RESEARCH-ARTICLE

Detection and Recovery of Adversarial Slow-Pose Drift in Offloaded Visual-Inertial Odometry

SOURYA SAHA, The City University of New York, New York, NY, United States

MD NURUL ABSUR, The City University of New York, New York, NY, United States

SAPTARSHI DEBROY, The City University of New York, New York, NY, United States

Open Access Support provided by:

The City University of New York

Published: 27 October 2025

Citation in BibTeX format

MobiHoc '25: Twenty-sixth International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing
October 27 - 30, 2025
TX, Houston, USA

Conference Sponsors:
SIGMOBILE

https://doi.org/10.1145/3704413.3765307
ISBN: 9798400713538

Detection and Recovery of Adversarial Slow-Pose Drift in Offloaded Visual-Inertial Odometry

Sourya Saha*
City University of New York
New York, USA
ssaha2@gradcenter.cuny.edu

Md. Nurul Absur*
City University of New York
New York, USA
mabsur@gradcenter.cuny.edu

Saptarshi Debroy
City University of New York
New York, USA
saptarshi.debroy@hunter.cuny.edu

ABSTRACT

Visual–Inertial Odometry (VIO) supports immersive Virtual Reality (VR) by fusing camera and Inertial Measurement Unit (IMU) data for real-time pose. However, current trend of offloading VIO to edge servers can lead to server-side threat surface where subtle pose spoofing can accumulate into substantial drift, while evading heuristic checks. In this paper, we study this threat and present an unsupervised, label-free detection and recovery mechanism. The proposed model is trained on attack-free sessions to learn temporal regularities of motion to detect runtime deviations and initiate recovery to restore pose consistency. We evaluate the approach in a realistic offloaded-VIO environment using ILLIXR testbed across multiple spoofing intensities. Experimental results in terms of well-known performance metrics show substantial reductions in trajectory and pose error compared to a no-defense baseline.

CCS CONCEPTS

• **Human-centered computing** → **Mobile devices**; • **Computing methodologies** → *Neural networks*; • **Security and privacy** → **Domain-specific security and privacy architectures**; • **Computer systems organization** → *Availability*.

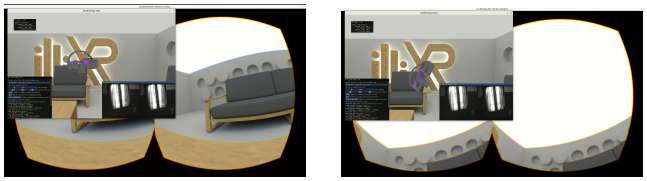
KEYWORDS

Virtual reality, pose spoofing attack, unsupervised anomaly detection, adversarial perturbations, quality of experience

ACM Reference Format:
Sourya Saha, Md. Nurul Absur, and Saptarshi Debroy. 2025. Detection and Recovery of Adversarial Slow-Pose Drift in Offloaded Visual-Inertial Odometry. In *The Twenty-sixth International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing (MobiHoc '25), October 27–30, 2025, Houston, TX, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3704413.3765307

1 INTRODUCTION

Virtual Reality (VR) systems are expanding across domains, such as entertainment, education, and public safety, enabled by lightweight headsets and high-fidelity rendering. At the core of these experiences lies Visual-Inertial Odometry (VIO) [18], which fuses camera and Inertial Measurement Unit (IMU) data to estimate six-Degrees of Freedom (DoF) head pose. To mitigate the power and thermal limits of mobile headsets, recent designs offload such VIO computation to nearby edge servers to preserve resource-constrained devices and extend battery life [4]. Although such offloading improves end-to-end performance, it opens a new threat surface where server-returned poses may get delayed, degraded, or adversarially manipulated before reaching the headset.



(a) No pose spoofing (b) Significant spoofing

Figure 1: QoE impact of subtle VIO perturbations. Same head pose: normal (left) vs. anomalous (right).

One of such subtle yet damaging vector to manipulate such server-returned poses can be *adversarial slow-pose drift* which are low-magnitude, temporally consistent perturbations injected into the offloaded head-pose stream. Such drifts can evade residual filters and conventional anomaly detectors, but compound over time to result in spatial misalignment, visual jitter, and degraded immersion (Figure 1). Existing edge-offloaded VIO pipelines typically assume a trustworthy server and lack explicit verification, including commercial and research frameworks such as ARCore [3], HoloLens [10], and ILLIXR [6], leaving them exposed to continuous, low-rate manipulation [14, 16]. Prior efforts to mitigate pose drift in VIO/SLAM include loop closures [12], pose-graph optimization [11], Kalman-filtering variants [1], residual-based thresholding [9, 15], predictive delay compensation, and adversarial filtering for SLAM [8, 13, 17]. *These methods can be effective in high-motion scenes or when deviations are large and easily flagged. However, they are less suited to slow, low-rate drift that preserves short-term consistency.*

In this paper, we present a lightweight, unsupervised, headset-side defense for edge-offloaded VIO. Our approach models the natural consistency between server-returned "slow" poses and locally integrated "fast" poses. We design a deep autoencoder, trained only on clean pose and IMU data from attack-free runs, that learns intrinsic motion regularities. At runtime, each incoming slow pose is evaluated using the autoencoder's reconstruction error over a window of preceding fast-pose and IMU features. Our simple policy design accepts normal poses, drops anomalies, or forces a pass after many consecutive drops to re-anchor the fast-pose stream and avoid its own drift. We evaluate our system on an ILLIXR-based testbed [6], under multiple spoofing configurations. We use clean-run trajectories as reference ground truth for offline Absolute Trajectory Error (ATE) and Relative Pose Error (RPE). The results

*Both authors contributed equally to this work.

