Deep Visual-Inertial Odometry RBE549 Project 4 Phase 2

Sumukh Porwal, Piyush Thapar, Sarah Semy
MS Robotics Engineering
Worcester Polytechnic Institute
Email: sporwal@wpi.edu, pthapar@wpi.edu, srsemy@wpi.edu

***Abstract*—Accurate state estimation is fundamental to robust robot navigation. This report explores deep-learning approaches to Visual–Inertial Odometry (VIO) by developing and comparing three neural architectures: (i) a vision-only network that infers motion directly from consecutive image frames, (ii) an inertial-only network that relies solely on raw IMU streams, and (iii) a hybrid visual–inertial fusion network that integrates both modalities within a unified framework. We detail the data pipelines, network designs, and training procedures on a suite of challenging trajectories.**

I. Introduction

As autonomous vehicles and robotic platforms proliferate, the demand for navigation systems that are both accurate and resilient has never been greater. Conventional solutions—relying on GPS signals or large sensor arrays—can be expensive and are easily disrupted by challenging environmental conditions. Visual-Inertial Odometry (VIO) offers a promising alternative by combining camera imagery with inertial measurements to continuously track a device's pose. However, standard VIO pipelines often lose reliability when operating in dynamic, low-light, or visually cluttered scenarios.

In this paper, we present a deep learning–enhanced VIO framework designed to boost both precision and robustness. Whereas traditional VIO methods struggle when image features degrade, our approach leverages neural networks to learn and extract richer, more discriminative representations, enabling more reliable pose estimates even under adverse conditions. Inspired by recent successes in fusing convolutional and sequence models for sensor data processing, our work integrates advances from computer vision and inertial analysis to address the inherent shortcomings of classical VIO.

II. Dataset Generation

A. Simulation Environment

All data were generated in *Blender 3.6 LTS* [**?**] using the Path-Tracing (Cycles) renderer to obtain photorealistic imagery and physically-consistent motion blur. A $100 \times 100\,\mathrm{m}$ planar mesh, textured with an $8\,\mathrm{k}$ high-frequency image, supplies dense visual features throughout the workspace. A six–degree-of-freedom quadrotor model carries a downward-facing pinhole camera (focal length $f = 450\,\mathrm{px}$, 640×480 resolution) rigidly co-located with a tri-axial inertial-measurement unit (IMU); thus the camera–IMU extrinsics satisfy $R_{CI} = I_3$ and $\mathbf{t}_{CI} = \mathbf{0}$.

Fig. 1: Reference Image for Visual Frames

B. Trajectory Design

Twelve $10\,\mathrm{s}$ trajectories ($\Delta t = 1\,\mathrm{ms}$) were scripted via Blender's *Animation Nodes* to cover a broad motion envelope (Fig. 2). Eight sequences are allocated to training and four to evaluation. Each path is parameterised by smooth polynomials, $\mathbf{p}(t) = [x(t),\, y(t),\, z(t)]^\top$, while respecting quadrotor thrust limits ($\|\mathbf{a}\| \le 6\,\mathrm{m/s^2}$). Attitude profiles are

$$\phi(t) = A_\phi \sin\!\big(2\pi t/T_\phi\big),\ \theta(t) = A_\theta \sin\!\big(2\pi t/T_\theta\big),\ \psi(t) = 0,$$

with amplitudes $(A_\phi, A_\theta) = (10^\circ, 10^\circ)$ and periods $(T_\phi, T_\theta) = (10, 10)\,\mathrm{s}$. At each frame Blender records ground-truth states $\big(\mathbf{p}, \dot{\mathbf{p}}, \mathbf{R}, \boldsymbol{\omega}\big)$.

C. Sensor Modelling

Ideal measurements: Ideal specific force $\mathbf{f}^*$ and angular rate $\boldsymbol{\omega}^*$ are computed from analytic derivatives:

$$\boldsymbol{\omega}^*(t) = \mathbf{R}^\top \dot{\mathbf{R}}, \quad \mathbf{f}^*(t) = \mathbf{R}^\top\big(\ddot{\mathbf{p}} - \mathbf{g}\big).$$

Noise and bias injection: To emulate a low-cost MEMS IMU, $(\mathbf{f}^*, \boldsymbol{\omega}^*)$ are routed through the *OysterSim* low-accuracy model [**?**]. Bias random walks ($\sigma_{b_a} = 1.0 \times 10^{-2}\,\mathrm{m/s^2}$, $\sigma_{b_\omega} = 1.0 \times 10^{-2}\,\mathrm{rad/s}$) are added, giving IMU packets at $1000\,\mathrm{Hz}$. Blender renders RGB frames at $100\,\mathrm{Hz}$; each image is therefore accompanied by 10 time-aligned IMU samples.

