM's output is then passed through another fully connected layer to align the dimensions with the target output variables.
- *Training and Loss Computation:* The model is trained using a mean squared error loss function, enabling iterative refinement of predictions. This process enhances the model's accuracy in estimating position and orientation from sequential image data.

This integration of CNNs and LSTMs allows the model to effectively utilize both spatial and temporal information, making it suitable for applications in navigation, robotics, and augmented reality.

B. Data Preprocessing and Training

During the training phase, the dimensions of the images are critical, significantly influencing both the model's efficacy and computational demands. Key aspects of the image processing include:

- *Image Dimensions:* Images are uniformly resized to 128x128 pixels using the `transforms.Resize((128, 128))` transformation. This standardization ensures consistency and facilitates processing by the neural network.
- *Grayscale Format:* To reduce computational complexity, images are converted to grayscale, which comprises a single channel denoting light intensity. This simplifies the processing compared to multi-channel color images (e.g., RGB).
- *Input to CNN Layers:* The resized grayscale images are fed into convolutional layers (CNN), which extract spatial features essential for visual odometry. These layers use filters to detect patterns, edges, and textures.
- *Pooling Operations:* Post-convolution, max-pooling operations reduce the feature map size while retaining important features, utilizing a 2x2 kernel and a stride of 2 (`nn.MaxPool2d(kernel_size=2, stride=2)`).

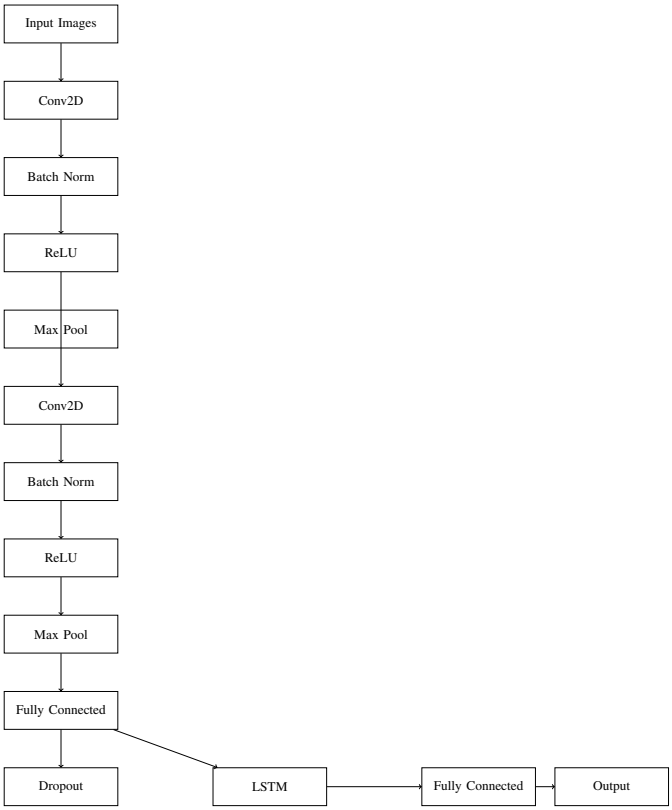


Fig. 7: CNN-LSTM Model Architecture for Visual Odometry

- *Flattening:* The output from the CNN layers is flattened into a one-dimensional vector before it proceeds to the fully connected layers. This process converts the spatial information into a format suitable for subsequent processing.

Standardizing image size and converting to grayscale ensures consistent input, promoting effective learning and feature extraction. Managing image resolution balances computational resources with the information content necessary for accurate predictions.

C. Results Analysis

Despite achieving convergence after 12 training epochs, the model exhibited inaccuracies on validation and test datasets, potentially due to overfitting. Overfitting likely occurred as the loss stabilized within the first 10 epochs with minimal subsequent improvement. Additionally, inaccuracies might also stem from improperly scaled orientation biases.

