Computer Vision - Project 4: Deep and Un-Deep Visual Inertial Odometry (Phase 2)

Jesdin Raphael
Worcester Polytechnic Institute
Worcester, MA, USA
Computer Science
Email: jraphael@wpi.edu

Harsh Verma
Worcester Polytechnic Institute
Worcester, MA, USA
Robotics Engineering
Email: hverma@wpi.edu

Muhammad Sultan
Worcester Polytechnic Institute
Worcester, MA, USA
Robotics Engineering
Email: msultan@wpi.edu

***Abstract*—In this phase of the project we build three Deep Neural Networks to estimate the trajectory of a UAV, using synthetic data. The first network only used visual data, the second one only used inertial data and the third one used both. The first network had a CNN architecture, because the data included images, the second had LSTM architecture because it took inertial data which was time-series which LSTM handles quite well, and the third network was, naturally, a combination of both.**

I. Introduction

Unmanned aerial vehicles (UAVs) have seen a sharp rise in use in recent years for a variety of purposes, including defense, surveillance, and rescue operations. Robust and precise assessment of the UAV's position and orientation play a critical role in these applications' success. A common method for determining a UAV's pose using camera and inertial sensor measurements is called visual and inertial odometry (VIO). However, a number of issues, including sensor noise and calibration mistakes, can cause VIO to drift and become unstable. Deep learning techniques have been used to solve these problems by utilising neural networks' ability to learn strong feature representations and simulate intricate nonlinear interactions, hence improving VIO performance. This project showcases a deep learning-based VIO implementation. Using a deep neural network design that leverages the complementing information offered by various sensors, our method combines optical and inertial measurements. In particular, we process the visual measurements using a convolutional neural network (CNN) and the inertial measures using an LSTM network. The research shows how deep learning can be used to enhance VIO performance.

II. Data Generation

The data was generated by ourselves, [1], available on the PeAR website. Preceding this, we tried other toolboxes based on MATLAB and more, but none of them gave us the exact data we needed, so we ended up using the referenced simulated for our final data generation.

A. Environment

We placed a large image plane under the quadrotor, and applied an image texture to it. The quadrotor had a camera pointing towards this plane, and as the quadrotor moved around it would take images at fixed intervals. The IMU data was taken at a frequency of 1000 Hz and the images were taken at a frequency of 100 Hz. This visual and intertial data was then time-synced using time stamps.

We generated data for multiple trajectories, using this simulation, however the data was exact/calculated, and we needed simulated IMU data, so we used a MATLAB toolbox [2] to add noise to the data to make it as realistic as possible.

B. Organizing The Data

Once the data from MATLAB was obtained, we converted the required matrices, namely- state, gyroReading and accelReading. All these values were compiled into a csv file with 13 columns:

$$[x, y, z, q_x, q_y, q_z, a_x, a_y, a_z, g_x, g_y, g_z] \tag{1}$$

where first three values are translations in the respective frames, followed by orientation in terms of quaternions, and finally the IMU data as acceleration and gyroscope values.

C. Dataset

7 sequences were made of 2000 images and 20,000 IMU readings each. The models were trained on six of these sequences, and tested on one trajectory. Figure 1 is an illustration of the network. Each image was of size 320x240 pixels, and the images were resized to 160x120 and grayscaled before passing to the networks.

III. Neural Networks

As the project required, we built three neural networks- Vision, Inertial and a Combined one.

A. Pure Vision:

In order to use deep learning to estimate the camera pose using purely vision, we created a neural network with just two types of layers- convolutional and linear. To introduce nonlinearity to the network, we also used ReLU activation. Learning rate was set to 0.001 and the network was trained for 21 epochs.

1) Loss: : We started off with using the MSE loss, but soon realized that it was inappropriate for the use case. A hybrid loss function with MSE and Geodesic component was then implemented. ($q_i = [q_x, q_y, q_z, q_w]$)

Loss = MSE Loss For XYZ + Quaternion Distance

$$D_{quaternion} = cos^{-1}(2 * |q_1.q_2|) \tag{2}$$

B. Inertial

For the inertial model, we decided to go ahead with LSTM layers along with Fully Connected Layers, since LSTM are better at processing time-series data than Convolutional layers. The model was trained with learning rate of 0.0001, to avoid NaN loss values. Figure 2 presents the architecture for this model.

C. Combined

We combined both the models, the inertial and visual to optimize the image and pose inputs. The architecture is shown in the figure 3

IV. Results

A. RMSE ATE error

The RMSE ATE error was calculated to be **4.4539312**.

