Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

HGRL-ATN: a hierarchical graph reinforcement learning framework for trustworthy UAV navigation under GPS spoofing attacks

Kiran Kumar Eepuri
Koneru Lakshmaiah Education Foundation
Rajasekhar Konari
Nadimpalli Satyanarayana Raju Institute of Technology
Sivavara Prasad J
Lakireddy Bali Reddy College of Engineering
Naishadh Vytla
Koneru Lakshmaiah Education Foundation
Hasane Ahammad Sk
Koneru Lakshmaiah Education Foundation
Mustafa SirElkhatim
mustafa1979@rsu.edu.sd
Red Sea University
Wallaaldin Eltayeb
Koneru Lakshmaiah Education Foundation

Article

Keywords: Unmanned aerial vehicles, GPS spoofing, autonomous navigation, deep reinforcement learning, hierarchical reinforcement learning, graph neural networks, sensor fusion, anomaly detection, cybersecurity

Posted Date: November 11th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11375683/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

HGRL-ATN: a hierarchical graph reinforcement learning framework for trustworthy UAV navigation under GPS spoofing attacks

Kiran Kumar Eepuri[1*], Rajasekhar Konari[2], Sivavara Prasad J[3], Naishadh Vytla[1], Hasane Ahammad Sk[1*], Mustafa SirElkhatim[4*], Wallaaldin Eltayeb[5]

[1*]Department of Electronics and Communication Engineering, Koneru Lakshmaiah Education Foundation, Guntur, Andhra Pradesh, India.
[2]Department of Electronics and Communication Engineering, Nadimpalli Satyanarayana Raju Institute of Technology, Visakhapatnam, Andhra Pradesh, India.
[3]Department of Electrical and Electronics Engineering, Lakireddy Bali Reddy College of Engineering, Mylavaram, Andhra Pradesh, India.
[4]Department of Chemical Engineering, Red Sea University, Port Sudan, 3315, Sudan.
[5]Department of Electrical and Electronics Engineering, Koneru Lakshmaiah Education Foundation, Guntur, Andhra Pradesh, India.

*Corresponding author(s). E-mail(s): kiraneepuri@kluniversity.in; ahammadklu@gmail.com; mustafa1979@rsu.edu.sd;
Contributing authors: rajkonari@gmail.com; janapatisivavaraprasad@gmail.com; vytlanaishadh2@gmail.com; walla129@gmail.com;

Abstract

Unmanned aerial vehicles (UAVs) that rely on the Global Positioning System (GPS) for localization are increasingly exposed to spoofing attacks that induce large positional deviations and mission failure. Established defense model-based estimators, Extended Kalman Filter (EKF) sensor fusion, and standard deep reinforcement learning (DRL)-are largely reactive, generalize poorly to unseen attack patterns, and do not model the relational structure that links heterogeneous sensor modalities. This paper presents HGRL-ATN, a framework that

couples graph-neural-network (GNN) sensor fusion, variational trust estimation, and hierarchical reinforcement learning (HRL) into a single end-to-end pipeline for secure UAV navigation. Sensors are represented as nodes of a graph and fused by a graph attention network, allowing cross-modal inconsistencies induced by spoofing to be detected relationally. A variational module assigns per-sensor trust and uncertainty scores that down-weight compromised inputs, while a two-level policy separates navigation strategy selection from continuous control. A digital-twin module predicts next-state dynamics to support anticipatory decision making. Evaluated on a synthetic UAV dataset, the VeReMi benchmark, and CARLA-generated scenarios, HGRL-ATN attains a detection accuracy of 97.0%, an F1-score of 0.965, and a navigation success rate of 94.8%, against 82.3%–90.4% for representative baselines. Ablation confirms that graph fusion and trust estimation are the dominant contributors, and trajectory analysis shows substantially reduced path deviation under active spoofing.

Keywords: Unmanned aerial vehicles, GPS spoofing, autonomous navigation, deep reinforcement learning, hierarchical reinforcement learning, graph neural networks, sensor fusion, anomaly detection, cybersecurity

1 Introduction

Advances in UAV technology have made autonomous operation practical across surveillance, precision agriculture, disaster response, and smart transportation. Accurate positioning underpins all of these applications, and GPS remains the dominant localization source despite being fundamentally vulnerable to adversarial manipulation [1–3]. GPS spoofing, in which an attacker transmits counterfeit satellite signals, is particularly damaging because it compromises the *integrity* of the position solution rather than merely its availability [4]. The distinction matters operationally: a jammed receiver reports that it has no fix and the vehicle can fall back to a safe mode, whereas a spoofed receiver reports a confident fix that happens to be wrong, and every downstream module continues to trust it.

