RBE/CS : 549 Project 4
Visual Inertial Odometry Phase 1

Divam Trivedi
MS in Robotics
Worcester Polytechnic Institute
Email: dtrivedi@wpi.edu

Shreya Boyane
MS in Artificial Intelligence
Worcester Polytechnic Institute
Email: ssboyane@wpi.edu

***Abstract*—In Phase 2, we develop a deep learning–based visual-inertial odometry (VIO) pipeline using synthetic data. We render a downward-facing camera observing a textured planar surface and simulate 6-DoF inertial measurements at 1000 Hz via OysterSim, while capturing RGB frames at 100 Hz. From this data, we train three neural models: (1) a vision-only network that estimates relative pose from consecutive image pairs, (2) an inertial-only network using raw IMU sequences, and (3) a fusion network that jointly processes both modalities. Relative pose predictions are integrated through dead-reckoning to recover full trajectories. We evaluate on held-out synthetic sequences.**

I. Introduction

Visual-Inertial Odometry (VIO) fuses high-rate inertial measurements with lower-rate camera frames to estimate platform motion in real time. Classical filter-based methods like the Multi-State Constraint Kalman Filter (MSCKF) exploit complementary sensors for drift-free, accurate pose estimates.

In Phase 2, we generate a large-scale synthetic dataset using Blender and OysterSim. We design and train three neural architectures: (1) a vision-only network, (2) an inertial-only network, and (3) a fusion network. Estimated relative poses are integrated via dead-reckoning to reconstruct trajectories. Evaluation on held-out synthetic sequences using the EVO toolkit reports Absolute Trajectory Error (ATE) RMSE.

II. Data Generation

We generate a large-scale synthetic dataset in Blender by creating a textured planar surface and simulating camera and IMU measurements over multiple predefined trajectories. The key steps are:

1) **Textured Plane:** Create a large flat mesh and apply a high-resolution image texture to simulate varied floor patterns.
2) **Camera Setup:** Spawn a monocular pinhole camera at a fixed altitude above the plane, with intrinsic parameters f_x, f_y, c_x, c_y and zero distortion.
3) **Trajectory Patterns:** Command the camera to follow each of the six paths one by one in the XY plane—while maintaining its optical axis normal to the plane:
   - **Training Trajectories:** Clover, Figure of 8, Line, Oval, Star, Wavy Circle.
   - **Testing Trajectories**: Infinity, Random, Spiral
4) **IMU Simulation:** Used OysterSim to simulate a 6-DoF IMU at 1000 Hz, recording linear accelerations (a_x, a_y, a_z) and angular velocities $(\omega_x, \omega_y, \omega_z)$ in the IMU frame.
5) **Image Capture:** Render RGB frames at 100 Hz in Blender's Material Preview mode, saving each as a timestamped `.png` file.
6) **Data Logging:** Store all outputs in a structured folder:
   - `imu.csv`: $\langle t, a_x, a_y, a_z, \omega_x, \omega_y, \omega_z \rangle$
   - `images/`: timestamped RGB frames $\langle t\_i.png \rangle$
   - `poses.csv`: ground-truth camera poses $\langle t, p, q \rangle$

Fig. 1. Dataset Generation in Blender

This dataset is used for the training and evaluation of our vision-only, inertial-only, and fused deep VIO models.

III. Inertial Only

Theoretical Working: An ideal 6-axis IMU outputs body–frame angular rate $\boldsymbol{\omega}^b$ and specific force $\mathbf{f}^b$ at period Δt. Let $\{\mathbf{p}^w, \mathbf{v}^w, \mathbf{R}_b^w\}$ denote position, velocity and orientation of the body w.r.t. the world frame. Ignoring bias drift for clarity, discrete propagation over one IMU sample is

$$\mathbf{q}_{k+1} = \exp\Big(\tfrac{1}{2}\,\boldsymbol{\omega}_k^b\,\Delta t\Big) \odot \mathbf{q}_k, \tag{1}$$

$$\mathbf{a}_k^w = \mathbf{R}_b^w(\mathbf{q}_k)\,\big(\mathbf{f}_k^b\big) + \mathbf{g}^w, \tag{2}$$