III. Inertial Odometry

1) Introduction: Inertial Odometry plays a crucial role in navigation systems, offering estimates of vehicle position and orientation by integrating measurements from inertial sensors such as gyroscopes and accelerometers. Conventional methods typically involve complex signal processing techniques and mathematical models, which may encounter challenges in handling non-linear motion dynamics and environmental uncertainties. Deep learning approaches, particularly those based

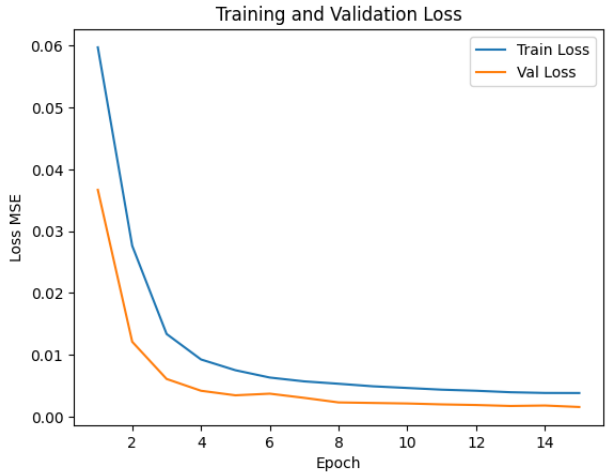


Fig. 8: Training and Validation Loss - Visual Network

on Long Short-Term Memory (LSTM) networks, offer promising capabilities in learning temporal patterns and capturing intricate dependencies within sequential data. In this paper, we introduce an LSTM-based architecture for Inertial Odometry, designed to enhance prediction accuracy and robustness in diverse operating conditions.

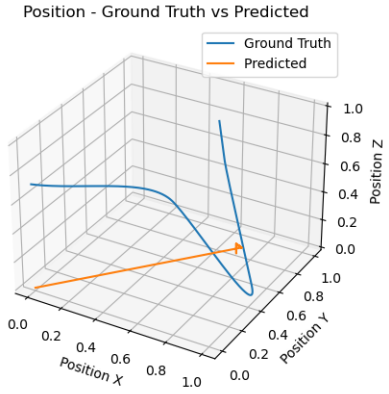
A. Network Pipeline

Our proposed architecture consists of an LSTM network followed by a fully connected layer. The LSTM network processes sequential input data from inertial sensors, while the fully connected layer generates predictions for vehicle position and orientation. The LSTM architecture enables the model to effectively capture temporal dependencies and adapt to varying motion dynamics, making it well-suited for Inertial Odometry tasks. Additionally, the use of deep learning techniques allows the model to automatically learn relevant features from raw sensor data, reducing the reliance on handcrafted algorithms and improving overall performance.

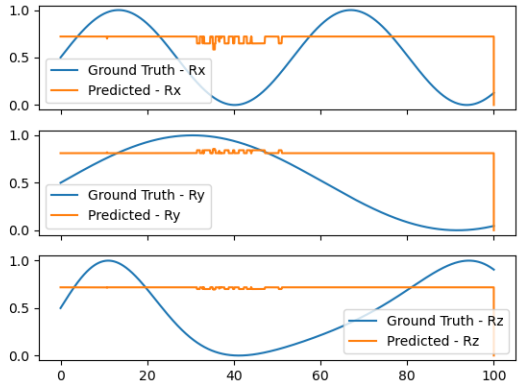
B. Data Preprocessing and Training

The training process is designed to optimize the model parameters to effectively minimize the Mean Squared Error (MSE) loss between the predicted values and ground truth. The overall training strategy includes:

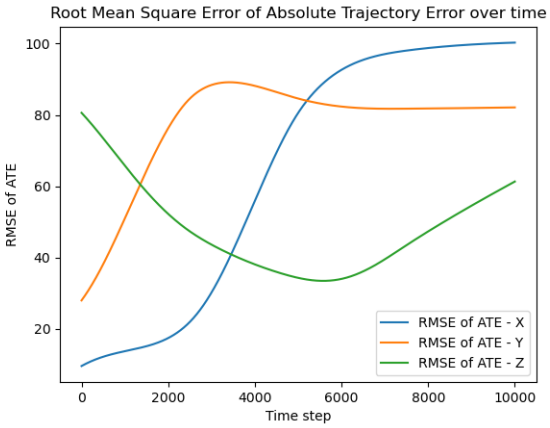
- *Optimizer and Learning Rate:* We utilize the Adam optimizer, known for its efficiency in handling sparse gradients and adaptive learning rate capabilities. The initial learning rate is set to 0.001.
- *Epochs and Evaluation:* The model undergoes training for 100 epochs. During this period, it is evaluated periodically on both the training and validation sets to monitor loss convergence and ensure that overfitting is minimized.
- *Training Goal:* The primary aim is to learn robust representations from inertial sensor data that accurately capture the dynamics of vehicle motion and orientation.