Modern spoofers can synthesize signals that closely imitate authentic ones [5] and then drift the reported position slowly enough to evade threshold-based alarms. Field demonstrations have shown that a civilian UAV can be captured and steered along an attacker-chosen path without the flight controller registering a fault [6], and that commercial platforms can be safely hijacked using adaptive spoofing that tracks the victim's own state estimate [7]. The consequences range from trajectory divergence and mission abort to full vehicle hijack. Classical model-based and adaptive controllers offer strong stability guarantees but degrade sharply when the disturbance is adversarial rather than stochastic [8], because their robustness margins are derived under noise models that an intelligent attacker is free to violate.

Countermeasures fall into three broad families. Model-based methods, such as EKF-driven fusion, rely on a known system model and fail when several sensors are spoofed in a coordinated, mutually consistent way [9, 10]. Data-driven detectors trained on labeled attack traces achieve high accuracy in-distribution but generalize

poorly to unseen attacks [11, 12]. Learning-based control adapts online but, when used alone, treats sensing as a black box and provides no explicit notion of which measurement to believe.

The gap common to all three is the absence of a single formulation that jointly addresses sensor-level robustness, explicit trust reasoning, and trust-aware control. Operating safely under attack requires not only identifying that a measurement is unreliable but also altering the navigation decision in proportion to that unreliability. A detector that fires an alarm while the controller keeps acting on the corrupted estimate has not improved safety; it has only produced a log entry. Conventional fusion further ignores the relational structure among modalities, discarding precisely the cross-sensor evidence that exposes a spoofed channel: a GPS bias is hard to see against the GPS noise floor, but easy to see against a simultaneously observed inertial and visual estimate that disagrees with it.

This paper introduces HGRL-ATN (Hierarchical Graph Reinforcement Learning for Adaptive Trustworthy Navigation), which casts UAV navigation under attack as trust-aware, graph-structured decision making. The contributions are as follows.

- A graph attention formulation of multi-sensor fusion in which sensors are nodes and learned edge weights expose cross-modal inconsistencies caused by spoofing.
- A variational trust module that produces per-sensor reliability and uncertainty estimates, together with spoofing logits, and uses them to attenuate compromised inputs during control.
- A hierarchical policy in which a high-level controller selects a navigation regime and a low-level controller emits continuous commands, trained end-to-end with the perception and trust modules under a single composite objective.
- An empirical study across one synthetic and two benchmark datasets, including an ablation that isolates the contribution of each module.

The remainder of the paper is organized as follows. Section 2 reviews related work. Section 3 presents the HGRL-ATN architecture and its learning formulation. Section 3.9 describes the experimental setup, Section 4 reports and discusses results, and Section 6 concludes.

2 Related work

2.1 Spoofing Threat Model and Attack Capabilities

Spoofing differs from jamming in that the adversary imitates legitimate satellite signals rather than suppressing them, which makes it markedly harder to detect [1–3, 5]. Psiaki and Humphreys [4] organize the space into an attack/defense matrix, distinguishing simplistic replay from receiver-lock-aligned attacks in which the counterfeit signal is phase-matched to the authentic one before the victim is walked away from its true position. The severity of the threat has been established experimentally rather than only analytically: an attacker-controlled trajectory was imposed on a civilian UAV in flight [6], adaptive spoofing has been used to hijack consumer drones while respecting their internal consistency checks [7], and a systematic study of commodity platforms

characterized the conditions under which takeover succeeds [13]. Surveys of the UAV-specific threat landscape [14, 15] converge on the same conclusion: because the injected offset can be applied gradually, the victim can be driven far off course without any single residual exceeding its alarm threshold.

2.2 Signal- and Filter-Level Countermeasures

The earliest defenses operate at the receiver. Automatic gain control monitoring, carrier-phase and multi-antenna processing, and cryptographic signal authentication all raise the cost of a successful attack [1, 4], and dedicated anti-spoofing receiver designs can recover a usable position solution while an attack is in progress [16]. These defenses are effective but require specialized front-end hardware, which limits their applicability to the small, cost-constrained platforms that dominate commercial UAV deployment.

