Deep and Un-Deep Visual Inertial Odometry Using CNN, FlowNet and LSTM

Abhijeet Sanjay Rathi
Worcester Polytechnic Institute
asrathi@wpi.edu

Anuj Jagetia
Worcester Polytechnic Institute
ajagetia@wpi.edu

***Abstract*—In this paper, we propose a novel deep learning architecture for predicting relative camera poses utilizing both visual and inertial sensor data. Our approach integrates Convolutional Neural Networks (CNNs), Long Short-Term Memory (LSTM) networks, and a fusion strategy to effectively leverage information from image sequences and inertial measurements. We begin by constructing a combination of architectures, comprising a 3-layered CNN and an LSTM network, to predict relative poses based solely on visual input. Additionally, we develop an LSTM network to infer relative poses using inertial data alone. Subsequently, we introduce a Deep Visual-Inertial (VI) Fusion Network that seamlessly integrates the outputs of the vision-based and IMU-based networks. Our proposed framework, termed Deep VI Fusion, demonstrates promising results in accurately predicting relative camera poses. We didn't achieve a better accuracy for test dataset but for validation, we did achieve a accuracy of 79%.**

I. Introduction

The training pipeline involves several key steps to effectively train a visual-inertial odometry (VIO) model. At the core of the process is the utilization of a dataset containing both image data and inertial measurement unit (IMU) sensor readings. Each training iteration revolves around generating batches of samples from this dataset, pairing images with corresponding IMU measurements.

With the data prepared, the model undergoes a forward pass to predict the pose based on the provided inputs. This prediction is then compared to the ground truth pose using a predefined loss function. Through backpropagation, the model's parameters are adjusted to minimize this loss, aligning the predicted poses more closely with the ground truth values.

Throughout the training phase, meticulous monitoring of the model's performance occurs. This entails evaluating its performance on both training and validation datasets. Loss values are recorded for each iteration and epoch, offering insights into the model's learning progress over time. Additionally, periodic checkpoints are saved to capture the model's state at various stages, facilitating easy resumption of training and ensuring the preservation of learned insights.

To optimize the learning process, the Adam optimizer is employed, featuring a dynamic learning rate schedule that adjusts based on the current epoch. This adaptive learning rate mechanism enables fine-tuning of the optimization process, potentially enhancing convergence speed and overall performance.

Visualizations of loss metrics are generated and saved throughout training, offering a visual representation of the model's performance trends. These visualizations serve as valuable diagnostic tools, aiding in the identification of potential issues and guiding adjustments to the training process. Furthermore, the training process is logged using the Weights and Biases (wandb) library, enabling seamless tracking and sharing of experimental results, fostering collaboration and facilitating reproducibility.

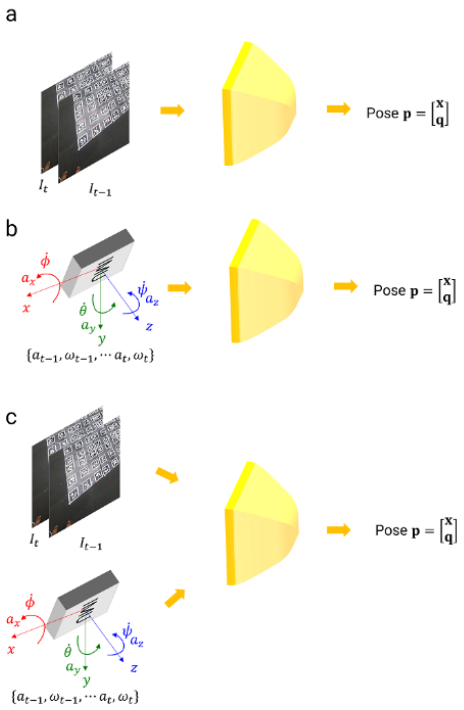

Fig. 1. *Visual , Inertial , VIO*[1]

II. Related Work

This section provides an overview of the evolution of Visual-Inertial Odometry (VIO) techniques, which are classified into three main categories: Classical Methods, Data-Driven Methods, and Temporal Adaptive Inference.

Classical Methods, which were prevalent in the early stages of VIO development, primarily employed sensor fusion techniques such as Extended Kalman Filters (EKF) and Unscented Kalman Filters (UKF) to integrate visual and inertial measurements for accurate motion estimation. A notable contribution in this domain is the work by Mourikis and Roumeliotis, who introduced a tightly-coupled VIO framework using an EKF for state

estimation. Subsequent research efforts expanded upon classical VIO methods to tackle challenges like sensor calibration, feature tracking, and drift mitigation.