((a)) Position



((b)) Orientation

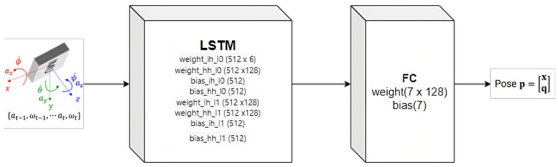


((c)) RMSE ATE

Fig. 9: Ground Truth vs Predicted Data - Visual Odometry

Before the actual training of the LSTM model, specific preprocessing steps are undertaken to optimize data compatibility and learning efficiency:

- *Normalization:* Sensor measurements are normalized using Min-Max scaling. This process scales the data to fall



((a))

LSTM Model Architecture

- Input Size: 6
- Hidden Size: 128
- Number of Layers: 2
- Output Size: 6

The LSTM model consists of an input layer, LSTM layers, and a fully connected (dense) layer.

$$\text{Input Layer: } X \in R^{(batch\_size, sequence\_length, input\_size)}$$

$$\text{LSTM Layers: } \begin{cases} \text{Input: } X \in R^{(batch\_size, sequence\_length, input\_size)} \\ \text{Hidden State: } h_0, c_0 \in R^{(num\_layers, batch\_size, hidden\_size)} \end{cases}$$

$$\text{LSTM: } (h_t, c_t) = \text{LSTM}(X, (h_0, c_0))$$
$$h_t \in R^{(batch\_size, sequence\_length, hidden\_size)}$$
$$c_t \in R^{(batch\_size, sequence\_length, hidden\_size)}$$

$$\text{Output Layer: } \hat{Y} = \text{Linear}(h_t)$$

$$\hat{Y} \in R^{(batch\_size, sequence\_length, output\_size)}$$

((b))

Fig. 10: LSTM Network Architecture - Inertial Network

within a specific range, which is crucial for mitigating issues due to varying sensor scales and aids in faster convergence during training.

- *Data Splitting:* The dataset is divided into training and validation sets. This split is critical for assessing model performance comprehensively and preventing overfitting during the training process.

Hyperparameters:

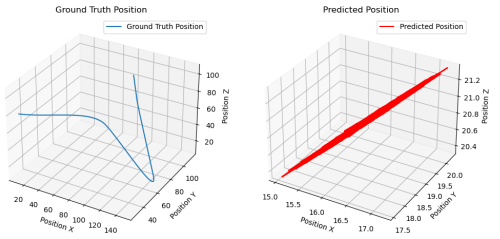
- *Optimizer:* Adam
- *Learning Rate:* 0.001
- *Epochs:* 100
- *Loss Metric:* Mean Squared Error (MSE)

These steps prepare the data in a format that is well-suited for training the LSTM-based Inertial Odometry model, facilitating efficient and effective learning.

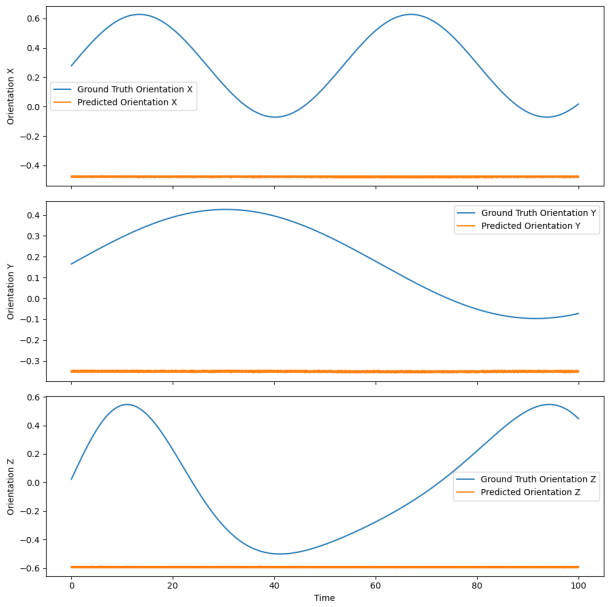
C. Results Analysis

The evaluation of the LSTM-based Inertial Odometry model revealed disappointing outcomes, as the predicted trajectory outputs were significantly inaccurate when compared to the ground truth. The primary observation was that the predicted trajectories were linear and failed to replicate the curved paths evident in the 3D ground truth data.