D. Ground-Truth Pose and Relative Motion

Absolute camera poses $(\mathbf{p}, \mathbf{q})$ are logged per render. Relative motion between frames $k-1$ and k is

$$\Delta\mathbf{q}_k = \mathbf{q}_{k-1}^{-1}\mathbf{q}_k, \quad \Delta\mathbf{p}_k = \mathbf{p}_k - \mathbf{p}_{k-1}$$

E. Data Organisation

Each sequence resides in `sequence_id/{images,imu}`. Tab. I details file layouts.

TABLE I: Dataset file formats

File	Columns (units)
`cam_poses.csv`	$t,\ x, y, z,\ q_w, q_x, q_y, q_z$
`imu.csv`	$t,\ a_x, a_y, a_z,\ \omega_x, \omega_y, \omega_z$
`rel_pose.csv`	$t,\ \Delta x, \Delta y, \Delta z,\ \Delta q_w, \Delta q_x, \Delta q_y, \Delta q_z$
`images/`	RGB PNGs named `frame_#.png`

The final corpus contains 16,000 image–IMU pairs with sub-millimetre ground truth, forming a controlled yet challenging benchmark for learning-based visual–inertial odometry.

III. Network Architectures

A. Deep Inertial Odometry Network

The proposed D-IO model—illustrated in Fig. 3—stacks **two bidirectional LSTM (Bi-LSTM) layers** [1] followed by a lightweight fully connected (FC) head:

- **Bi-LSTM 1**: input size 6, hidden size 256, 1 layer, bidirectional. Output dimensionality 512 (256×2).
- **Bi-LSTM 2**: input size 512, hidden size 256, 1 layer, bidirectional. Output dimensionality 512.
- **FC 1**: $\mathbb{R}^{512} \to \mathbb{R}^{128}$, ReLU activation.
- **FC 2**: $\mathbb{R}^{128} \to \mathbb{R}^{7}$ (3-D translation + 4-D quaternion).

Only the final hidden state of the second Bi-LSTM layer is forwarded to the FC head, enforcing the network to compress the entire temporal context into a single latent vector.

We empirically set $N = 11$ IMU frames, corresponding to a 0.01 s window when the IMU is sampled at 1000 Hz (our default setting). The choice of a short window balances two competing objectives: (i) capturing sufficient motion dynamics and (ii) limiting drift accumulation within the integration horizon.

B. Deep Visual Odometry Network

The Deep-Visual Odometry (D-VO) model, denoted `VO_Net`, follows the following design: the two input frames are processed by a shared-weight feature extractor, concatenated, and then split into translation and rotation heads (Fig. 4).

- **Input tensors**: two RGB images $\mathbf{I}_1, \mathbf{I}_2 \in \mathbb{R}^{3\times H\times W}$, intensity-normalised to $[-1, 1]$.
- **Shared feature extractor** (*per frame*)
  1) **BasicConvEncoder**
     - 7×7 conv, 32 ch., stride 2
     - 5×5 conv, 64 ch., stride 2
     - 3×3 conv, 128 ch., stride 2
     - 3×3 conv, 256 ch., stride 2

     All layers use BatchNorm and ReLU; the spatial resolution is reduced by a factor 16.
  2) **POLAUpdate** block one 3×3 conv → ReLU, acting as a lightweight feature-refinement stage (a proxy for patch-overlapping self-attention).