B. Predicted and Ground Truth Trajectory

A circular trajectory was plotted for ground truth and estimated trajectory by the model in figure 4 and 5

C. Conclusion

The results were not satisfactory for us. The ground truth and the predicted trajectories look similar in shape, however, the values if the coordinates are heavily off.

Scope for improvement: Better network architecture and loss functions. More experimentation with the architecture and types of layers is needed.

V. Research Problems:

Feature Selection and Tracking Optimization: The goal of feature selection and tracking optimisation is to minimise feature drift and maximise tracking efficiency while exploring new feature descriptors and tracking algorithms to improve robustness against occlusions, changes in lighting, and non-rigid scene deformations.

Scale Ambiguity and Drift Mitigation To reduce scale drift and preserve long-term pose consistency,

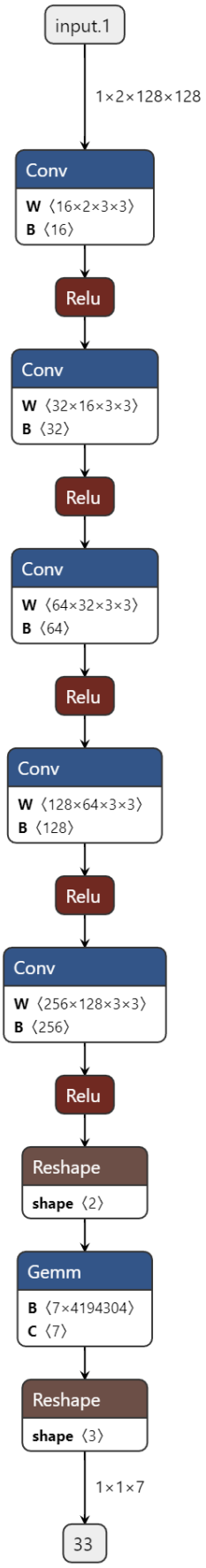

Fig. 1: Pure Vision Model Architecture

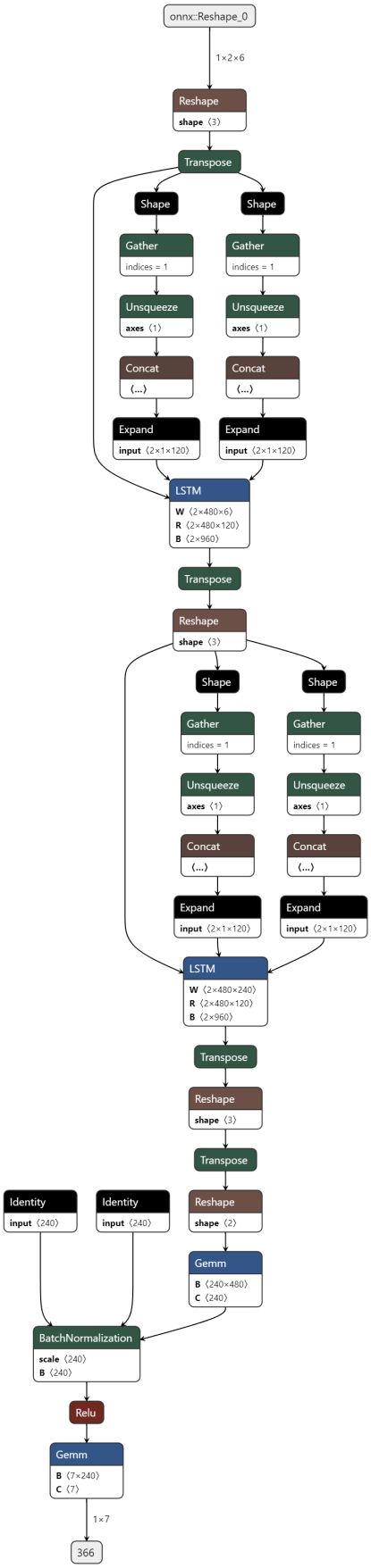

Fig. 2: Inertial Model Architecture

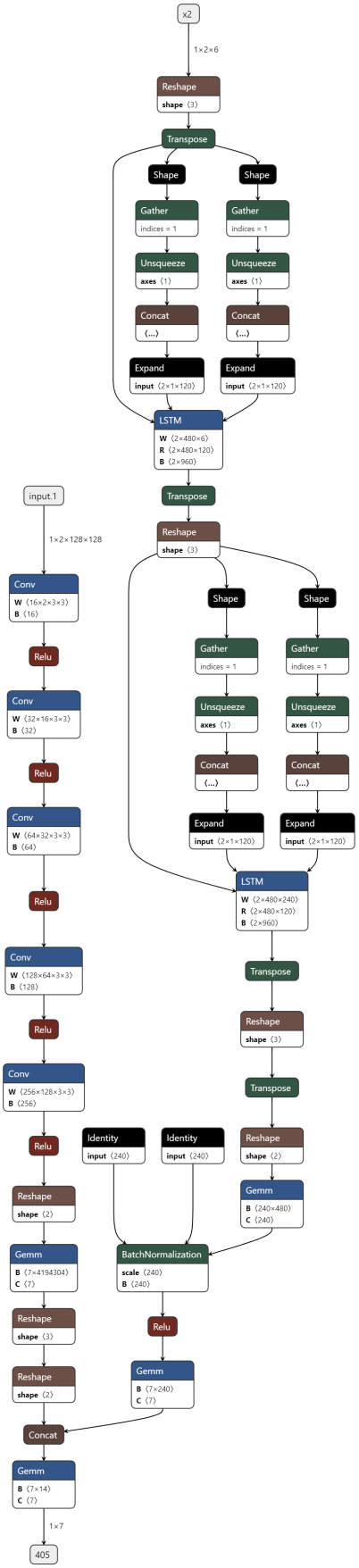

Fig. 3: Combined Model Architecture

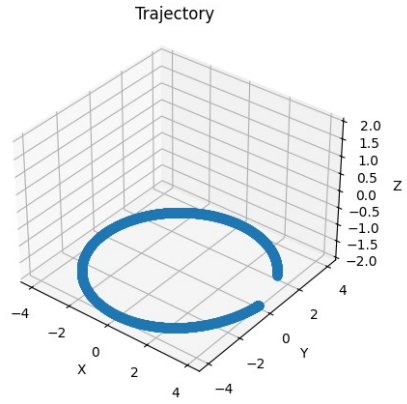

Fig. 4: Ground Truth Circular Trajectory

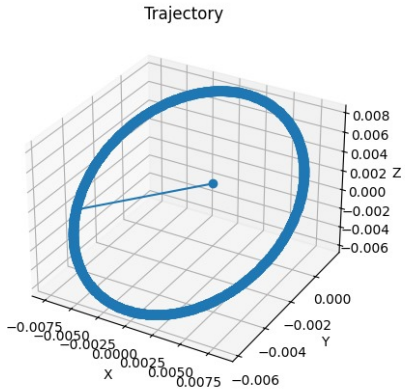

Fig. 5: Predicted Circular Trajectory

REFERENCES

[1] "RBE595-F02-ST – Hands-On Autonomous Aerial Robotics — rbe549.github.io," https://rbe549.github.io/rbe595/fall2023/proj/p0/, [Accessed 28-04-2024].

[2] "IMU simulation model - MATLAB — mathworks.com," https://www.mathworks.com/help/nav/ref/imusensor-system-object.html, [Accessed 28-04-2024].