MobiHoc '25, October 27–30, 2025, Houston, TX, USA

ACM ISBN 979-8-4007-1353-8/25/10
https://doi.org/10.1145/3704413.3765307

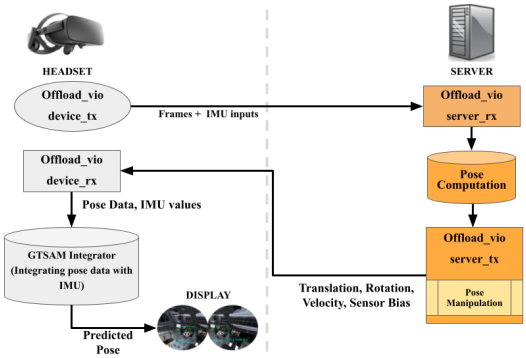


Figure 2: Headset offloads sensors inputs to edge and computes local fast-pose window; edge server returns slow-pose.

show that under moderate spoofing, the defense improves both metrics by more than 10x with minimal added latency, on the order of a few milliseconds per slow-pose frame on a typical edge server.

The remainder of the paper is organized as follows. Section 2 details the problem formulation. Section 3 presents the defense pipeline. Section 4 reports the experimental results. Section 5 concludes the paper.

2 PROBLEM FORMULATION

2.1 System Model

We build on the ILLIXR framework and the RemoteVIO paradigm [7] by simulating the VR client on a PC using the ILLIXR application, that streams prerecorded image and IMU data to an edge server for full VIO computation, while retaining on-device local pose computation to preserve responsiveness.

As shown in Figure 2, on the client, the `device_tx` plugin packages the latest image frames and IMU readings into a Protocol Buffer message and sends it to the server via TCP. While awaiting the server response, the headset integrates IMU data to maintain a local *fast-head-pose buffer* $\mathcal{F}_i = \{\mathbf{f}_{i,1}, \ldots, \mathbf{f}_{i,K}\}$, ensuring real-time visual feedback.

Upon receiving the slow head pose $\mathbf{s}_i$, the `device_rx` plugin deserializes it and uses it to re-anchor local fast-pose integration for the subsequent window $\mathcal{F}_{i+1}$. This prevents drift from propagating across windows and keeps the client trajectory aligned with the server-computed global pose.

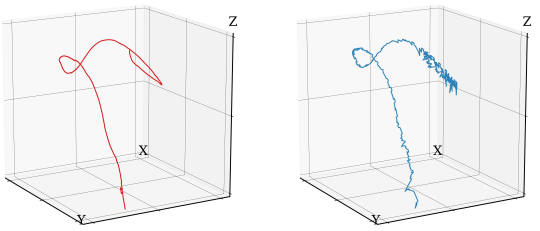
At the server, the `server_rx` plugin deserializes the sensor data and runs a full VIO pipeline (OpenVINS) to estimate $\mathbf{s}_i$:

$$\mathbf{s}_i = \left[\mathbf{p}_i, \mathbf{q}_i, \mathbf{v}_i, \mathbf{b}_i^{\text{acc}}, \mathbf{b}_i^{\text{gyro}}\right], \tag{1}$$

where $\mathbf{p}_i$ is 3D position, $\mathbf{q}_i$ the orientation quaternion, $\mathbf{v}_i$ the linear velocity, and $\mathbf{b}_i^{\text{acc}}, \mathbf{b}_i^{\text{gyro}}$ the accelerometer and gyroscope biases. The `server_tx` plugin serializes and returns $\mathbf{s}_i$ to the client, completing the round and re-aligning local integration with the global server trajectory.

2.2 Threat Model

We assume the edge server that executes the offloaded VIO pipeline is adversary-controlled (Figure 2). At each offload round i, the adversary spoofs the returned slow pose with probability p; let $b_i \sim \text{Bernoulli}(p)$ denote this event. If $b_i = 1$, the true slow-pose state $\mathbf{s}_i = [\mathbf{p}_i, \mathbf{q}_i, \mathbf{v}_i, \mathbf{b}_i]$, with $\mathbf{b}_i = [\mathbf{b}_i^{\text{acc}}; \mathbf{b}_i^{\text{gyro}}]$, is perturbed by small, bounded additive drifts $\delta_i = [\delta_i^{\text{pos}}, \delta_i^{\text{ang}}, \delta_i^{\text{vel}}, \delta_i^{\text{bias}}]$ to produce $\tilde{\mathbf{s}}_i = \mathbf{s}_i + \delta_i$; otherwise ($b_i = 0$) the pose is returned unmodified. The client does not observe b_i and has no ground-truth labels or information to distinguish spoofed from clean rounds. Its inputs at round i are the possibly spoofed slow pose $\tilde{\mathbf{s}}_i$ and the locally integrated fast-pose buffer $\mathcal{F}_{i-1}$ accumulated since the previous slow-pose update. The manipulation is injected after server-side pose computation and before the transmission back to the headset.