  Output: $\mathbf{F}_1, \mathbf{F}_2 \in \mathbb{R}^{256\times \frac{H}{16}\times \frac{W}{16}}$.
- **Feature fusion**: channel-wise concatenation $\mathbf{F} = \big[\mathbf{F}_1\,;\,\mathbf{F}_2\big] \in \mathbb{R}^{512\times \frac{H}{16}\times \frac{W}{16}}$.
- **Pose-specific heads**
  - **Translation branch**
    1) 3×3 conv $(512 \to 256)$
    2) Adaptive MaxPool $\to \mathbb{R}^{256}$
    3) FC $256 \to 128$ (ReLU)
    4) FC $128 \to 3$ $\quad(\Delta\mathbf{p})$
  - **Rotation branch** identical topology but the final layer outputs a 4-D unit quaternion $\Delta\mathbf{q}$.
- **Output vector**: $\big[\Delta\mathbf{p}\,;\,\Delta\mathbf{q}\big] \in \mathbb{R}^7$.

C. Transformer-Based Visual–Inertial Odometry

- **Inputs:**
  - *Image pair:* $\mathbf{I}^{\pm} \in \mathbb{R}^{B\times 6\times 128\times 128}$ (two RGB frames stacked on channel axis).
  - *IMU window:* $\mathcal{I} \in \mathbb{R}^{B\times 11\times 6}$ (ax,ay,az,wx,wy,wz over 11 timesteps).
- **Visual encoder:**
  1) **ResNet-18 backbone:** [2] instantiate with no pre-trained weights.
  2) **Conv1 modification:** replace first layer with

     $$\text{Conv2d}(6 \to 64,\ 7\times 7,\ \text{stride} = 2,\ \text{pad} = 3).$$

  3) Remove the global pooling and FC head, yielding feature map $(B, 512, 1, 1)$.
  4) **Projection:** flatten to $(B, 512)$ and apply

     $$\text{Linear}(512 \to \text{embed\_dim} = 512)$$

     to produce a single *visual token* of shape $(B, 1, 512)$.
- **IMU encoder:**
  1) **LSTM:** single-layer, unidirectional with

     $$\text{input\_size} = 6, \quad \text{hidden\_size} = \text{embed\_dim} = 512$$

  2) Processes $(B, 11, 6) \to (B, 11, 512)$, yielding a sequence of 11 *IMU tokens*.
- **Positional embeddings:**

  $$\mathbf{P} \in \mathbb{R}^{1\times(1+11)\times 512}$$

  learned and added to the concatenated token sequence to encode order.
- **Transformer fusion:** [3]
  - Stack visual + IMU tokens to $(B, 12, 512)$, add $\mathbf{P}$.

