RBE/CS549: Deep and Un-Deep Visual Inertial Odometry - Phase 2 with Deep Learning

Shrishailya Chavan
WPI Robotics Engineering
Worcester Polytechnic Institute
schavan@wpi.edu

Sreejani Chatterjee
WPI Robotics Engineering
Worcester Polytechnic Institute
schatterjee@wpi.edu

***Abstract*—In this paper we build a Deep Learning based architecture to predict relative camera poses of the two cameras using the images and IMU data provided in EuRoC Dataset. We first implement a combination of architectures with a 3 layered Convolutional Neural Network, and an LSTM network to predict relative pose of two cameras by stacking their respective images as input. We then build an LSTM network to predict the relative poses of the same cameras by using the IMU data as input. Finally we combine the two architectures for vision only input and IMU only input to implement a Deep VI fusion network to predict relative camera poses. We tried with two different customized loss functions to train the network and comparing the data we finalized one loss function. With the selected loss functions.**

***Index Terms*—Deep VO, Deep Learning VI fusion**

I. Introduction

After implementing Phase1 we observed that traditional methods largely depend on feature detection and tracking. It was natural to ponder if it's possible to simply substitute this with deep feature matching techniques such as SuperPoint or SuperGlue. Even though this is a viable option, it can be challenging to identify a similar set of features for inertial data. Though there have been remarkable advancements in Inertial navigation, only a handful of studies address VI-fusion using deep learning. In this paper we attempted to bridge that gap by implementing three deep learning architectures to predict relative camera poses using the images and IMU data provided in EuRoC dataset. We first build a combination of CNN and LSTM network to predict camera poses with only stacked image inputs. We then built an LSTM network to predict the same relative camera poses using only the IMU data as inputs. Finally we combine the two architectures to accomplish the same task with both vision and inertial input. For all the inputs our predicted output should be a trabslation vector of x, y, z and a rotation quaternion vectore of [x, y, z, w]. We were able to test the networks with two different loss functions and compare the results.

II. Dataset

We used the entire EuRoC Dataset that is Machine Hall 01 Easy, Machine Hall 02 Easy, Machine Hall 03 Medium, Machine Hall 04 Difficult, Machine Hall 05 Difficult. We first combined all the subsets. Then we aligned the images with the IMU data using timestamps. The aligning was important as the data were collected in different time instances and is important to figure out the relative poses of the camera at a particular time instant.

III. Methodology

We divide the section into Vision only, Inertial Only and Visual Inertial architectures

A. Vision Only:

We started with only image inputs. We started to implement Resnet101 as it has image weights by default

1) Resnet101:: The Resnet101 results were not close to what we were expecting. Fig 1, 2 shows the result we received

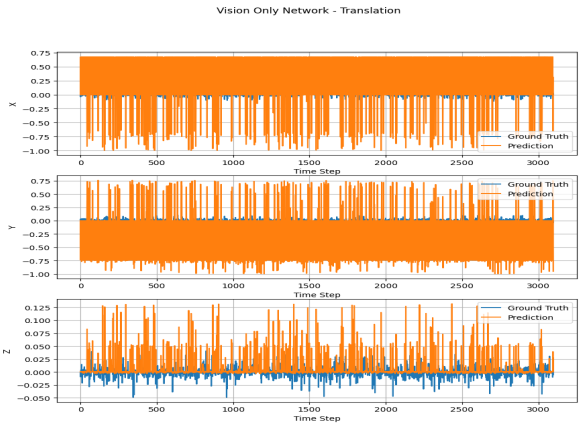

Fig. 1. Translation Plot for Resnet101

2) Resnet50:: We tried with resnet50 with not much luck. Fig 3, 4 shows the result we received.

3) Customized Architecture - CNN + LSTM:: We created a simple network of 3 layers of Convolutional Neural Network and added them with a layer of LSTM. We did this extra addition since the data are time stamps and LSTM tend to do well for data collected over a period of time. Fig 5 is an overview of the architecture. We divided the dataset in train, test and validation with 80% training data, 10% each for test and validation data. We trained the network for 50 epochs and 16 batches with a learning rate of 0.0001.