(a) No spoofing (0%) **(b) Significant spoofing (75%)**

Figure 3: Client 3D trajectories: normal vs. adversarial spoofing; Subplot b shows spoofing-induced drift.

2.3 Problem Evidence Analysis

Edge-offloaded VIO pipelines inherently trust slow-pose updates from the server, with no access to ground truth or cues about potential perturbations. Small, consistent drifts can silently corrupt the motion trajectory, accumulating over time without triggering residual filters or anomaly thresholds.

Each slow pose anchors the next fast-pose integration window, and when spoofed, errors propagate downstream, compounding drift in fast poses. As depicted in Figure 3, trajectories diverge and exhibit jitter with the onset of spoofing, which worsens with increased spoofing probability; even severe drift affects slow poses, degrading spatial fidelity. These errors compromise scene stability, creating jitter and user fatigue. Given the stealthy nature of perturbations and the lack of client verification capabilities, a real-time detection and correction mechanism on the headset is crucial to preserving motion accuracy and QoE in immersive XR.

2.4 Problem Statement

Given input pairs $\{(\tilde{\mathbf{s}}_i, \mathcal{F}_{i-1})\}_{i=1}^{N}$, where each slow pose $\tilde{\mathbf{s}}_i$ may be adversarially perturbed, the client must:

- **Detect:** Identify which slow poses are anomalous by evaluating their consistency with preceding motion.
- **Decide:** Choose whether to accept $\tilde{\mathbf{s}}_i$ or drop it, thereby influencing how the subsequent fast poses are integrated.

We evaluate the effectiveness of this decision process using the headset's *fast pose outputs* $\{\mathbf{f}_t^{\text{out}}\}_{t=1}^{T}$, where $T \gg N$ denotes the total number of high-rate fast-pose estimates produced during runtime. The evaluation metrics are:

$$\text{ATE} = \sqrt{\frac{1}{T}\sum_{t=1}^{T}\left\|\mathbf{f}_t^{\text{out}} - \mathbf{f}_t^{\text{gt}}\right\|^2}, \quad \text{RPE} = \sqrt{\frac{1}{T-1}\sum_{t=1}^{T-1}\left\|\Delta\mathbf{f}_t^{\text{out}} - \Delta\mathbf{f}_t^{\text{gt}}\right\|^2}, \tag{2}$$

where $\mathbf{f}_t^{\text{gt}}$ denotes the ground-truth pose at timestamp t, and $\Delta\mathbf{f}_t = \mathbf{f}_{t+1} \ominus \mathbf{f}_t$ represents the relative motion (translation or rotation). The goal is to ensure that corrupted slow poses do not degrade the downstream fast-pose trajectory.

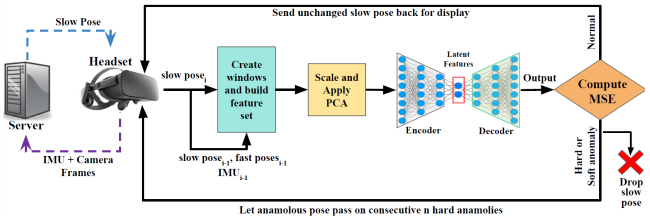


Figure 4: Runtime pipeline: features for each slow pose constructed using preceding fast-pose/IMU window; reconstruction error decides accept/drop/force-pass.

3 DEFENSE FRAMEWORK

We propose a lightweight, client-side framework to detect and mitigate adversarial drift in slow pose estimates using an unsupervised approach. The system leverages the temporal consistency between fast and slow pose streams and operates without labeled spoofed data, making it practical for real-time deployment on XR headsets.

Since the client has no access to ground-truth drift labels or attacker strategy, detection must rely solely on motion patterns learned from clean, attack-free runs. This is viable because pose estimation in XR is a deterministic function of sensor input: for a given motion, the resulting pose trajectory is governed by sensor fusion and is independent of the XR application in use. Thus, clean motion windows recorded under normal conditions generalize across applications, enabling an autoencoder trained on these sequences to detect deviations introduced by spoofed poses.

3.1 Feature Extraction and Temporal Encoding

For each slow pose s_i, we form features from the prior window between the last accepted s_{i-1} and s_i—fast poses $\mathcal{F}_{i-1} = \{\mathbf{f}_{i-1,k}\}_{k=1}^{K}$ and IMU samples $\mathcal{I}_{i-1} = \{\mathbf{z}_{i-1,k}\}_{k=1}^{K}$, to avoid contamination from spoofed s_i. The vector includes count K, duration; pose residuals between $\mathbf{f}_{i-1,k}$ and s_i for position/velocity summarized by mean/std/min/max and ℓ_1 norm; quaternion geodesic orientation error; accelerometer/gyroscope bias summaries; and IMU activity (norms and axis-wise statistics). Features are z-normalized, PCA-reduced to 97% variance, and fed to the autoencoder for anomaly scoring.