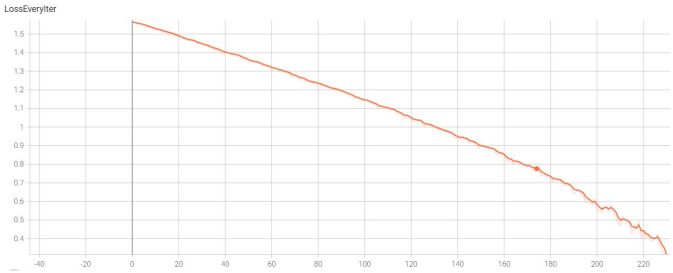

Fig. 6: Pure Vision - Training loss across iterations

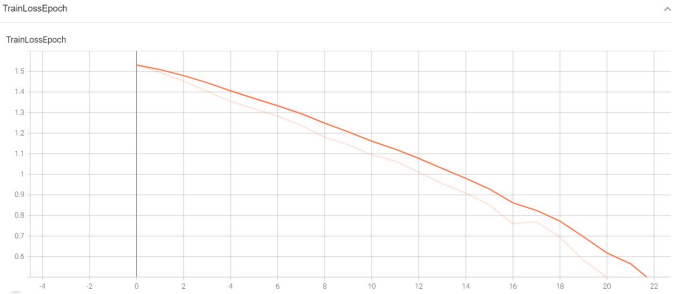

Fig. 7: Pure Vision - Training loss per epoch

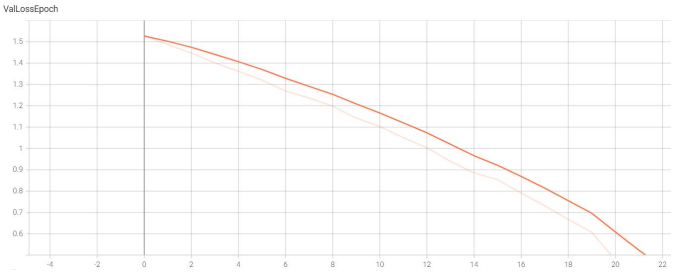

Fig. 8: Pure Vision - Validation Loss

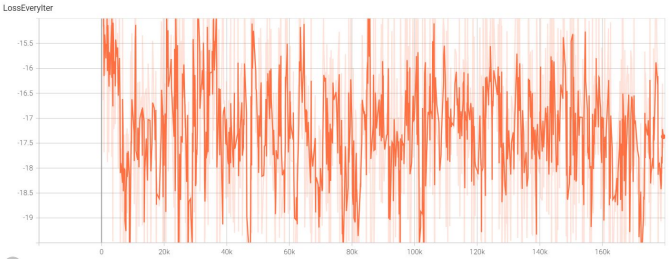

Fig. 9: Inertial - Training loss across iterations

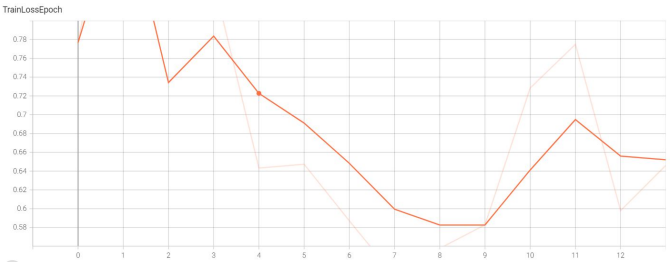

Fig. 10: Inertial - Training loss per epoch

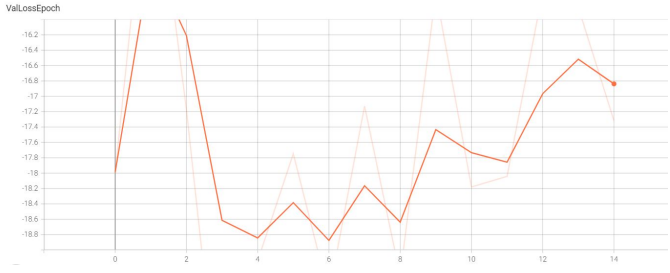

Fig. 11: Inertial - Validation Loss

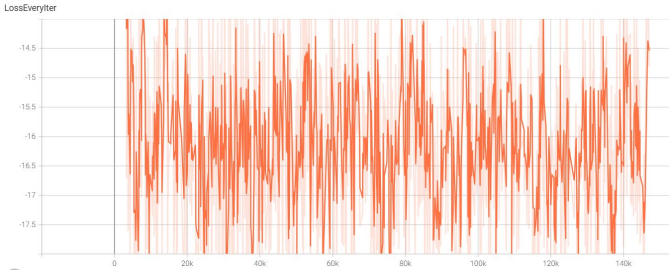

Fig. 12: Combined - Training loss across iterations

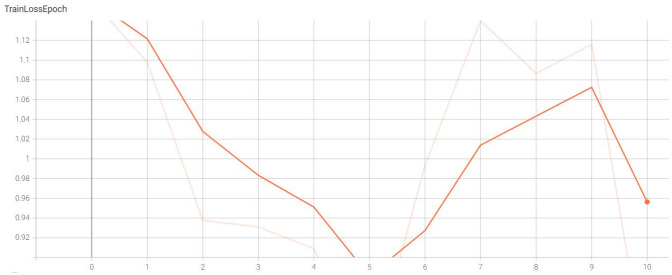

Fig. 13: Combined - Training loss per epoch

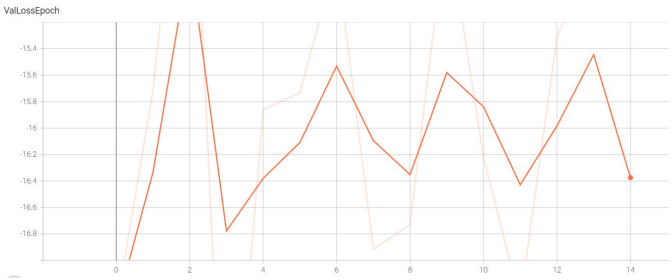

Fig. 14: Combined - Validation Loss