Fig. 2. Rotation Plot for Resnet101

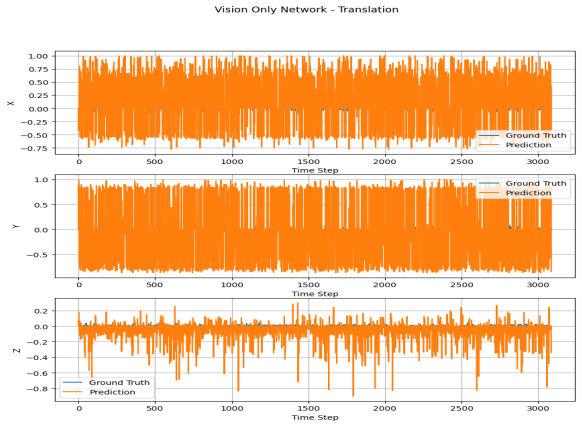

Fig. 3. Translation Plot for Resnet50

4) Loss:: We created a Loss function where we used Mean Squared Error Loss in Eq. 1 function for translation and Geodesic Loss described in Eq. 2 for rotation prediction of the relative pose.

$$L_{\text{MSE}} = \|\mathbf{p} - \mathbf{p}^*\|^2 \tag{1}$$

$$L_{\text{Geodesic}} = d(R_s, R_{GT}) = \cos^{-1}\left(\frac{\text{tr}(R_s^T R_{GT}) - 1}{2}\right) \tag{2}$$

This loss gave us really bad training and validation loss plot as shown in Figs. 6 and 7

We then created another loss where we are combining a weighted average of MSE Loss eq. 1 and Cosine Similarity Loss in eq. 3 and applying them to train both translation and rotation data of the poses. This loss gave us promising plot as shown in Figs. 8 and 9

$$L_{\text{cosine}} = 1 - \frac{\mathbf{a} \cdot \mathbf{b}}{\|\mathbf{a}\|\|\mathbf{b}\|} \tag{3}$$

Fig. 4. Rotation Plot for Resnet50

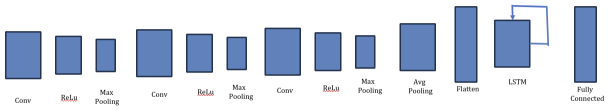

Fig. 5. Architecture for Vision Data

B. Inertial Only:

Here we are using only IMU data from our dataset to predict relative camera poses.

1) Long Short Term Memory - LSTM: We used one layer of LSTM network for this section. Fig. 10 shows the architecture we used. We divided the dataset in train, test and validation with 80% training data, 10% each for test and validation data. We trained the network for 50 epochs and 16 batches with a learning rate of 0.0001. We also used the same working loss described in III-A4. The training and validation losses are described in Figs. 11 and 12.

C. Visual Inertial Fusion:

This is where we combine our architectures from III-A3 and III-B1 to predict relative camera poses using aligned data from image and IMU data as inputs.

1) CNN+LSTM+LSTM:: The architecture is briefly overviewed in the Fig. 13. We divided the dataset in train, test and validation with 80% training data, 10% each for test and validation data. We trained the network for 50 epochs and 16 batches with a learning rate of 0.0001. We used the same working loss as used in Vision and Inertial Only networks in III-A4. Figs. 14 and 15 depicts the plots of training and validation loss for this architecture.