In recent years, there has been a significant shift towards Data-Driven Methods, fueled by the success of deep learning in various computer vision tasks. Researchers have explored the application of convolutional neural networks (CNNs), recurrent neural networks (RNNs), and hybrid architectures to directly learn motion patterns from sensor data. For example, Clark et al. proposed a deep learning-based VIO system trained on large-scale datasets, demonstrating superior performance in challenging environments. Similarly, Zhu et al. introduced a novel VIO framework leveraging recurrent neural networks for temporal adaptive inference, enabling robust motion estimation in dynamic scenes.

A key idea in data-driven VIO techniques is temporal adaptive inference, which enables dynamic motion estimate modifications in response to shifting ambient factors and sensor attributes. Long Short-Term Memory (LSTM) networks, which are well-known for their capacity to represent sequential data and long-term dependencies, are essential to this methodology. Temporal adaptive inference improves the durability and accuracy of VIO systems by combining feedback mechanisms and adaptive learning algorithms, especially in situations when motion dynamics or sensor noise levels shift.

Overall, the development of VIO techniques shows a shift away from conventional sensor fusion strategies and toward more advanced data-driven approaches. To improve real-time performance and adaptability in dynamic contexts, temporal adaptive inference is becoming increasingly important.

III. Organization

Further this paper discusses about Methodology: Describes the proposed deep learning architecture. Each Network subsection explains the architecture and functionality of these components. Experiment Setup: Details the setup for generating synthetic data using Blender, introducing noise, preprocessing images, and implementing training procedures. It also discusses the evaluation metrics and tools used for visualizing and comparing trajectory results. Conclusion: Summarizes the key findings and contributions of your work, emphasizing the promising results achieved in predicting relative camera poses using the proposed deep learning architecture. Future Work: Outlines potential directions for future research, highlighting opportunities to enhance model performance, robustness, and adaptability in real-world scenarios.

IV. Methodology

A. Visual Encoder Net

This component processes image data using convolutional neural networks (CNNs) to extract relevant features. It comprises of 4 encoded convolutional layers with kernel size varying from 7 to 3, with batch normalization, leaky ReLU activation with a neagtive slope of 0.1, and dropout of 0.2 for regularization. Additionally, an LSTM layer with input size of 512 and the batch_first parameter is set to True, this layer is utilized to capture temporal dependencies in sequential data, enhancing the model's ability to handle time-varying inputs.

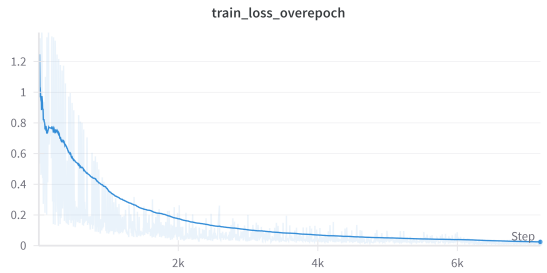

Fig. 2. *Train loss vs epoch*

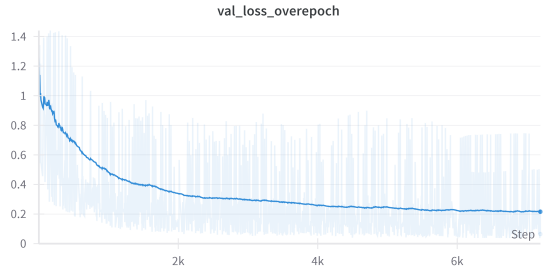

Fig. 3. *Train Val vs epoch*

B. Inertial Encoder Net

The inertial encoder processes inertial sensor data using 3 1D convolutional layers with input of channel 6 followed by batch normalization, leaky ReLU activation with a negative slope of 0.1, and dropout of 0.1. Similar to the visual encoder, it employs 1 LSTM layer which takes the input as the output of the conv 1D layers to capture temporal dynamics in the sensor readings, enabling the model to learn from sequential data and atlast there is one linear layer which fives back the output with 6 channels.

Fig. 4. *Avg Epoch vs Train loss*

C. Visual Inertial Fusion Net

This component combines information from both the visual and inertial encoders to make joint predictions. The outputs from the visual and inertial encoders are concatenated and passed through additional layers, including 1D convolutional layer followed by LeakyRelu with slope of -0.1 and again a linear layer, for fusion and final pose prediction.