3.2 Autoencoder-Based Anomaly Detection

Figure 4 illustrates the defense pipeline. We train a compact fully connected autoencoder on clean logs to reconstruct PCA-reduced features summarizing the fast-pose/IMU window preceding each slow pose (architecture in Table 1). The encoder maps inputs to a low-dimensional latent and the decoder reconstructs them; training uses SmoothL1 (Huber) loss with light Gaussian jitter and early stopping. At runtime, the same feature/PCA steps feed the model and the reconstruction mean-squared error (MSE) serves as the anomaly score; spoofed inputs typically yield larger errors. Thresholds calibrated on a clean validation set define 3 anomaly types: *Normal* when MSE is below median + 3× the median absolute deviation (MAD), *Soft anomaly* when the MSE lies between that value and the 98th percentile, and *Hard anomaly* when at or above the 98th percentile. Normal poses are accepted; soft and hard anomalies are dropped. To avoid prolonged drift during extended hard anomaly drops, we allow a forced pass after 12 consecutive hard classifications to re-anchor the fast-pose stream.

Table 1: Proposed autoencoder architecture.

Stage	Layer	Output Size	Activation / Notes
Encoder	Linear (input → 256)	256	LReLU + BN1d
	Linear (256 → 128)	128	LReLU + BN1d
	Linear (128 → 64)	64	LReLU + BN1d
	Linear (64 → LATENT_DIM)	32	–
Decoder	Linear (LATENT_DIM → 64)	64	LReLU
	Linear (64 → 128)	128	LReLU
	Linear (128 → 256)	256	LReLU
	Linear (256 → input dim)	input size	Tanh

4 EVALUATION

4.1 Experimental Setup

We evaluate the framework in real time on ILLIXR using two PCs (Intel Core i9-14900K, NVIDIA RTX 2000, 32 GB RAM): the client emulates the headset and the server runs VIO. The server's VIO plugin emits clean and spoofed slow poses, with drift magnitudes and Bernoulli injection probability p set via a configuration file to sweep attack intensities. For autoencoder training, we collect client-side IMU, fast poses, and server-returned slow poses across 100 clean runs, logging per-run poses (position, velocity, quaternion), sensor biases, and raw IMU traces. On the client, a lightweight Python sidecar receives fast poses/IMU over ZMQ and slow poses over TCP, builds per–slow-pose features from the preceding fast-pose/IMU window, applies normalization+PCA, and scores anomalies with the trained autoencoder; the resulting decision (accept, soft/hard reject, or forced pass) is returned to ILLIXR to anchor the next fast-pose window. All detection/decision logic remains external to ILLIXR with minimal hooks.

4.2 Metrics

We evaluate our framework with three complementary measures that capture global drift, local motion consistency, and detector fidelity: Absolute Trajectory Error (ATE)—Euclidean distance between estimated and reference trajectories; Relative Pose Error (RPE)—deviation in frame-to-frame motion; and autoencoder reconstruction error (MSE)—mean-squared difference between the PCA-reduced feature input and its reconstruction. We do not use native (non-offloaded) ILLIXR ground truth as the ATE/RPE reference because even small, systematic trajectory drift between native and offloaded runs would overshadow the subtler effects of spoofing and the defense. Instead, clean offloaded trajectories collected under identical conditions serve as the reference, isolating the impact of perturbations and the defense.

4.3 Results

Unless otherwise noted, all results in Sections 4.3.1–4.3.4 use a fixed set of attacker parameters to inject low-magnitude drifts into slow-pose updates. The spoofing was controlled via environment variables on the server in Table 2.

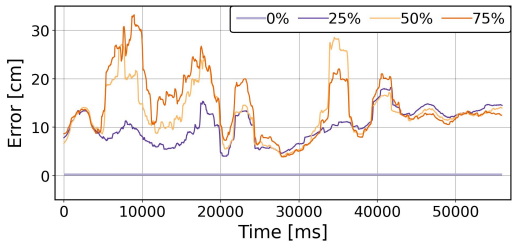
Table 2: Server environment variables.

Variable	Value
ILLIXR_VIO_SPOOF_BIAS_DRIFT	0.05
ILLIXR_VIO_SPOOF_VELOCITY_DRIFT	0.10
ILLIXR_VIO_SPOOF_POSITION_DRIFT	0.02
ILLIXR_VIO_SPOOF_ANGLE_DRIFT	0.20

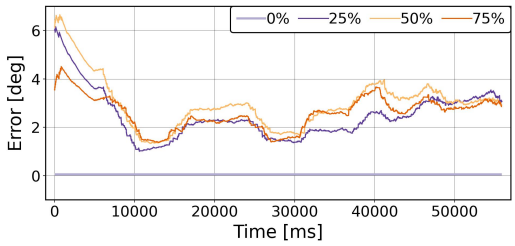
These values were held constant across spoofing rates (0%, 25%, 50%, and 75%) to isolate the impact of spoof frequency under a uniform attack model. Section 4.3.6 explores defense robustness by

Table 3: Mean ATE and RPE under varying spoofing levels without any defense strategy.