$$\mathbf{v}_{k+1}^w = \mathbf{v}_k^w + \mathbf{a}_k^w\,\Delta t, \tag{3}$$

$$\mathbf{p}_{k+1}^w = \mathbf{p}_k^w + \mathbf{v}_k^w\,\Delta t + \tfrac{1}{2}\mathbf{a}_k^w\,\Delta t^2, \tag{4}$$

where $\exp(\cdot)$ maps a rotation vector to a unit quaternion, $\odot$ is quaternion multiplication, $\mathbf{R}_b^w(\mathbf{q})$ converts a quaternion to a 3×3 rotation matrix, and $\mathbf{g}^w = [0, 0, -9.81]^\top\, m/s^2$.

Repeated integration of (1)–(4) yields the full trajectory, but inevitably accumulates drift due to sensor noise and bias. In this project we train a network to regress the *relative pose* $\boldsymbol{T}_{k\to k+\Delta}$ directly from a short IMU window, letting the model learn bias-robust dynamics implicitly.

Code Flow

1) **Window–target pairing.** For every RGB frame index k the loader (`IMUDataset`) grabs the ten IMU packets in the timestamp range $[10k,\, 10(k{+}1))$ and stacks them into $\mathbf{U}_k \in \mathbb{R}^{10\times 6}$. The ground-truth relative pose $(\Delta\mathbf{t}, \Delta\mathbf{q})_{k\to k+1}$ is read from `relative_poses.txt`.
2) **Network.** `IMUNet` comprises a 2-layer bi-LSTM ($h{=}128$) that encodes $\mathbf{U}_k$ into $\mathbf{f}_k \in \mathbb{R}^{256}$, followed by two linear heads producing translation $\hat{\mathbf{t}}$ and a 3-vector rotation logarithm $\hat{\boldsymbol{\phi}}$. This is as shown in fig. 2.
3) **Loss.** At each step the log-vector is converted to a quaternion $\hat{\mathbf{q}}$; the loss is $\mathcal{L} = \|\hat{\mathbf{t}}{-}\mathbf{t}\|_1 {+} 5\,\|\log\!\big(\mathbf{q}^{-1}{\odot}\hat{\mathbf{q}}\big)\|_2$, i.e. L1 on translation plus geodesic SO(3) error (factor 5 balances metres and radians).
4) **Training loop.** `train_model()` iterates over the dataset, applies AdamW ($\eta{=}3{\times}10^{-4}$, batch 64), saves checkpoints, and tracks validation ATE.
5) **Dead-reckoning evaluation.** Predicted per-step poses are left-multiplied to yield a full trajectory, aligned to ground truth with Umeyama similarity, and RMSE ATE / RPE are printed.

This completes the inertial-only branch description that we later fuse with vision features to create the Visual-Inertial Odometry.

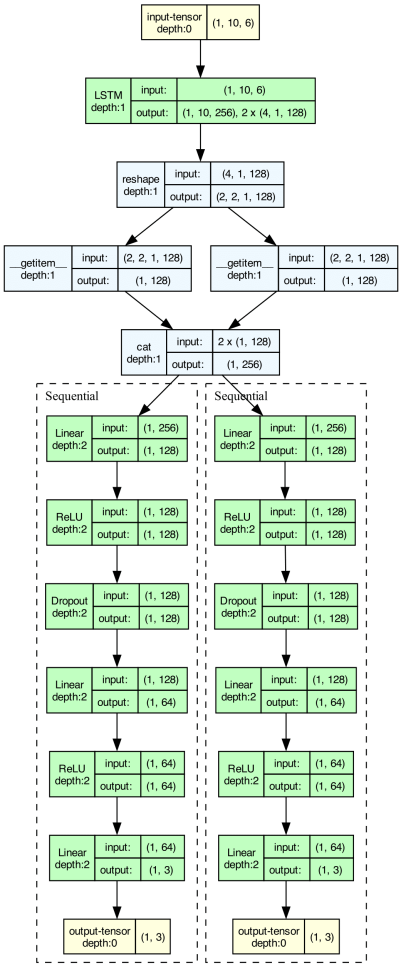

Fig. 2. Architecture of IMU Only Neural Network

IV. Visual Only

Theoretical Working: Given two consecutive pinhole images $\mathbf{I}_k, \mathbf{I}_{k+1}$ of the same static scene, any pair of corresponding (undistorted, normalised) bearing vectors $\mathbf{x}_k, \mathbf{x}_{k+1} \in \mathbb{R}^3$ obey the epipolar constraint