IV. Results:

In this section we will compare the ground truth translation and rotation data with predicted translation and rotation data for relative camera poses as implemented for III-A3, III-B1

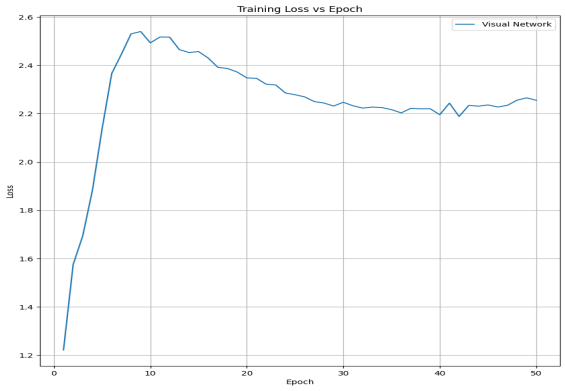

Fig. 6. Original Loss Training

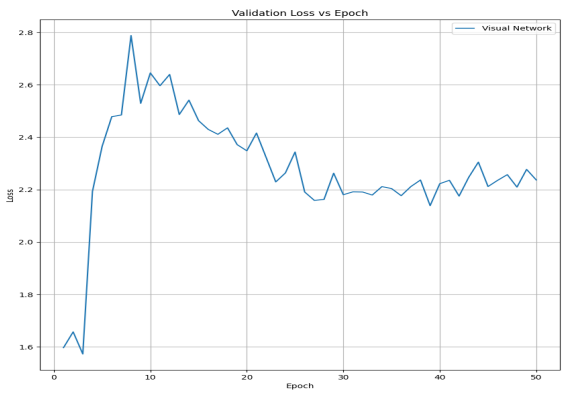

Fig. 7. Original Loss Training

and III-C1. Figs. 16, 17, 18, 19, 20, 21 are the respective plots. Here we have converted quaternion rotation to Euler angles before plotting. We have also plotted the 3D trajectory of combined rotation and translation results for all three network's output as depicted in Figs. 22, 23 and 24.

V. Conclusion and Future Work:

- We implemented three Deep Learning network to predict camera pose
  - With only image inputs
  - Only IMU Data
  - Both Images and IMU Data
- We were able to create and compare two customized loss functions.
- Train with better hyperparameters tuning in the future might give better result
- Use the prediction data to properly plot camera poses with translation and rotational data
- Align timestamped data correctly in future

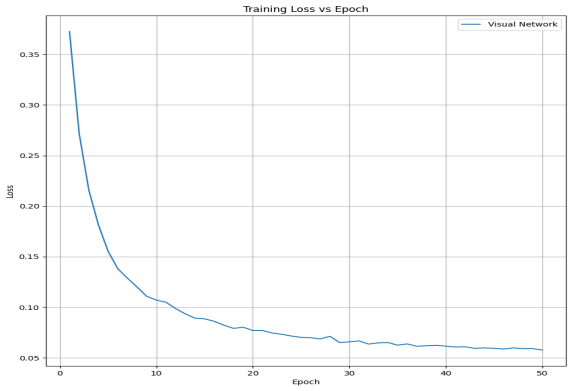

Fig. 8. Customized Loss Training

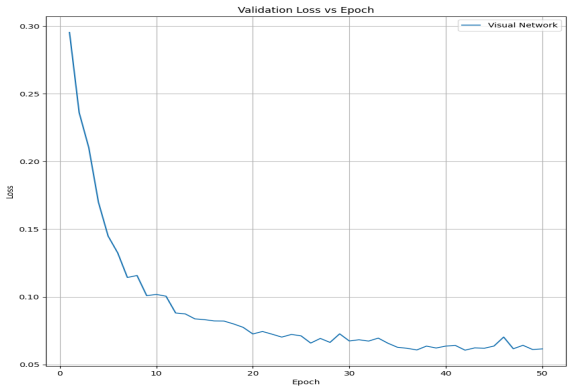

Fig. 9. Customized Loss Training

References

[1] https://ieeexplore.ieee.org/document/7989236
[2] https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6438698/pdf/nihms-1520737.pdf