Spoofing Level	T-ATE (cm)	R-ATE (deg)	T-RPE (cm)	R-RPE (deg)
0%	0.074	0.008	0.033	0.0019
25%	10.319	16.353	0.815	2.453
50%	13.043	33.904	1.449	2.980
75%	14.336	49.465	1.709	2.567



(a) Smoothed translation ATE over time



(b) Smoothed rotation RPE over time

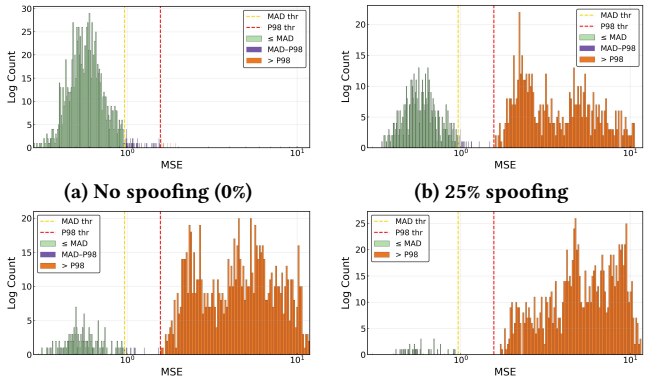
Figure 5: Evaluation of pose degradation without defense under different spoofing levels. Higher spoofing levels lead to increasingly persistent and severe trajectory errors.

varying these parameters, testing generalization to unseen attack profiles. For all experiments, we report results from the best of 10 ILLIXR runs per condition, both with and without the defense enabled.

4.3.1 ATE and RPE without the Defense Strategy. We begin by quantifying the impact of spoofed slow poses on the offloaded VIO pipeline in the absence of any defense. Table 3 reports the mean translation and rotation ATE and RPE across spoofing levels of 0%, 25%, 50%, and 75%. As expected, all metrics remain negligible for the clean 0% case. However, even moderate spoofing at 25% causes a noticeable increase (e.g., T-ATE: 10.3 cm, R-RPE: 2.45°). As spoofing frequency increases to 50% and 75%, the degradation compounds further, with rotational errors rising to 2.98° and 2.57°, respectively.

Figure 5 presents the temporal trends of translation ATE and rotation RPE. The clean run shows flat error curves, whereas higher spoofing levels exhibit growing plateaus of drift. Notably, the 75% spoofed run reveals extended regions of high trajectory error, showing how compounded spoofed poses corrupt fast-pose propagation over time. These results highlight the vulnerability of offloaded VIO to persistent low-magnitude spoofing and establish the need for robust real-time anomaly detection on the client side.

4.3.2 Autoencoder Performance. We analyze the autoencoder's reconstruction error (MSE) across four spoofing levels—0%, 25%, 50%, and 75%—with results shown in Figure 6. Each histogram plots per-frame MSE over 100 bins, along with the MAD-based anomaly



(a) No spoofing (0%) **(b) 25% spoofing**

(c) 50% spoofing **(d) 75% spoofing**

Figure 6: Histogram of autoencoder reconstruction errors (MSE) across four spoofing scenarios. The yellow line denotes the MAD-based threshold; the red line shows the 98th percentile. As spoofing increases, MSE distribution shifts right.

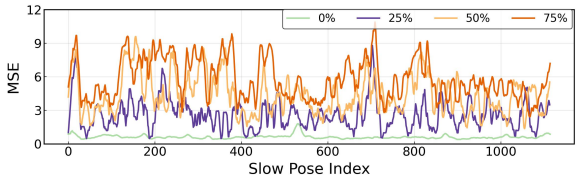
threshold (yellow dashed line) and the 98th percentile threshold (red dashed line).

In the clean setting, MSE values remain tightly clustered well below the threshold, with over 95% of slow poses classified as normal. As spoofing increases, the distribution shifts rightward. At 25%, only 44% of slow poses remain normal, while 54% are hard anomalies. This trend intensifies at 50% and 75%, where 86.7% and 96.8% of poses are flagged as hard anomalies.

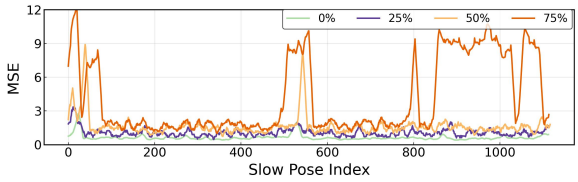
Interestingly, the fraction of detected anomalies exceeds the spoofing probability. This amplification stems from corrupted slow poses contaminating downstream fast-pose windows, leading to broader temporal inconsistency detectable by the autoencoder—even if the corruption was sparse. Despite this, the response remains monotonic: higher spoofing leads to more detections, while the clean case avoids false positives. This confirms the autoencoder's sensitivity and generalization capability under varied attack intensities.