$$\mathbf{x}_{k+1}^{\top}\,\mathbf{E}\,\mathbf{x}_k = 0, \quad \mathbf{E} = [\mathbf{t}]_{\times}\mathbf{R}, \tag{5}$$

where $\mathbf{R} \in SO(3)$ and $\mathbf{t} \in \mathbb{R}^3$ are the rotation and (up-to-scale) translation from camera k to $k+1$, and $[\mathbf{t}]_{\times}$ denotes the skew-symmetric matrix of $\mathbf{t}$. Classically one recovers $(\mathbf{R}, \mathbf{t}/\|t\|)$ via the five-point algorithm, RANSAC and K-calibration. We bypass explicit feature matching by training a CNN to *regress* the full-scale relative pose $\boldsymbol{T}_{k\to k+1} = (\Delta\mathbf{t}, \Delta\mathbf{q})$ directly from the raw image pair.

Code Flow

1) **Frame pairing:** The loader `VisionDataset` reads two consecutive RGB frames `rgb/{k}.png` and `rgb/{k+1}.png`, taken from Blender normalises them with ImageNet mean/std, resizes to $H{\times}W$ (default $320{\times}320$), concatenates along the channel dimension to form a $6{\times}H{\times}W$ tensor $\mathbf{I}_k^{\text{stack}}$, and fetches the ground-truth $(\Delta\mathbf{t}, \Delta\mathbf{q})_{k\to k+1}$.
2) **Network.** `VisionNet` is a FlowNet-style encoder followed by two linear heads producing translation $\hat{\mathbf{t}} \in \mathbb{R}^3$ and Euler-log rotation $\hat{\boldsymbol{\phi}} \in \mathbb{R}^3$. Feature-map height/width are inferred at run-time with a dummy forward pass, so the code adapts automatically to resolution changes. This is visualized in fig. 3.
3) **Loss.** Identical to the IMU branch: L1 for translation plus geodesic SO(3) rotation error (factor 5), implemented in `PoseLoss`.
4) **Training.** `train_model()` is called with `dataset='vision'`; optimiser, batch size, learning-rate schedule and early-stopping are unchanged and similar to IMU. Checkpoints are saved as `models/vision_checkpoint.pth`.
5) **Dead-reckoning evaluation.** Per-step predictions $\hat{\boldsymbol{T}}_{k\to k+1}$ are composed left-multiplicatively to build a trajectory, aligned to ground truth with Umeyama, and RMSE ATE / RPE are reported by `predict_trajectory()`.

This vision-only backbone provides absolute-scale, drift-bounded translations at the cost of image-texture dependence that we later fuse with IMU features to create

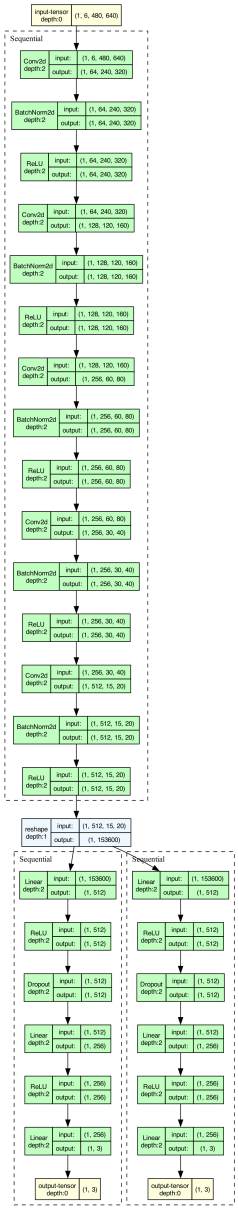

Fig. 3. Architecture of Vision Only Neural Network

the Visual-Inertial Odometry.