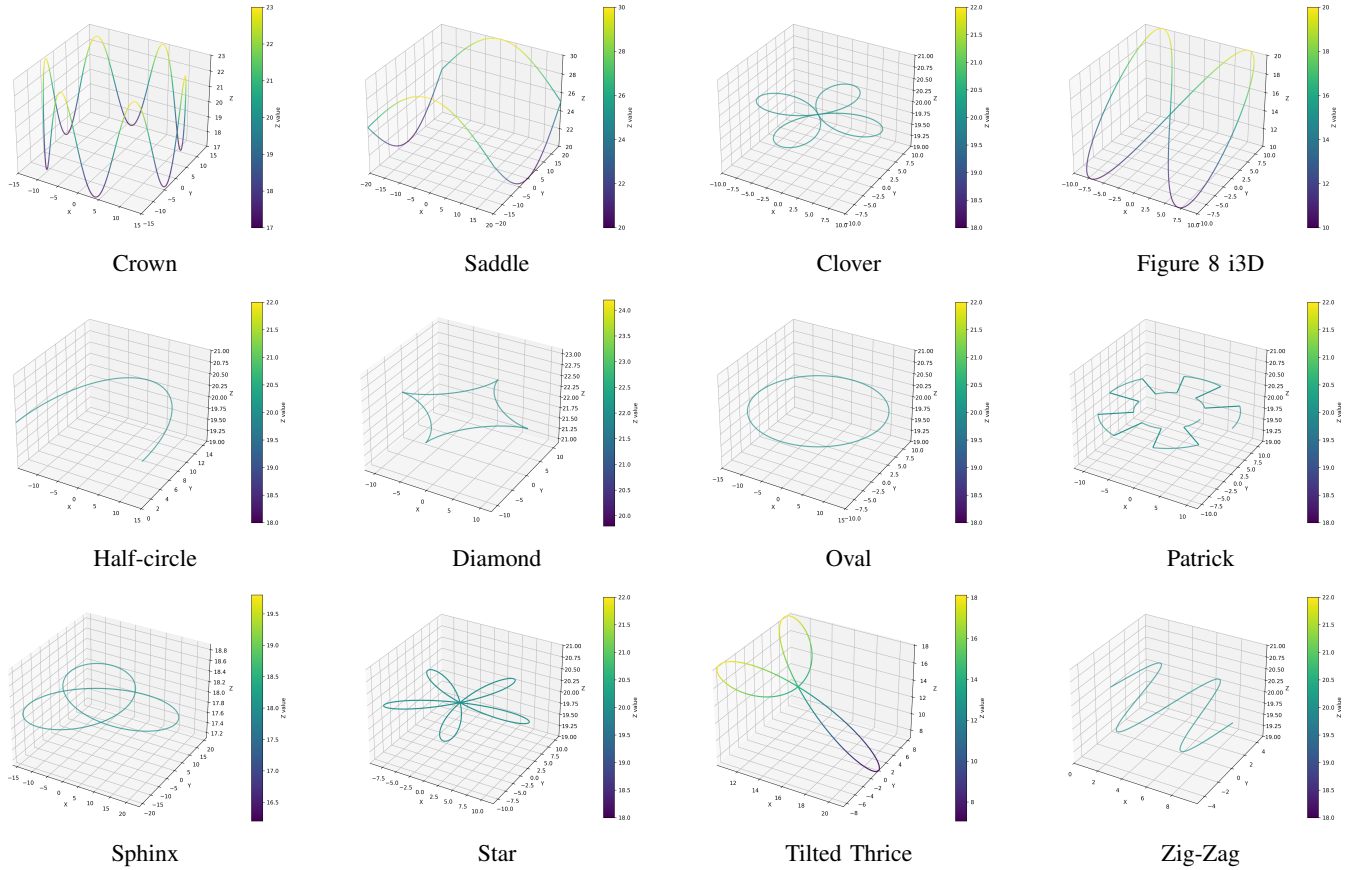

Fig. 2: Train and test trajectories used for dataset generation. Numbers correspond to trajectory IDs discussed in Section II.

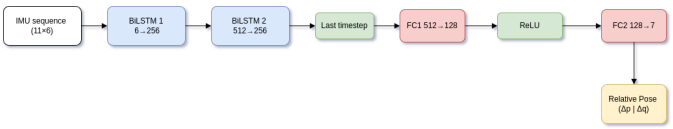

Fig. 3: Deep Inertial Odometry architecture.

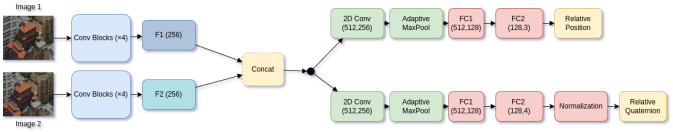

Fig. 4: Deep-Visual Odometry architecture.

– **TransformerEncoder**: 4 layers, 8 heads, feed-forward dim $= 4 \times 512$, batch-first.
– Take the first (visual) output token, yielding $(B, 512)$ as the fused representation.

- **Regression head:**

1) MLP:

$$512 \xrightarrow{\text{Linear}} 256 \xrightarrow{\text{ReLU}} 256 \xrightarrow{\text{Linear}} 7.$$

2) Split into translation $\Delta\mathbf{p} \in \mathbb{R}^3$ and quaternion $\Delta\mathbf{q} \in \mathbb{R}^4$.

3) *Quaternion normalisation:* enforce $\|\Delta\mathbf{q}\|_2 = 1$ via F.normalize.

- **Output:** concatenated $(\Delta x, \Delta y, \Delta z, \Delta q_w, \Delta q_x, \Delta q_y, \Delta q_z) \in \mathbb{R}^{B\times 7}$.

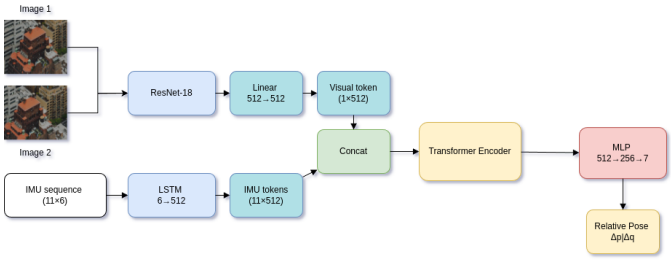

Fig. 5: Visual–Inertial Odometry architecture.