4.3.3 Autoencoder Behavior in Passive Detection Mode. To isolate detector behavior from decision-making, we run full real-time feature extraction and autoencoder inference on the client but forward all slow poses unchanged. Figure 7a shows MSE traces for spoofing rates of 0%, 25%, 50%, and 75%. Clean runs remain low and stable, while increasing spoofing produces progressively higher and denser spikes, culminating in near-continuous peaks at 75%—indicating complete breakdown of fast–slow correlation. This confirms that reconstruction error alone reliably captures spoofing-induced motion inconsistency.

4.3.4 Anomaly Detection Behavior (MSE Time Series). With the full pipeline active, each slow pose is scored in real time and classified as normal, soft anomaly (drop), or hard anomaly (normally drop, forced pass after a number of subsequent hard anomalies). Figure 7b shows that at 0% spoofing, MSE stays low, avoiding false positives. As spoofing increases, spikes occur but are shorter and less persistent than in passive mode (Section 4.3.3). At high spoofing (75%), periodic forced passes cause brief MSE surges, but the system



(a) Passive detection mode



(b) Active detection and recovery mode

Figure 7: MSE over time (averaged over 100 slow-pose bins) for different spoofing frequencies, where higher spoofing causes larger and more frequent error spikes.

Table 4: Mean ATE and RPE under varying spoofing levels

Spoofing Level	T-ATE (cm)	R-ATE (deg)	T-RPE (cm)	R-RPE (deg)
0 %	0.400	0.049	0.043	0.0036
25 %	0.388	0.049	0.044	0.0039
50 %	1.369	2.166	0.082	0.041
75 %	27.812	23.213	0.979	0.146

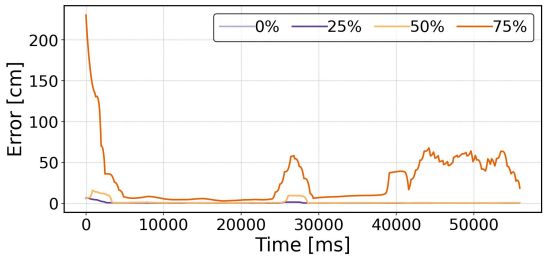
quickly re-stabilizes, preventing cumulative drift. Overall, active defense both limits spike duration and suppresses downstream error growth under sustained attack.

4.3.5 Pose Quality Under Full Defense. We now evaluate the full defense system's ability to suppress trajectory degradation under increasing spoofing levels. Table 4 reports mean translational and rotational ATE and RPE for spoofing rates of 0%, 25%, 50%, and 75%. Compared to the unprotected case (Table 3), our defense substantially reduces error in almost all metrics across all spoofing intensities.

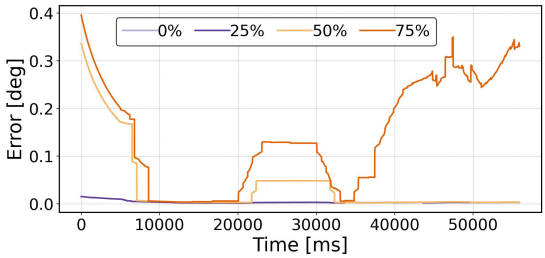
At 0% and 25% spoofing, ATE and RPE remain nearly indistinguishable from clean baseline: all metrics are below 0.5 cm and 0.05°, indicating robust rejection of false positives and stability in benign scenarios. At 50% spoofing, the defense suppresses translational ATE from over 13 cm to 1.37 cm and rotational ATE from 33° to 2.17°. Translational and rotational RPE both fall below 0.1, confirming strong correction of local motion consistency.

At 75%, although RPE remains low (0.98 cm and 0.15°), cumulative ATE rises (27.81 cm and 23.21°). This occurs due to frequent forced passes of anomalous poses that re-anchor fast pose streams with uncorrected values. While this helps preserve scene continuity under severe spoofing, it leads to global drift over time.

Temporal plots in Figure 8 (top: ATE, bottom: RPE) further highlight the contrast. Compared to the unprotected case in Figure 5, the defense curves are significantly flatter. Translation ATE remains below 10 cm for 25% and 50%, and even the 75% curve shows rapid recovery after spikes. Rotation RPE stays under 0.4° across all cases—an order-of-magnitude improvement over the 6° peak seen without defense.



(a) Smoothed translation ATE over time (500-frame window).



(b) Smoothed rotation RPE over time (2000-frame window).

Figure 8: Evaluation with complete defense under different spoofing levels. Panels show translation ATE (top) and rotation RPE (bottom) over time.

4.3.6 Robustness to Varying Attack Configurations. To evaluate the generalization ability of our defense, we test it against four *increasingly aggressive* attack configurations (Config 1–4), each defined by different drift magnitudes in bias, velocity, position, and orientation (Table 5). We fix the spoofing probability at 50%—a balanced choice that allows degradation to surface without fully destabilizing the system. Lower spoofing (25%) proved too subtle for clear differentiation, while 75% often led to collapse regardless of configuration.