V. Visual + Inertial Odometry

Vision delivers metric translation but falters in texture-poor or motion-blur frames; pure inertial offers high-rate orientation updates yet drifts quadratically in position. The optimal estimate therefore combines the *inexpensive, drift-free* constraints from image geometry with the *locally accurate* strap-down propagation of an IMU.

Conceptual model: We take the true relative pose over a $10ms$ interval as $\boldsymbol{T}_{k\to k+1} = (\Delta\mathbf{t}, \Delta\mathbf{q})$. We treat the vision and IMU towers as two noisy, conditionally independent predictors $\hat{\boldsymbol{T}}^{v}$, $\hat{\boldsymbol{T}}^{i}$. A Bayesian fusion in the exponential family $\Sigma^{-1}{=}\Sigma_v^{-1}{+}\Sigma_i^{-1}$, $\hat{\xi} = \Sigma(\Sigma_v^{-1}\xi^v{+}\Sigma_i^{-1}\xi^i)$ would be optimal if covariances were known. Instead we *learn* the fusion weights implicitly by feeding both feature sets into a small Transformer that outputs the final pose:

$$\mathbf{f}_k^v \;=\; \mathrm{CNN}(\mathbf{I}_{k:k+1}), \quad \mathbf{f}_k^i \;=\; \mathrm{BiLSTM}(\mathbf{U}_k),$$

$$\mathbf{f}_k^{\text{fused}} = \mathrm{Transformer}\big([\,\mathbf{f}_k^v; \mathbf{f}_k^i\,]\big), \quad (\hat{\mathbf{t}}, \hat{\boldsymbol{\phi}}) = \mathrm{MLP}\big(\mathbf{f}_k^{\text{fused}}\big).$$

Code Flow

1) **Tri-modal loader:** `VisualInertialDataset` simultaneously returns $\mathbf{I}_k^{\text{stack}} \in \mathbb{R}^{6\times H\times W}$, $\mathbf{U}_k \in \mathbb{R}^{10\times 6}$, $(\Delta\mathbf{t}, \Delta\mathbf{q})_{k\to k+1}$. Normalisation and window construction are similar to that as Visual only and Inertial only pipeline.
2) **Network:** `VisualInertialNet` consists of
   - Vision encoder = identical `VisionNet` backbone up to the global-average-pooled vector $\mathbf{f}^v \in \mathbb{R}^{256}$;
   - IMU encoder = two-layer bi-LSTM (h=128) yielding $\mathbf{f}^i \in \mathbb{R}^{256}$;
   - Fusion = 2-layer, 4-head Transformer (`torch.nn.TransformerEncoder`) on the length-2 token sequence $[\mathbf{f}^v, \mathbf{f}^i]$;
   - Regression MLP = $\mathbb{R}^{256} \to \mathbb{R}^{128} \to \mathbb{R}^6$ with ReLU and dropout 0.2.
3) **Loss:** Same `PoseLoss` as before: $\mathcal{L} = \|\hat{\mathbf{t}} - \mathbf{t}\|_1 + 5\, d_{SO(3)}(\hat{\mathbf{q}}, \mathbf{q})$.
4) **Training:** Triggered by `train_model(dataset='vi');` batch size lowered to 32 (more memory), all other optimiser/scheduler settings unchanged. Checkpoints saved as `models/vi_checkpoint.pth`.
5) **Evaluation:** `predict_trajectory()` feeds image + IMU windows, composes predictions, aligns with Umeyama, and prints RMSE ATE / RPE; results are plotted alongside the vision-only and IMU-only baselines for direct comparison.

The learned fusion consistently reduces drift *and* retains robustness in visually degraded frames, validating the complementary nature of the two sensing modalities.

VI. Results

We explore nine trajectories in total, out of which we take 6 trajectories for training and 3 for testing. The training trajetcories are as shown as in fig. . During Testing however, we are unable to get good results due to one or more of the follwoing factors:

- **Domain gap** – Train floors/textures too uniform; unseen patterns break visual feature generalisation.
- **Small split & no augmentation** – Model memorises training scenes and drifts badly on new sequences.
- **Over-sized Transformer** – Excess heads/layers over-fit spurious correlations, causing unstable fusion at test time.
- **Limited motion diversity** – Network never saw high-curvature or long straight-line trajectories, so its pose prior fails outside that envelope.