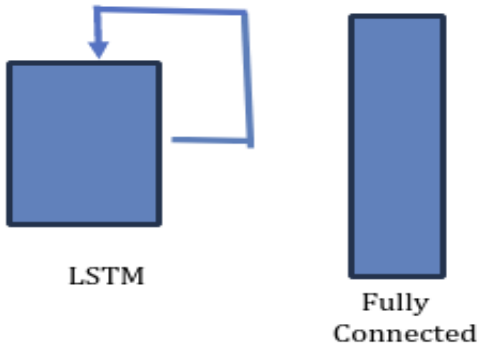

Fig. 10. Architecture for Inertial Only

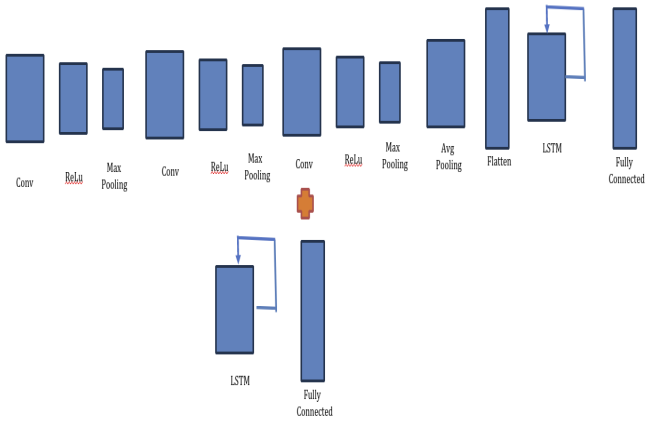

Fig. 13. Architecture for Visual Inertial Fusion

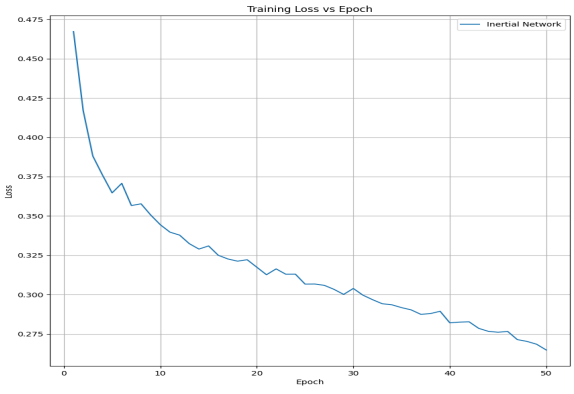

Fig. 11. Training Loss on IMU data

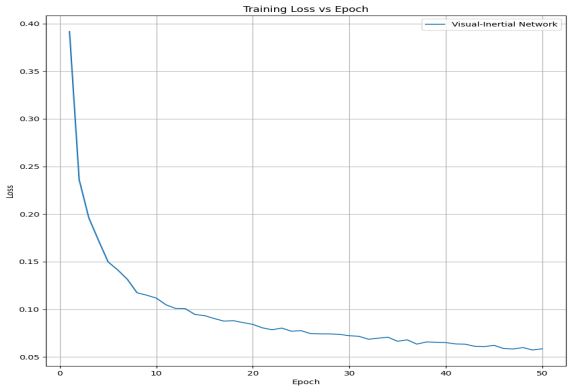

Fig. 14. Training Loss for Visual Inertial Fusion

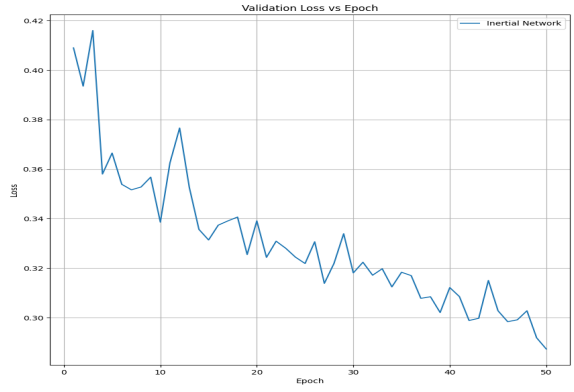