A second family works at the estimator level. Kalman filtering [17] and its nonlinear extensions provide a residual-the innovation sequence-whose statistics change when a measurement becomes inconsistent with the predicted state; CUSUM and chi-squared variants formalize this test [18], and innovation-based versions have been applied directly to GNSS/INS spoofing detection [10]. Tightly coupled inertial monitors extend this idea by exploiting the fact that a spoofer able to track the victim's position still struggles to remain consistent with its measured accelerations [19]. Pasqualetti *et al.* [8] give the limiting result for this family: attacks confined to the null space of the detector are structurally unobservable, so an adversary with sufficient knowledge of the estimator can construct an offset that produces no residual at all. The consequence for design is that residual-based monitoring cannot be the only line of defense, and that diversity across modalities is more valuable than sensitivity within one.

2.3 Learning-Based Spoofing Detection

Machine learning shifts the problem from residual thresholding to pattern discrimination. Early work applied support vector machines to receiver-level features [20]; subsequent studies broadened the feature set and the model class, using ensembles and conventional classifiers on GPS observables [21], information-theoretic divergence between nominal and observed signal statistics [22], and control-semantic features derived from the relationship between commanded and observed flight behavior [23]. Deep architectures have since been shown to improve detection on UAV traces, including capsule networks for receiver-feature classification [24] and deep induced-spoofing detectors trained on generic GNSS measurements [25]. Recognizing that a learned detector is itself an attack surface, Alhoraibi *et al.* [26] train against adversarially perturbed inputs.

Two limitations recur across this literature. First, detectors are typically evaluated in-distribution, and performance degrades when the attack profile at test time differs from the one seen during training [11, 12]. Second, and more consequential for autonomy, detection is treated as the terminal output. The system reports that an attack is under way but does not specify how the navigation policy should change, leaving the response to a hand-designed fallback that is tuned separately from the detector and therefore inherits none of its confidence information.

2.4 Sensor Fusion and State Estimation

Probabilistic fusion combines complementary sensors so that the weaknesses of one are covered by another, the classical example being GPS bounding the long-term drift of an inertial measurement unit (IMU) while the IMU bridges short GPS outages [9, 27]. Inertial errors accumulate without an external fix, so IMU-only navigation degrades quickly, which is why integrated GNSS/INS architectures dominate practical systems [27]. Learning-based fusion has since been shown to outperform fixed-gain filters when noise is non-Gaussian or adversarial [11, 12], and surveys of UAV navigation and control identify robustness of the fusion layer as the principal open problem for mission-critical autonomy [28].

What classical fusion does not provide is an explicit, per-channel statement of belief. A Kalman gain implicitly weights each measurement by its assumed covariance, but that covariance is a design parameter rather than a quantity inferred online from the current evidence. Under spoofing the assumed covariance is precisely what becomes wrong, and the filter continues to weight the compromised channel as though it were healthy.

2.5 Graph Learning for Relational Sensing

GNNs operate on data with explicit relational structure by propagating messages along edges [29, 30]. Representing each sensor as a node and each pairwise dependency as an edge makes cross-modal disagreement a first-class, learnable signal. This framing has proven effective for multivariate anomaly detection in cyber-physical systems, where Deng and Hooi [31] learn a sensor dependency graph and flag deviations from the learned inter-sensor relationships rather than from per-sensor bounds-directly the property that a coordinated spoofing attack must violate in order to remain internally consistent. Broader surveys of graph anomaly detection [32, 33] and of GNNs applied to time series [34] report consistent gains over models that treat channels independently.

Graph attention networks extend message passing by learning the weight of each neighbor rather than fixing it a priori [35], which allows the model to discount a compromised sensor adaptively and without a hand-specified rule. This property is what makes attention, rather than plain convolution over the sensor graph, the appropriate operator here: the attack changes which neighbors should be believed, and the weights must change with it. High-dimensional perception streams that feed such graphs are commonly encoded with convolutional or transformer backbones [36, 37], and temporal dependence in sensor sequences with recurrent models [38].

2.6 Reinforcement and Hierarchical Reinforcement Learning

Reinforcement learning (RL) derives a control policy from interaction rather than from an explicit model, which suits environments whose disturbance statistics are unknown [39, 40]. Recurrent and actor–critic variants extend RL to partial observability and continuous action spaces [41–43], and architectural refinements such as duelling value decomposition improve sample efficiency [44]. Applied to UAVs, deep

RL has produced obstacle-aware navigation policies that outperform planned trajectories in cluttered environments [45], while robustness-oriented formulations explicitly account for bounded perturbation of the observed state [46].