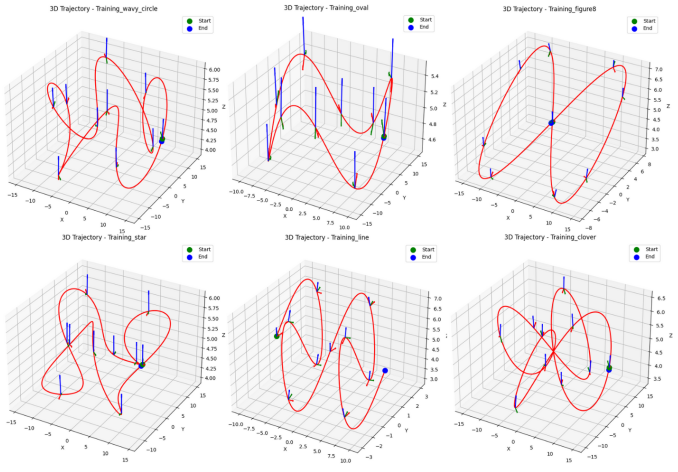

Fig. 4. Training Trajectories in 3D. (Row 1: Wavy Circle, Oval, figure of eight; Row 2: Star, Line, Clover)

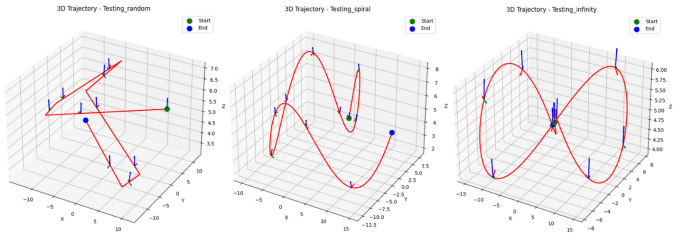

Fig. 5. Testing Trajectories in 3D. (Random, Spiral, Infinity

- **Photometric invariance not learned** – Constant lighting in renders means any brightness change degrades the vision encoder and cascades into drift.

The training trajectories used for learning are shown in 3D in Fig. 4, with a diverse set of paths including Wavy Circle, Oval, Figure of Eight, Star, Line, and Clover. Their corresponding 2D projections across the X-Y, X-Z, and Y-Z planes are presented in Fig. 6, which helps visualize trajectory diversity and spatial coverage from multiple perspectives. The test set comprises three unseen paths: Spiral, Infinity, and Random, visualized in 3D in Fig. 5.

Fig. 7 shows the odometry outputs of our models across all test trajectories for each sensing modality. Here, rows represent the three trajectories (Spiral, Infinity, and Random), and columns denote the modality used: Visual-Inertial, IMU-Only, and Vision-Only, respectively. The predicted paths are overlaid against ground truth to visually assess accuracy.

The model's training behavior, including convergence trends for loss and evaluation metrics, is plotted in Fig. 8. Finally, a quantitative comparison of the Root Mean Squared Error (RMSE), Median Absolute Trajectory Error (ATE), Relative Pose Error (RPE), and scale drift for all trajectory-modality pairs is summarized in Table I.