We focus on RPE instead of ATE as the primary metric. This because ATE is sensitive to transient drift and may remain elevated even after recovery, whereas RPE better reflects real-time pose stability and local consistency. Across all four configurations, the system maintains sub-1 RPE values for both translation and rotation—even in Config 4, which features the most severe drift values (e.g., 1.2 bias, 1.5 velocity, 1.0 position, 2.0 angle). This indicates strong generalization of our defense to unseen perturbation levels and attack directions.

Each result corresponds to the best of 10 runs. While higher drifts causes sharper divergence at spoofed frames, the system consistently stabilizes shortly afterward. These results confirm the

Table 5: Attack configurations and corresponding mean RPE values (50% spoofing).

Config	Bias Drift	Velocity Drift	Position Drift	Angle Drift	Trans RPE Mean (cm)	Rot RPE Mean (deg)
Config 1	0.05	0.10	0.02	0.2	0.0477	0.0045
Config 2	0.10	0.30	0.09	0.5	0.0582	0.0062
Config 3	0.60	0.80	0.50	0.9	0.3139	0.0591
Config 4	1.20	1.50	1.00	2.0	0.0591	0.0065

resilience of our defense pipeline under a wide range of adversarial conditions, including strong, previously unseen perturbations.

4.3.7 Real-Time Inference Under Mobile CPU Constraints. To assess deployability on XR headsets, we benchmark our anomaly detection pipeline under conditions approximating the Meta Quest 3s, which features a Snapdragon XR2 Gen 2 SoC with octa-core Kryo CPUs (2.0–3.19 GHz).

We replicate this on a FABRIC [2] node with 8 GB RAM, 100 GB storage, and 8 vCPUs from an AMD EPYC 7532 processor, frequency-locked at 2.4 GHz. Turbo scaling is disabled for consistency.

The full defense pipeline—including feature extraction, preprocessing (StandardScaler + PCA), TorchScript autoencoder inference, and threshold-based postprocessing—executes over 1,035 slow-pose windows. Model and scaler loading occur only once at startup.

As shown in Table 6, all stages complete well within real-time constraints. Feature extraction is the most expensive at 4.77 ms per frame, while preprocessing, inference, and postprocessing each require under 0.3 ms. The one-time model load (37.4 ms) occurs at headset startup, before VIO offloading, and plays no role in per-frame decisions or inference. All evaluation related codes and data are available through Github [5].

Table 6: Mean latency per detection stage (in milliseconds), averaged over 1,035 slow pose windows. Model and scaler load time occurs once at startup.

Stage	Mean Time (ms)
Model + Scaler Load (once)	37.40
Feature Extraction	4.77
Preprocessing + PCA	0.23
Autoencoder Inference	0.26
Threshold Postprocessing	0.05

5 CONCLUSIONS

In this paper, we present a real-time, headset-side defense for adversarial pose spoofing in offloaded XR systems. The proposed lightweight autoencoder-based pipeline monitored incoming slow poses using temporal features from fast-pose and IMU windows, flagging anomalies based on reconstruction error and deciding whether to accept, reject, or forward each pose. Evaluated on the ILLIXR platform, our method improved ATE and RPE by over 10× at moderate spoofing levels (50%), while maintaining low latency on mobile-class CPU hardware. Even under high spoofing (75%), the defense preserved frame-to-frame consistency and prevents jitter, though partial drift did occur due to forced passes of corrupt poses. Latency benchmarks confirmed deployability on XR-class hardware where the full pipeline ran in real time, with the slowest stage (feature extraction) taking under 5 ms per slow pose.

Moving forward, we plan to explore reinforcement learning-based strategies for visual degradation recovery, allowing the system to dynamically adapt to persistent or evolving spoofing patterns.

ACKNOWLEDGMENTS

This work is supported by the National Science Foundation under Award Number CNS-2401928.

REFERENCES

[1] Shaza I. Kaoud Abdelaziz, Haidy Y. Elghamrawy, Aboelmagd M. Noureldin, and Georgia Fotopoulos. 2024. Body-Centered Dynamically-Tuned Error-State Extended Kalman Filter for Visual Inertial Odometry in GNSS-Denied Environments. *IEEE Access* 12 (2024), 15997–16008. https://doi.org/10.1109/ACCESS.2024.3357458

[2] Ilya Baldin, Anita Nikolich, James Griffioen, Indermohan Inder S Monga, Kuang-Ching Wang, Tom Lehman, and Paul Ruth. [n. d.]. FABRIC: A national-scale programmable experimental network infrastructure. *IEEE Internet Computing* 23, 6 ([n. d.]), 38–47.

[3] Komang Candra Brata, Nobuo Funabiki, Sritrusta Sukaridhoto, Evianita Dewi Fajrianti, and Mustika Mentari. 2023. An Investigation of Running Load Comparisons of ARCore on Native Android and Unity for Outdoor Navigation System Using Smartphone. In *2023 Sixth International Conference on Vocational Education and Electrical Engineering (ICVEE)*. 133–138. https://doi.org/10.1109/ICVEE59738.2023.10348201

[4] Steven Gao, Jeffrey Liu, Qinjun Jiang, Finn Sinclair, William Sentosa, Brighten Godfrey, and Sarita Adve. 2025. XRgo: Design and Evaluation of Rendering Offload for Low-Power Extended Reality Devices. In *Proceedings of the 16th ACM Multimedia Systems Conference* (Stellenbosch, South Africa) *(MMSys '25)*. Association for Computing Machinery, New York, NY, USA, 124–135. https://doi.org/10.1145/3712676.3714444

[5] GitHub. 2025. Github repository. https://github.com/dissectlab/Mobihoc-XR2025.git. Accessed: Sep 10, 2025.