HRL decomposes a long-horizon task into a goal-setting level and an execution level, which shortens the effective credit-assignment path and improves exploration [47–49]. The decomposition is natural for navigation under attack: the regime decision-proceed nominally, enter a defensive mode, or fall back to GPS-denied operation-changes on a slower timescale than the actuator commands that implement it, and forcing both onto a single policy requires that policy to learn a discontinuous mapping from a continuous state. Existing HRL work, however, assumes trustworthy observations and therefore does not address what the high-level policy should do when the evidence supporting its own state estimate is suspect.

2.7 Trust, Uncertainty, and Rapid Adaptation

Trust and reputation systems formalize how an agent should weight information from sources of varying reliability, typically by maintaining a belief that is updated as evidence accumulates [50]. In a learning setting the analogous machinery is Bayesian or variational: attaching calibrated confidence to predictions by modeling the posterior over latent variables [51–54]. The distinction between aleatoric and epistemic uncertainty [51] is directly useful in a security context, because a noisy sensor and a spoofed sensor produce different signatures: the former raises irreducible observation noise, whereas the latter produces readings that are individually plausible but jointly improbable, surfacing as elevated model uncertainty.

Because novel attack variants appear faster than labeled corpora can be assembled, adaptation speed matters as much as in-distribution accuracy. Meta-learning addresses this by optimizing for parameters from which a few gradient steps suffice to fit a new task [55], and surveys of the area document its use for few-shot adaptation in precisely the low-label regimes that characterize zero-day attacks [56].

2.8 Digital Twins for Predictive Security

A digital twin maintains a synchronized virtual model of the physical system and compares predicted against observed behavior, which converts detection from a reactive test into an anticipatory one. Twin-based frameworks have been used to detect cyber-attacks in manufacturing and control systems by monitoring divergence between the model and the plant [57, 58]. Within a navigation stack the same mechanism yields a short-horizon forecast that lets a controller change regime before the positional error becomes large, provided the twin itself is conditioned on which measurements are currently believed.

Prior work treats fusion, detection, and control as separable stages, and the interfaces between them discard information: a detector emits a binary alarm, a fusion layer emits a point estimate, and the controller receives neither the per-channel confidence that produced them nor any gradient linking its own performance back to how that confidence was computed. HGRL-ATN differs in that the trust estimate produced by the perception graph is an input to the control policy and is optimized jointly with it,

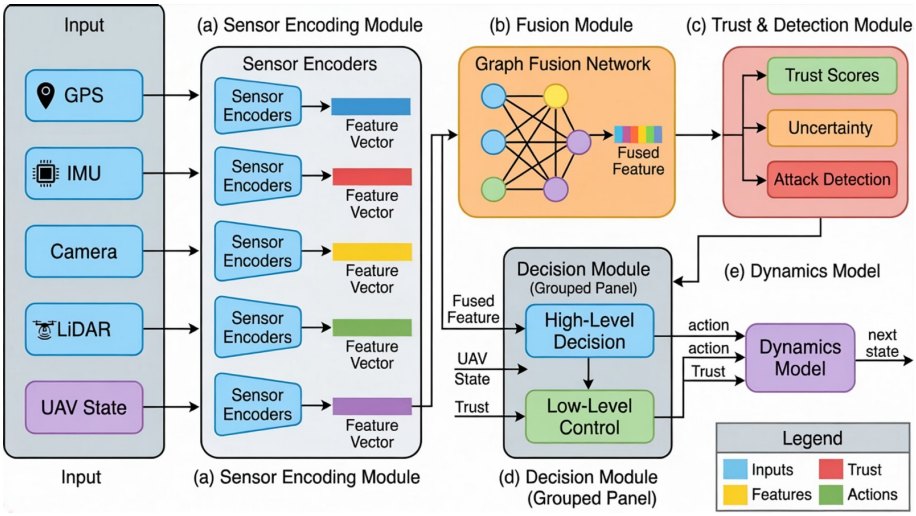


Fig. 1: Overall architecture of the proposed HGRL-ATN framework, comprising (a) the sensor encoding module, (b) the graph fusion module, (c) the trust and detection module, (d) the hierarchical decision module, and (e) the digital-twin dynamics model.

so the navigation behavior and the belief about sensor reliability are learned consistently rather than tuned independently. The digital twin is likewise conditioned on the trust vector rather than on the raw state, so its forecast degrades gracefully instead of propagating the spoofed channel forward.