Across all three trajectories, the IMU-Only model consistently

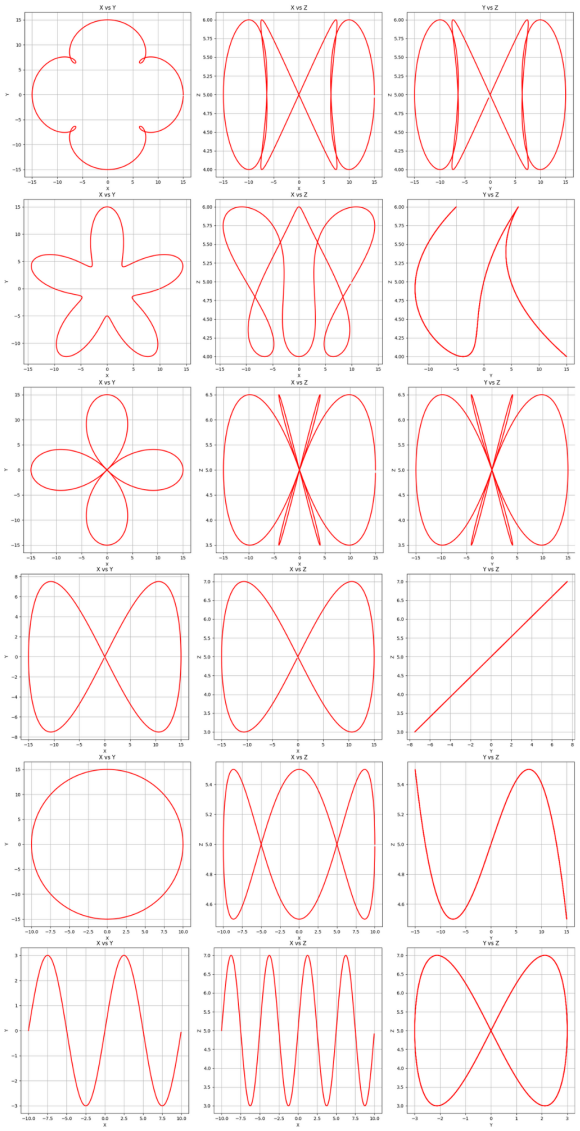

Fig. 6. Training Trajectories in 2D. Rows correspond to trajectories (top to bottom: Wavy circle, Star, Clover, Line, Oval, Figure of eight), and columns represent the orientation of view (left to right: X-Y, X-Z, Y-Z).

achieves the lowest *RMSE ATE* and *Median ATE*, indicating superior absolute trajectory accuracy. However, in terms of *RMSE RPE* and *Median RPE*, differences are less pronounced, with Vision-Only sometimes outperforming others, as seen in the Spiral trajectory.

Interestingly, the Visual-Inertial fusion does not consistently outperform the unimodal baselines. While it achieves a more balanced performance across some metrics (e.g., lower ATE Median in Infinity), it sometimes shows higher scale drift than IMU-Only (e.g., 0.15 vs. 0.11 for Infinity). This suggests potential issues in sensor fusion alignment or overfitting in the combined model. Overall, while IMU-Only excels in absolute pose estimation, Vision-Only shows competitive relative accuracy, and the fusion model needs tuning to fully leverage complementary modalities.

TABLE I
ATE, RPE, AND SCALE DRIFT COMPARISON ACROSS METHODS AND TRAJECTORIES

Trajectory	Method	RMSE ATE	Median ATE	RMSE RPE	Median RPE	Scale Drift
Infinity	Vision-Only	17.93	10.23	2.32	2.03	0.13
Infinity	IMU-Only	8.91	8.61	2.39	2.18	0.11
Infinity	Visual-Inertial	12.17	7.86	2.47	2.12	0.15
Random	Vision-Only	31.50	21.51	3.23	2.80	0.23
Random	IMU-Only	9.50	7.22	3.11	2.78	0.24
Random	Visual-Inertial	20.81	12.60	3.25	2.90	0.24
Spiral	Vision-Only	15.98	10.67	1.79	1.84	0.16
Spiral	IMU-Only	7.91	7.73	1.94	1.74	0.22
Spiral	Visual-Inertial	10.98	8.41	1.89	1.79	0.19

VII. RESEARCH CHALLENGES IN DEEP VIO

Deep learning-based Visual-Inertial Odometry (VIO) holds significant promise but still faces several core research challenges:

- **Robust Sensor Fusion:** Fusing asynchronous and noisy visual-inertial inputs remains difficult, especially under fast motion or degraded visual conditions. Learning-based temporal and cross-modal attention mechanisms are a promising direction.
- **Domain Generalization:** Deep VIO models often overfit to specific environments or motion types. Designing models that generalize across diverse domains, lighting, and sensor configurations remains an open problem.
- **Scale Drift and Global Consistency:** Maintaining consistent metric scale over long trajectories is challenging. Incorporating global constraints or loop closures into learned odometry is an active research area.
- **Uncertainty Estimation:** Most current models provide deterministic pose estimates. Learning to predict pose uncertainty can enable downstream fusion with SLAM and improve safety in real-world deployments.
- **Data Efficiency:** Supervised methods are limited by the scarcity of ground-truth pose data. Self-supervised learning, synthetic data augmentation, and domain adaptation offer paths to more scalable training.