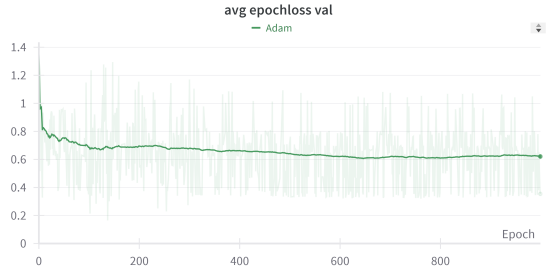

Fig. 5. *Avg Epoch vs Val Loss*

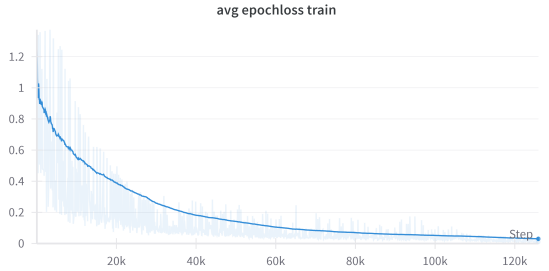

Fig. 7. *Avg Epoch vs Train loss*

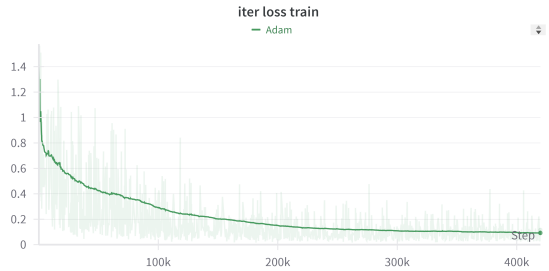

Fig. 6. *Avg Iteration vs Loss*

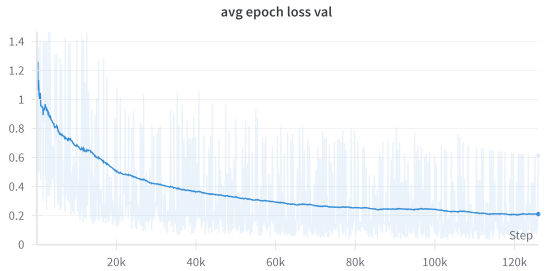

Fig. 8. *Avg Epoch vs Val Loss*

D. FlowNet

This FlowNet-S network is a pretrained on the FlyingChairs dataset for optical flow estimation. We implemented this pre-trained network in the backend which changes the weight of the input to the convlution layers.

V. Experiment Setup

To generate synthetic data, We used blender, in blender we created a large plane with varied image textures, ensuring that the camera was positioned to capture images of the floor. To generate the trajectory in the blender so that the camera follow the path we wrote the equations for the trajectory like circle , spiral and etc., it can be as many as we want to create the data, which gives us the ground truth in a csv file with position , quaternion angles, velocity and angular velocity. While the data rates of the IMU and camera were not synchronized, both were generated at the same rate for simplicity, with only every 10th sample of the image utilized for training while all IMU values were retained. This mimicked a scenario of a 1000Hz IMU and a 100Hz camera. Image textures on the floor were randomized, sourced from internet images, and scaled appropriately to ensure that the edges of the plane were not visible from the camera, thus maintaining data quality.

To introduce noise in the data we ran our reading from the ground truth and gave it to a imu sensor and calculated acceleration with the velocities we had and also converted quaternion into euler angles. From this we get data which accounts for real world conditions.

We resized the image from the data we got from blender into 320x180. this image is passed to our network. Before sending the data directly to the encoder network, first we used the Flownet network by loading a pre-trained model directly. The input size of the image is 320x180 and the total number of Epochs is 300 with batch size of 15. We tried implementing SGD, Adam and AdamW optimizer with varying learning rate of 5e-5 for epochs less than 250 and after that from 250-300 epochs we kept the learning rate to be 1e-6 and we kept a constant weight decay of 5e-6 throughout the whole process.

During training, we apply the mean squared error (MSE) loss to reduce the pose estimation error given and we also apply the cosine embedded loss to reduce the angle estimation error. For the overall lost we took the weighted average of these two loss with 40 % of pose loss and 60 % of angle loss.

To visualise and evaluate the trajectory we used a rpg toolbox repo from github by giving it the trained model we got from all three networks and the ground truth data which we had to get the comparison.