Fig. 12. Validation Loss on IMU data

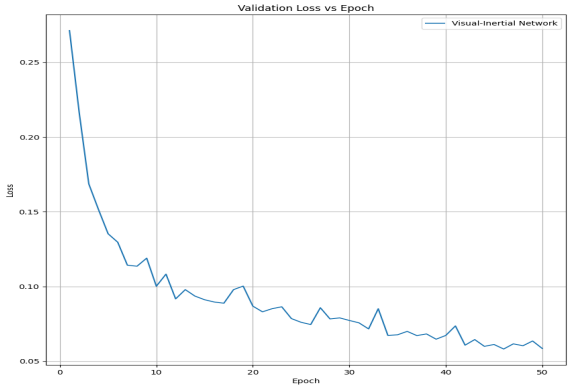

Fig. 15. Validation Loss for Visual Inertial Fusion

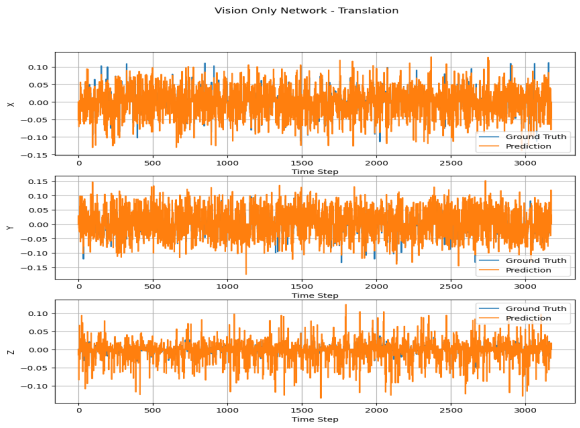

Fig. 16. Translation Vision

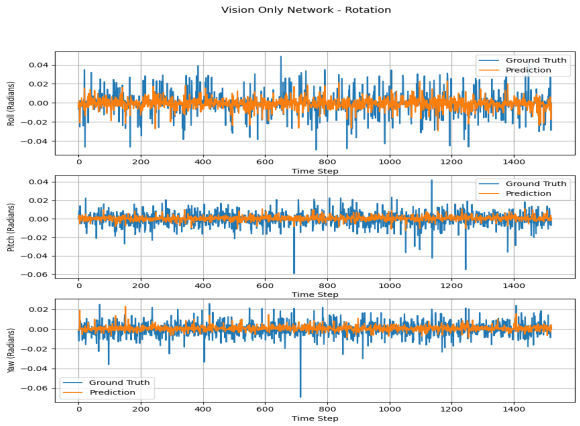

Fig. 17. Rotation Vision

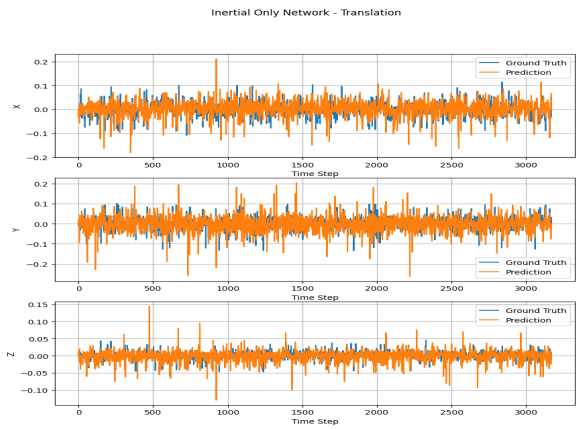

Fig. 18. Translation Inertial

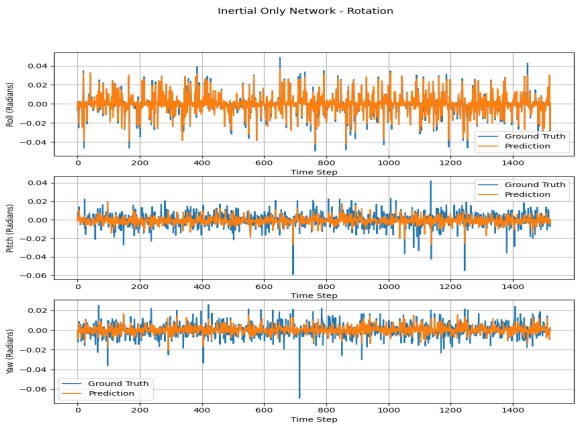