Addressing these challenges is key to deploying deep VIO systems in real-world robotics, autonomous navigation, and AR/VR applications.

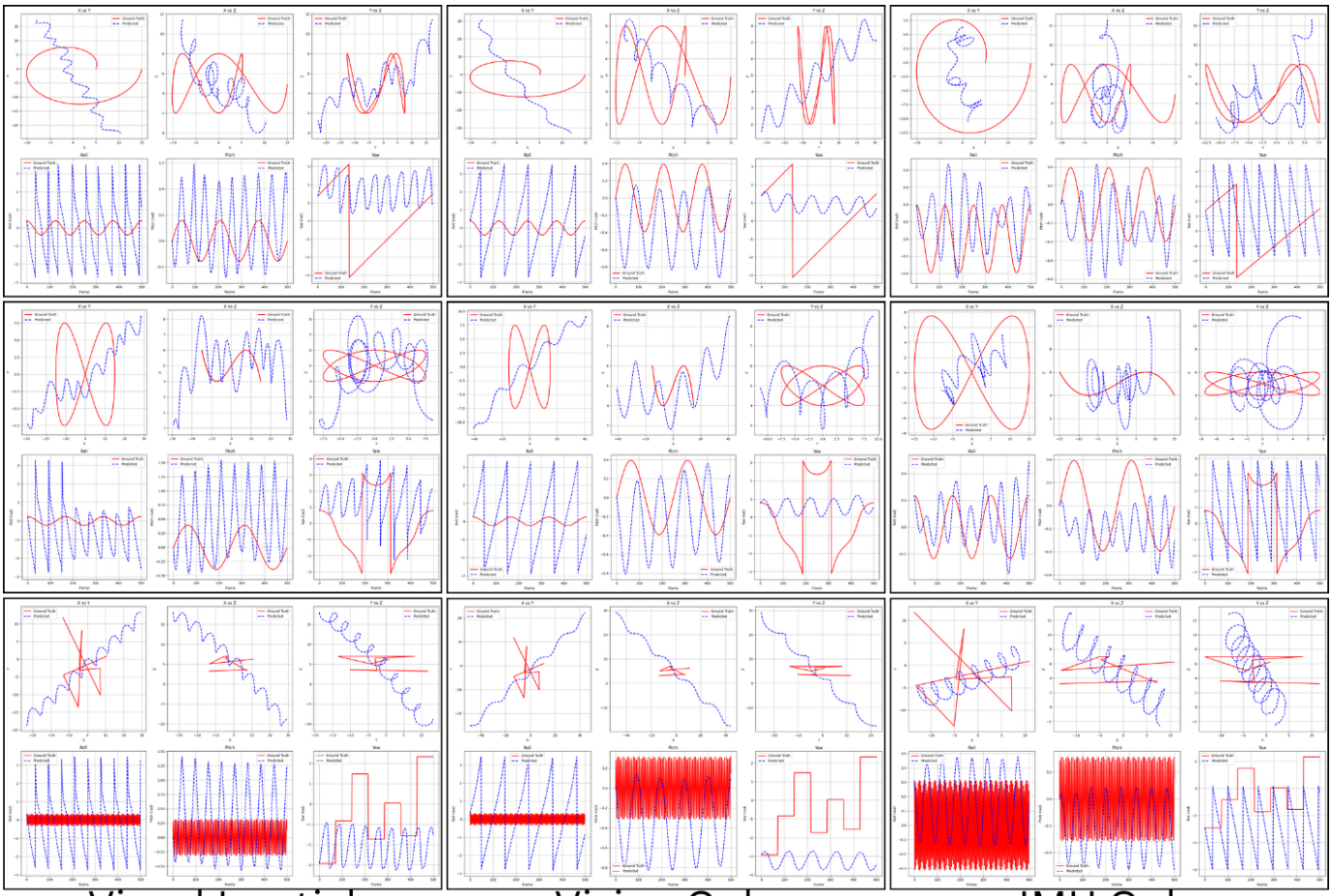

Fig. 7. Odometry Outputs on testing trajectories. (Spiral, Infinity, Random)