3 Methods

3.1 Architecture Overview

The HGRL-ATN framework, shown in Fig. 1, comprises five modules: a per-sensor encoding module, a graph fusion module, a trust and detection module, a hierarchical decision module, and a digital-twin dynamics model.

At each time step the UAV supplies a state vector $s \in \mathbb{R}^{16}$ (position, velocity, and orientation components) together with four sensor readings: GPS $x_{gps} \in \mathbb{R}^8$, IMU $x_{imu} \in \mathbb{R}^{12}$, camera $x_{cam} \in \mathbb{R}^{32}$, and LiDAR $x_{lid} \in \mathbb{R}^{24}$. Each modality is mapped by a dedicated encoder into a common 64-dimensional space, and the four embeddings form the nodes of a complete graph that a graph attention network fuses into a single descriptor. A variational module converts that descriptor into per-sensor trust scores, uncertainty estimates, and spoofing logits. The hierarchical policy consumes the fused descriptor together with the trust signals: the high-level policy selects a navigation regime and the low-level policy emits continuous control commands. In parallel, the digital twin predicts the next state from the current state, action, and trust vector, providing a look-ahead signal for anticipatory adaptation. Algorithm 1 summarizes one training iteration.

The ordering of these modules is deliberate. Trust is estimated *after* graph fusion rather than before it, because the evidence that a channel is compromised is relational and therefore only becomes available once the modalities have been allowed to interact. Estimating trust from each sensor in isolation would reduce the module to a per-channel plausibility check of the kind that coordinated spoofing is designed to defeat.

3.2 Sensor Encoding

The four modalities differ in dimension, physical units, and noise characteristics, so they cannot be compared directly. Each sensor input $x_s \in \mathbb{R}^{d_s}$ is therefore encoded independently as

$$h_s = \mathrm{ReLU}(W_s x_s + b_s)\,, \qquad W_s \in \mathbb{R}^{64 \times d_s}, \tag{1}$$

where W_s and b_s are learnable. Because each modality has its own W_s, the network is free to learn a different scaling and feature basis per sensor, which is what allows raw readings to be used without hand-tuned normalization. Projecting heterogeneous modalities into a shared 64-dimensional latent space is what makes the subsequent graph operation well defined: attention between nodes requires that all nodes live in a common space. Stacking the four embeddings gives the initial node matrix $H^{(0)} \in \mathbb{R}^{4 \times 64}$.

3.3 Graph Attention Fusion

Sensors are treated as the vertices of a complete graph $G = (V, E)$, so every modality may attend to every other. A complete graph is used rather than a sparse hand-specified topology because the dependencies that matter under attack are not known in advance; with only four nodes the quadratic cost of full connectivity is negligible, and the attention mechanism is left to discover which edges carry information.

The unnormalized attention score between nodes i and j is

$$e_{ij} = \mathrm{LeakyReLU}\big(a^\top \left[W h_i \,\|\, W h_j\right]\big)\,, \tag{2}$$

with a a learnable attention vector and $\|$ denoting concatenation. The score is computed from the concatenated pair rather than from an inner product so that the relation can be asymmetric: how much the controller should let the inertial channel inform the GPS channel is not the same question as the reverse. Scores are normalized over the neighborhood of node i,

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k \in \mathcal{N}(i)} \exp(e_{ik})}, \tag{3}$$

so that the weights incident on each node form a distribution and no single sensor can dominate the aggregate simply by producing large activations. Node representations

are then updated by the attention-weighted aggregation

$$h_i' = \sigma\left(\sum_{j\in\mathcal{N}(i)} \alpha_{ij} W h_j\right), \tag{4}$$

where $\sigma(\cdot)$ is a nonlinear activation. Because α_{ij} is data dependent, a sensor whose readings disagree with the rest of the graph receives reduced weight without any hand-specified rule. This is the mechanism by which relational inconsistency, rather than absolute implausibility, drives spoofing detection: a slowly injected GPS bias remains well inside the plausible range for GPS alone, but it steadily reduces the attention that the remaining nodes assign to the GPS node.

Two attention heads are concatenated in the first layer, giving $H^{(1)} \in \mathbb{R}^{4\times 128}$; multiple heads let the layer track more than one relational pattern at once, which matters when an attack affects position and velocity consistency differently. A second layer returns $H^{(2)} \in \mathbb{R}^{4\times 64}$, giving each node a two-hop receptive field that spans the entire graph. The fused descriptor is obtained by mean pooling followed by a linear projection,

$$\bar{h} = \tfrac{1}{4}\sum_{i=1}^{4} H_i^{(2)}, \qquad f = W_f \bar{h} + b_f, \quad f \in \mathbb{R}^{128}. \tag{5}$$

Mean pooling is chosen over concatenation so that the descriptor is invariant to node ordering and so that the module can accommodate a different sensor complement without changing the downstream dimensions.