IV. Training Details

All three models—**D-VO**, **D-IO**, and **V-IO**—are optimised under the same data-sampling schedule, loss function, and optimisation hyper-parameters.

Mini-batch composition: Every training sample represents the motion between two consecutive keyframes spaced $\Delta t = 0.01\,\text{s}$ (i.e.,ten IMU ticks).

- **VO**: stacked RGB pair $\left[\mathbf{I}_t, \mathbf{I}_{t+\Delta t}\right] \in \mathbb{R}^{6\times H\times W}$.

- **IO**: IMU window $\mathcal{I}_{t:t+N-1} \in \mathbb{R}^{N\times 6}$ with $N = 11$.
- **VIO**: *both* inputs above, synchronised.

Ground-truth supervision is the 7-DoF relative pose $\Delta\mathbf{T} = [\Delta\mathbf{p},\ \Delta\mathbf{q}] \in \mathbb{R}^{3+4}$ obtained by differencing the simulator states.

Loss function: A unified translation–rotation loss is applied:

$$\mathcal{L} = \lambda_p \|\hat{\mathbf{p}} - \mathbf{p}\|_2^2 + \lambda_q \left[1 - \cos\angle(\hat{\mathbf{q}}, \mathbf{q})\right], \qquad (1)$$

where $\cos\angle(\hat{\mathbf{q}}, \mathbf{q})$ is the cosine similarity between unit quaternions. Weights are fixed to $\lambda_p = 0.6$ and $\lambda_q = 0.4$, biasing the network toward accurate translation while still regularising orientation.

Optimisation: Adam ($\beta_1 = 0.9$, $\beta_2 = 0.999$) with an initial learning rate of 10^{-3} is used throughout. Each model is trained for 50 epochs with a batch size of 16 on a single RTX 4070 GPU. Datasets from eight simulated trajectories are concatenated via PyTorch's `ConcatDataset` wrapper to enhance motion diversity, and images are resized to 128×128 pixels.

Quaternion post-processing: The final FC layer of every network is followed by a *Quaternion-Normalisation* module that projects the predicted four-vector onto $\mathrm{SO}(3)$; this stabilises training and obviates any explicit unit-norm constraint in (1).

V. Results

1) Inertial Odometry network:
   - RMSE on Training trajectory - 0.036244
   - RMSE on Testing trajectory - 0.088381
   - Refer figure 6 for Loss VS Epochs for Inertial Odometry Model.
   - Refer figure 7 for Predicted VS Ground Truth on Training Dataset for Inertial Odometry Model.
   - Refer figure 8 for Predicted VS Ground Truth on Testing Dataset for Inertial Odometry Model.
2) Visual Odometry network:
   - RMSE on Training trajectory - 0.020982
   - RMSE on Testing trajectory - 0.059533
   - Refer figure 9 for Loss VS Epochs for Visual Odometry Model.
   - Refer figure 10 for Predicted VS Ground Truth on Training Dataset for Visual Odometry Model.
   - Refer figure 11 for Predicted VS Ground Truth on Testing Dataset for Visual Odometry Model.
3) Visual-Inertial Odometry network:
   - RMSE on Training trajectory - 0.031114
   - RMSE on Testing trajectory - 0.039211
   - Refer figure 12 for Loss VS Epochs for Visual-Inertial Odometry Model.
   - Refer figure 13 for Predicted VS Ground Truth on Training Dataset for Visual-Inertial Odometry Model.
   - Refer figure 14 for Predicted VS Ground Truth on Testing Dataset for Visual-Inertial Odometry Model.

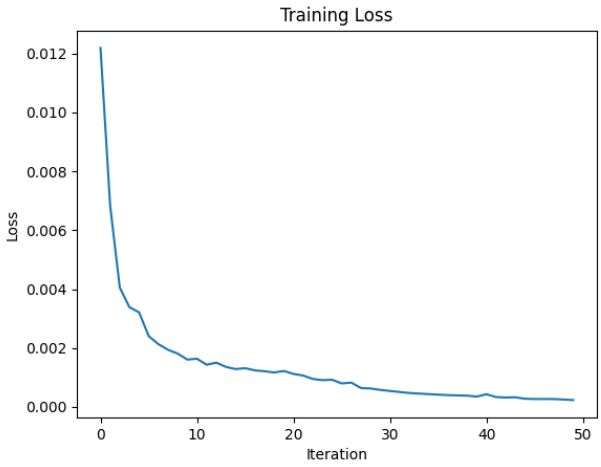

Fig. 6: Loss VS Epochs for Inertial Odometry Model.