Optimizer	AdamW	SGD	Adam
Train	0.0297	0.0394	0.08572
Validation	0.2149	0.313	0.2271

TABLE I
Comparison of Different Optimizers

VI. Conclusion

We've developed a novel deep learning architecture for predicting relative camera poses using both visual and inertial sensor data. Our approach combines Convolutional Neural Networks (CNNs), Long Short-Term Memory (LSTM) networks, and a fusion strategy to effectively utilize information from image sequences and inertial measurements. Our framework, termed Deep VI Fusion, demonstrates promising results in

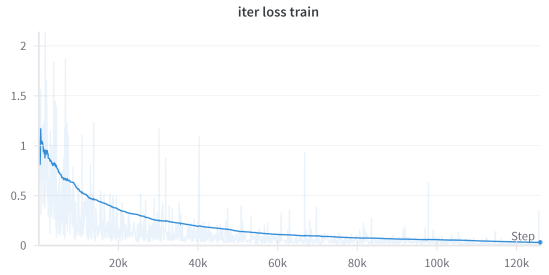

Fig. 9. *Avg Iteration vs Train*

Fig. 10. *Trajectory Path in Blender*

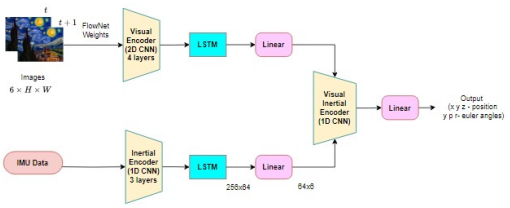

Fig. 11. *Network Architecture*

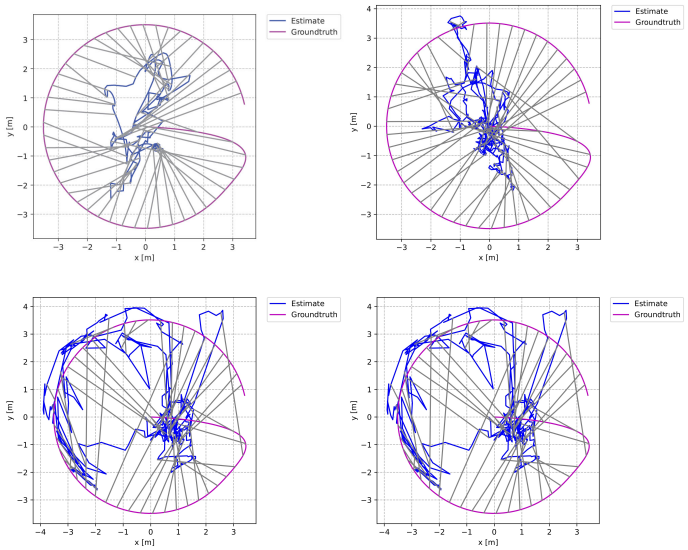

Fig. 12. Circle Trajectory (Top Left: Inertial, Top Right: Visual, Bottom Left: VIO-SGD, Bottom Right: VIO-AdamW)

accurately predicting relative camera poses. This achievement represents a significant advancement in visual-inertial odometry, showcasing the potential of deep learning approaches to enhance performance and efficiency in pose estimation tasks.

VII. Future Work

Future work in the realm of Deep learning-based Visual-Inertial Odometry (VIO) involves addressing both opportunities and challenges. While these methods hold promise in surpassing classical techniques by harnessing deep neural networks' capacity to extract intricate features from raw data, there are critical areas for improvement. Initial experiments often rely on synthetic data in controlled environments; however, future endeavors should prioritize incorporating real-world datasets to validate model performance across diverse and dynamic conditions. Moreover, efforts should be directed towards enhancing model robustness to environmental variations like lighting changes, surface textures, and dynamic scenes. This could entail exploring data augmentation techniques or domain adaptation methods. Additionally, investigating adaptive fusion strategies that dynamically adjust the weighting between visual and inertial modalities based on scene complexity, motion dynamics, or sensor data reliability could significantly enhance efficiency and accuracy across various scenarios.

VIII. Acknowledgement

We express our sincere gratitude to Nitin J. Sanket for their invaluable guidance. Thanks to Worcester Polytechnic Institute for providing resources and our peers for their support throughout the whole research.

References

[1] https://rbe549.github.io/spring2024/proj/p4/
[2] M. Yang, Y. Chen, H.S. Kim, *Efficient Deep Visual and Inertial Odometry with Adaptive Visual Modality Selection* 2020.
[3] P. Fischer, A. Dosovitskiy, E. Ilg, P. Hausser, C. Hazırbas¸, V. Golkov, University of Freiburg, Technical University of Munich *FlowNet: Learning Optical Flow with Convolutional Networks* 2020.
[4] https://github.com/uzh-rpg/rpg_trajectory_evaluation
[5] https://github.com/Aceinna/gnss-ins-sim