Fig. 19. Rotation Inertial

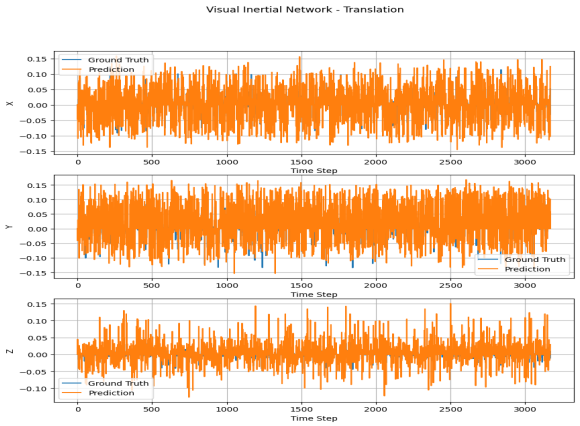

Fig. 20. Translation VI Fusion

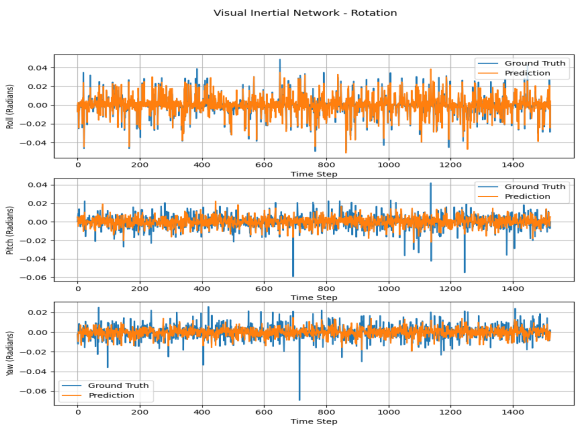

Fig. 21. Rotation VI Fusion

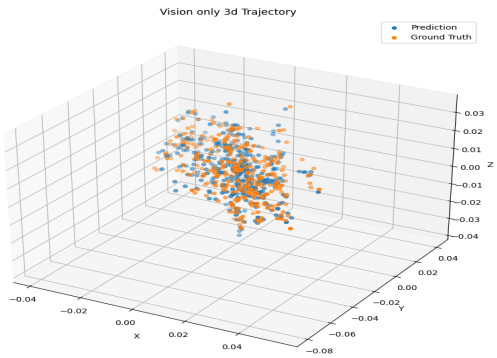

Fig. 22. 3D trajectory - Vision

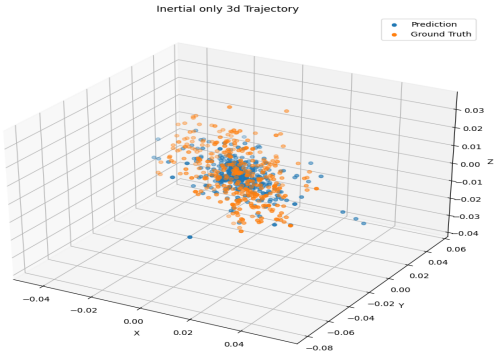

Fig. 23. 3D trajectory - Inertial

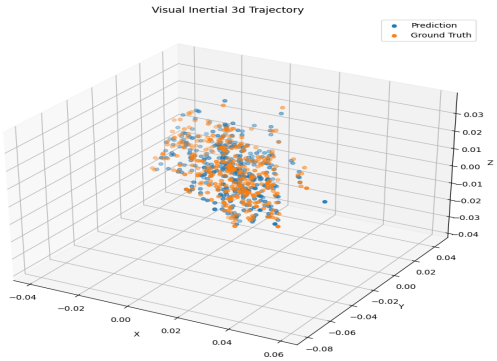

Fig. 24. 3D trajectory - VI Fusion