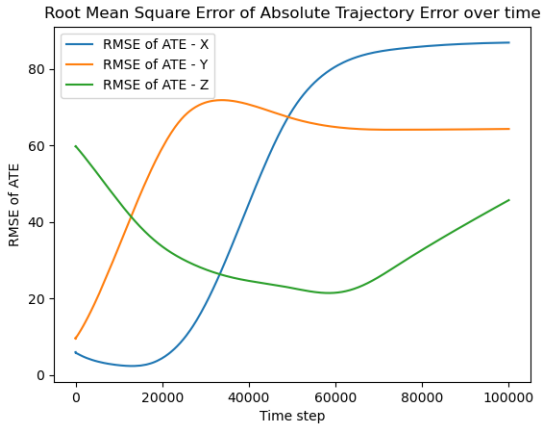
- *Deviation from Expected Path*: The most critical issue observed was the straight-line trajectory predicted by the model, which starkly contrasts with the complex, curved trajectories typical in realistic vehicle motions. This suggests fundamental flaws in either the model's capacity to capture complex dynamics or in the training process itself.



((a)) Position



((b)) Orientation



((c)) RMSE ATE

Fig. 11: Ground Truth vs Predicted Data - Inertial Odometry

- *Potential Causes for Poor Performance*:
  - *Inadequate Model Complexity*: The model may be underfitting the data, indicating that the architecture is not complex enough to learn the intricate patterns and relationships present in the training data.
  - *Improper Training or Convergence Issues*: The training process might not have been adequate. Possible issues include insufficient training epochs despite the use of 100 epochs, or premature convergence of the learning process, where the optimizer settled at a suboptimal point due to issues like high learning rate or inadequate loss function settings.
  - *Data Quality and Preprocessing*: There might be issues with how the data was scaled or normalized, or the initial assumptions used in data preprocessing could have distorted the sensor data's informative features.
- *Comparative Analysis*: Graphical representations comparing the predicted trajectories with the ground truth highlighted these inaccuracies clearly. Such visualizations are crucial for diagnosing the specific nature of the discrepancies.

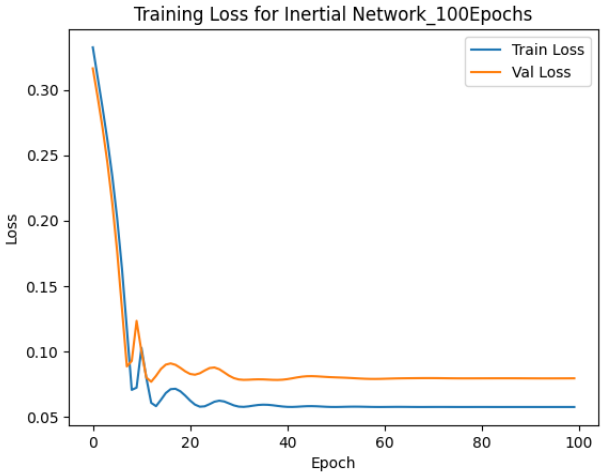


Fig. 12: Training and Validation Loss - Inertial Network

This analysis underscores the need for a thorough review of both the model architecture and the training regimen. Enhancements in these areas, along with a reassessment of data preprocessing techniques, will be vital for achieving more accurate and realistic trajectory predictions in future iterations of the model.

IV. Visual Inertial Odometry

We Combined the architectures from the visual and inertial networks and then passed it through a fully connected layer. We merged the network by passing the imu data and image data through the LSTM(combined) and then passed it through the Fully Connected layer. However, due to the mismatch in the IMU and Image frequencies, we weren't able to get viable results from the network. We present the network and the training and validation losses in fig. 13 and fig. 15 respectively.