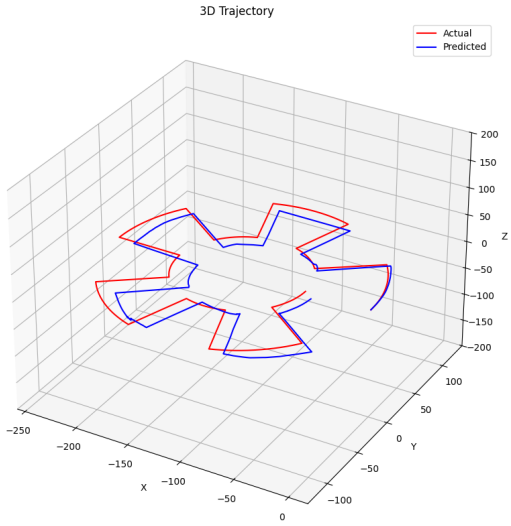

Fig. 7: Predicted VS Ground Truth on Training Dataset for Inertial Odometry Model.

References

[1] Zhiheng Huang, Wei Xu, and Kai Yu. Bidirectional lstm-crf models for sequence tagging, 2015.
[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.
[3] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023.

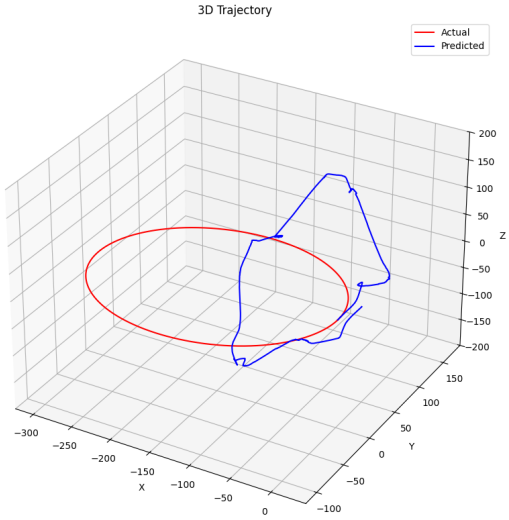

Fig. 8: Predicted VS Ground Truth on Testing Dataset for Inertial Odometry Model.

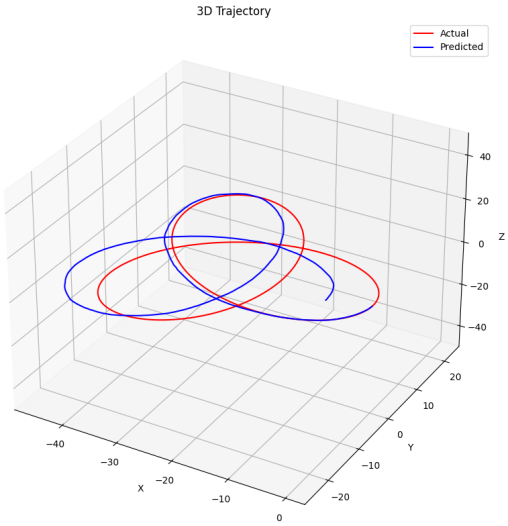

Fig. 10: Predicted VS Ground Truth on Training Dataset for Visual Odometry Model.

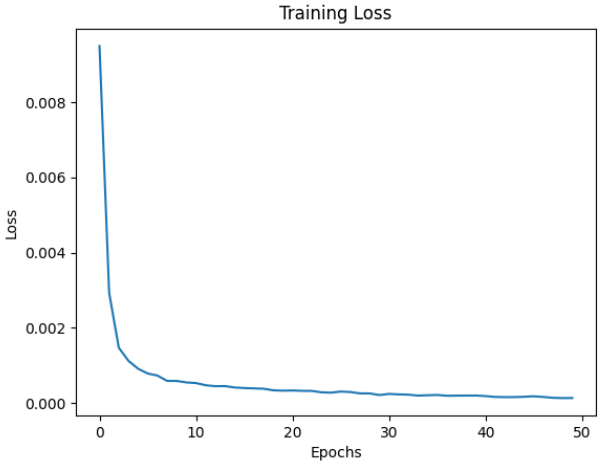

Fig. 9: Loss VS Epochs for Visual Odometry Model.