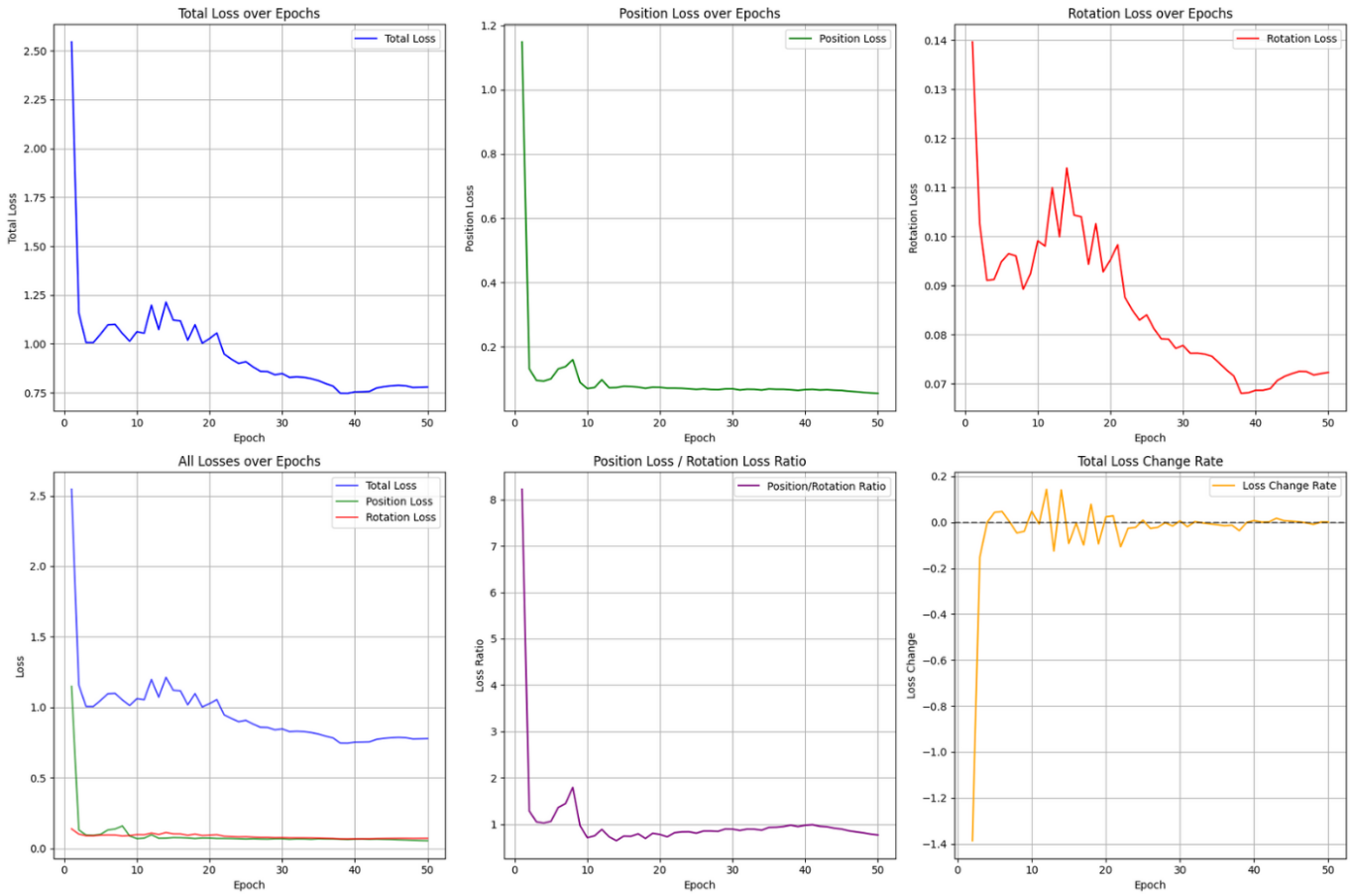

Fig. 8. Training Metrics