[6] Muhammad Huzaifa, Rishi Desai, Samuel Grayson, Xutao Jiang, Ying Jing, Jae Lee, Fang Lu, Yihan Pang, Joseph Ravichandran, Finn Sinclair, Boyuan Tian, Hengzhi Yuan, Jeffrey Zhang, and Sarita V. Adve. 2021. ILLIXR: Enabling End-to-End Extended Reality Research. In *2021 IEEE International Symposium on Workload Characterization (IISWC)*. 24–38. https://doi.org/10.1109/IISWC53511.2021.00014

[7] Qinjun Jiang, Yihan Pang, William Sentosa, Steven Gao, Muhammad Huzaifa, Jeffrey Zhang, Javier Perez-Ramirez, Dibakar Das, David Gonzalez-Aguirre, Brighten Godfrey, and Sarita Adve. 2025. RemoteVIO: Offloading Head Tracking in an End-to-End XR System. In *Proceedings of the 16th ACM Multimedia Systems Conference* (Stellenbosch, South Africa) *(MMSys '25)*. Association for Computing Machinery, New York, NY, USA, 101–112. https://doi.org/10.1145/3712676.3714442

[8] Pingfei Li, Lu Wang, Yutong Zu, Xuesong Bai, and Yuanbiao Hu. 2023. Multi-sensor fusion method based on FFR-FK for 3D trajectory measurement of underground pipelines. *Tunnelling and Underground Space Technology* 141 (2023), 105344. https://doi.org/10.1016/j.tust.2023.105344

[9] Mauricio Pereira and Branko Glisic. 2023. Detection and quantification of temperature sensor drift using probabilistic neural networks. *Expert Systems with Applications* 213 (2023), 118884. https://doi.org/10.1016/j.eswa.2022.118884

[10] Alicia Pose-Díez-de-la Lastra, Rafael Moreta-Martinez, Mónica García-Sevilla, David García-Mato, José Antonio Calvo-Haro, Lydia Mediavilla-Santos, Rubén Pérez-Mañanes, Felix von Haxthausen, and Javier Pascau. 2022. HoloLens 1 vs. HoloLens 2: Improvements in the New Model for Orthopedic Oncological Interventions. *Sensors* 22, 13 (2022). https://doi.org/10.3390/s22134915

[11] Yulun Tian, Kasra Khosoussi, David M. Rosen, and Jonathan P. How. 2021. Distributed Certifiably Correct Pose-Graph Optimization. *IEEE Transactions on Robotics* 37, 6 (2021), 2137–2156. https://doi.org/10.1109/TRO.2021.3072346

[12] Konstantinos A. Tsintotas, Loukas Bampis, and Antonios Gasteratos. 2022. The Revisiting Problem in Simultaneous Localization and Mapping: A Survey on Visual Loop Closure Detection. *IEEE Transactions on Intelligent Transportation Systems* 23, 11 (2022), 19929–19953. https://doi.org/10.1109/TITS.2022.3175656

[13] Hao Wei, Fulin Tang, Zewen Xu, Chaofan Zhang, and Yihong Wu. 2021. A Point-Line VIO System With Novel Feature Hybrids and With Novel Line Predicting-Matching. *IEEE Robotics and Automation Letters* 6, 4 (2021), 8681–8688. https://doi.org/10.1109/LRA.2021.3113987

[14] Max Wetterström and Patric Rönn. 2023. Virtual Reality over the Internet: An experimental study of common countermeasures when using VR applications over the Internet.

[15] Tianming Xie, Qifa Xu, Cuixia Jiang, Zhiwei Gao, and Xiangxiang Wang. 2024. A Robust Anomaly Detection Model for Pumps Based on the Spectral Residual With Self-Attention Variational Autoencoder. *IEEE Transactions on Industrial Informatics* 20, 6 (2024), 9059–9069. https://doi.org/10.1109/TII.2024.3381790

[16] Yuan Xu, Xingshuo Han, Gelei Deng, Jiwei Li, Yang Liu, and Tianwei Zhang. 2023. SoK: Rethinking sensor spoofing attacks against robotic vehicles from a systematic view. In *2023 IEEE 8th European Symposium on Security and Privacy (EuroS&P)*. IEEE, 1082–1100.

[17] Kota Yoshida, Masaya Hojo, and Takeshi Fujino. 2022. Adversarial scan attack against scan matching algorithm for pose estimation in lidar-based slam. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences* 105, 3 (2022), 326–335.

[18] Zichao Zhang and Davide Scaramuzza. 2018. A Tutorial on Quantitative Trajectory Evaluation for Visual(-Inertial) Odometry. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 7244–7251. https://doi.org/10.1109/IROS.2018.8593941