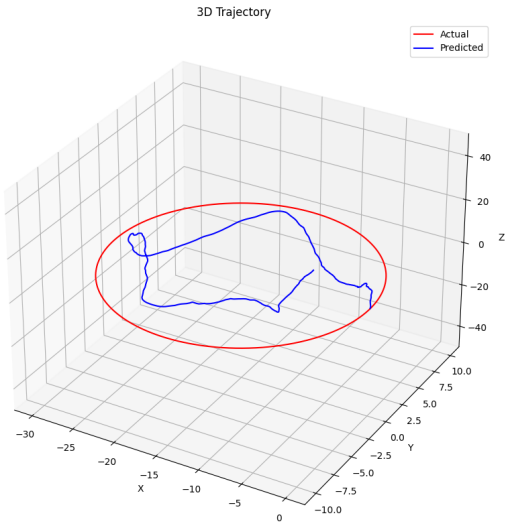

Fig. 11: Predicted VS Ground Truth on Testing Dataset for Visual Odometry Model.

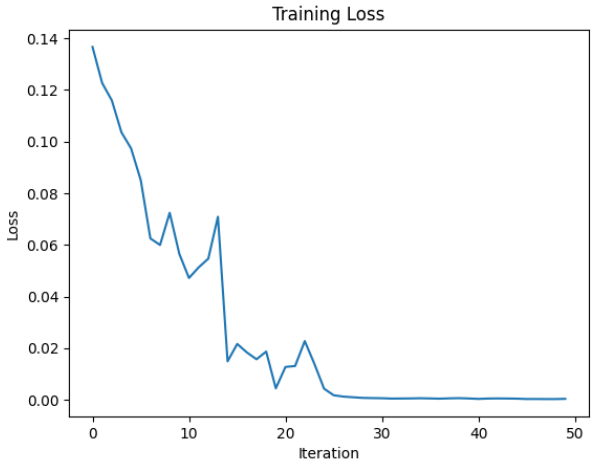

Fig. 12: Loss VS Epochs for Visual-Inertial Odometry Model.

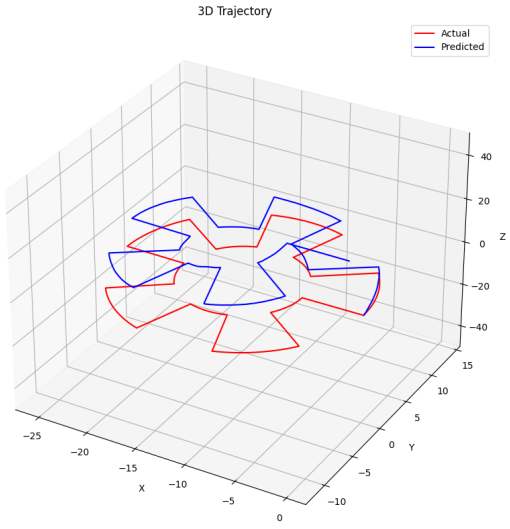

Fig. 13: Predicted VS Ground Truth on Training Dataset for Visual-Inertial Odometry Model.

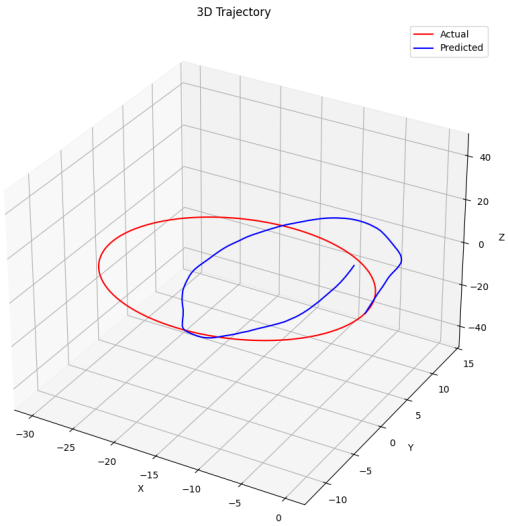

Fig. 14: Predicted VS Ground Truth on Testing Dataset for Visual-Inertial Odometry Model.