A. Network Pipeline

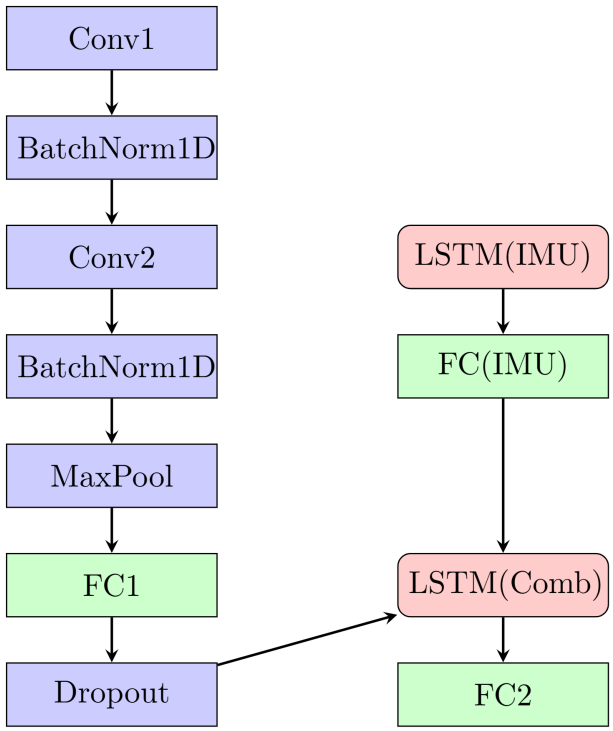


Fig. 13: Network Architecture -Visual and Inertial

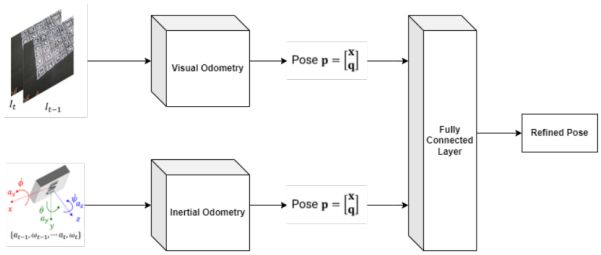


Fig. 14: General Network Architecture -Visual and Inertial

B. Training and Testing

We trained the network for 25 epochs after which the loss remained constant. However, the data used in the network training was one to one due to time constraint although we generated the data in the required format, we weren't able to train the network on the intended data. we used Adam optimizer with learning rate of 0.01 and MSE loss.

C. Results Analysis

The model exhibited suboptimal performance during both training and validation phases, indicating challenges in learning from the provided dataset. The model struggled to effectively capture the underlying patterns and relationships

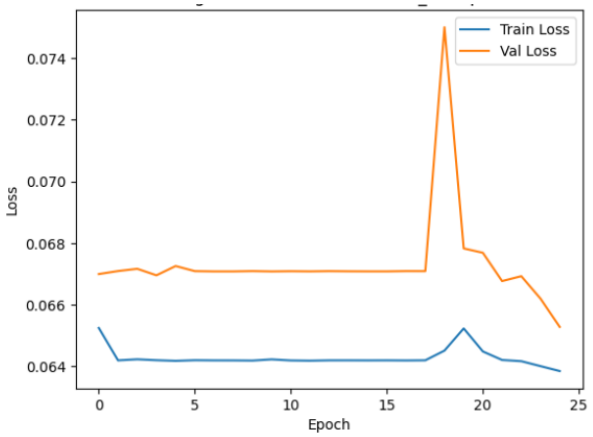


Fig. 15: Training and Validation Loss - Visual Inertial Network

within the data, leading to unsatisfactory results in terms of predictive accuracy and generalization. This poor performance underscores the complexity of the task at hand and highlights the need for further investigation into model architecture, data preprocessing, and training methodologies to improve overall performance.

Conclusion

This study aimed to estimate odometry using synthetic data generated in Blender for computer vision applications in robotics. We developed three distinct neural network architectures—visual-only, inertial-only, and a combined visual-inertial approach, utilizing LSTM and CNN-based networks. Although these models were designed to process and fuse RGB images and IMU data to predict translation vectors and rotation quaternions, the results were unsatisfactory. The integrated visual-inertial model, expected to outperform the single-sensor models, did not meet performance expectations, highlighting significant challenges in model training and data processing accuracy.

Future Work

The results underscore the need for substantial improvements in both the data quality and model architecture. Future research should focus on integrating additional sensor modalities, such as LIDAR or depth cameras, to enhance pose estimation accuracy. There is also a critical need to refine noise reduction techniques and develop more sophisticated neural network architectures that can handle complex environmental interactions more effectively. Additionally, experimenting with larger, more complex environments could provide deeper insights into the scalability and practical deployment of these models across diverse operational settings. Further efforts to optimize data preprocessing and model tuning are essential to achieving the high levels of accuracy required for practical applications in robotics.

References

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

[2] Juan Caballero, Christian Ledig, Andrew Aitken, Alejandro Acosta, Johannes Totz, Zehan Wang, Wenzhe Shi, "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, 2017, pp. 0-0.

[3] Huaijin Zhan, Chamara Saroj Weerasekera, Jiawang Bian, and Ian Reid, "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.

[4] Giuseppe Cioffi, Lennart Bauersfeld, Elia Kaufmann, and Davide Scaramuzza, "Learned Inertial Odometry for Autonomous Drone Racing," In IEEE Robotics and Automation Letters (RA-L), 2023.

[5] Ronald Clark, Sen Wang, Hongkai Wen, Andrew Markham, Niki Trigoni, "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, YEAR.

[6] Alex Kendall, Matthew Grimes, and Roberto Cipolla, "PoseNet: A Convolutional Network for Real-Time 6-DOF Camera Relocalization," YEAR.