3.4 Variational Trust and Detection

The fused node features are flattened to $h_{\text{flat}} \in \mathbb{R}^{256}$ and encoded into a Gaussian latent with mean μ and log-variance $\log\sigma^2$. Predicting the log-variance rather than the variance keeps the parameterization unconstrained and numerically stable. The latent variable is drawn using the reparameterization trick, which moves the stochasticity into an external noise source so that gradients still flow through μ and σ:

$$z = \mu + \exp\!\left(\tfrac{1}{2}\log\sigma^2\right) \odot \epsilon, \qquad \epsilon \sim \mathcal{N}(0, I), \tag{6}$$

with $z \in \mathbb{R}^{64}$. Three heads read from the same latent:

$$\boldsymbol{\tau} = \sigma(W_\tau z + b_\tau), \quad u = \text{Softplus}(W_u z + b_u), \quad l = W_a z + b_a, \tag{7}$$

where $\boldsymbol{\tau} \in (0,1)^4$ are per-sensor trust scores, $u \in \mathbb{R}^4$ are uncertainty estimates, and $l \in \mathbb{R}^2$ are spoofing detection logits. The activations are chosen to match the range of each quantity: a sigmoid confines trust to the unit interval so it can be read as a weight, and a softplus keeps uncertainty strictly positive without the saturation that an exponential would introduce.

Sharing one latent across the three heads is what ties them together. Because $\boldsymbol{\tau}$, u, and l are all functions of z, the representation must explain trust, uncertainty,

and attack evidence simultaneously, and the three outputs cannot drift into mutually inconsistent conclusions. Modeling trust as a latent-variable posterior rather than a point estimate is what allows the system to distinguish a noisy sensor, which yields low trust with high uncertainty, from a spoofed one, which yields low trust with comparatively confident attack logits.

3.5 Digital-Twin Dynamics Model

A learned dynamics model predicts the next state from the current state, the applied action, and the trust vector,

$$\hat{s}_{t+1} = D_{dt}(s_t, a_t, \boldsymbol{\tau}) \,. \tag{8}$$

The inclusion of $\boldsymbol{\tau}$ as an argument is the distinguishing feature. A conventional dynamics model conditioned only on (s_t, a_t) would propagate whatever spoofed component is present in s_t forward in time, amplifying the attack rather than revealing it. Conditioning the prediction on $\boldsymbol{\tau}$ lets the twin discount the contribution of channels currently believed to be compromised, so the look-ahead remains usable during an active attack and the discrepancy between $\hat{s}_{t+1}$ and the observed s_{t+1} becomes an additional attack indicator.

3.6 Hierarchical Policy

The high-level policy maps the fused descriptor and trust signals to a categorical distribution over navigation regimes,

$$z_H = W_H^{(2)} \,\mathrm{ReLU}\Big(W_H^{(1)}[f; \boldsymbol{\tau}; u] + b_H^{(1)}\Big) + b_H^{(2)}, \tag{9}$$

$$P(a_H = i) = \frac{\exp(z_{H,i})}{\sum_j \exp(z_{H,j})}, \qquad a_H \in \{0, 1, 2\}, \tag{10}$$

where the three regimes correspond to nominal navigation, defensive navigation, and GPS-denied operation. Passing $\boldsymbol{\tau}$ and u to this level, and not only the fused descriptor f, is what makes the regime decision trust-aware: the policy can respond to a drop in confidence even when the fused state estimate still looks unremarkable.

The low-level policy is a squashed Gaussian over continuous commands, taking $[f; s; \bar{h}; \boldsymbol{\tau}] \in \mathbb{R}^{151}$ as input:

$$a = \tanh(\mu_L + \exp(\log \sigma_L) \odot \epsilon) \,, \quad \epsilon \sim \mathcal{N}(0, I), \tag{11}$$

with $a \in [-1, 1]^2$. A stochastic rather than deterministic policy preserves exploration during training and yields a distribution whose log-probability is required by the actor update. The tanh squashing enforces the actuator limits directly rather than through a penalty term, so the policy cannot propose an infeasible command that a saturation block would later have to clip.

Table 1: Architecture Components, Dimensions, and Design Rationale

Stage	Component	Input dim.	Output dim.	Operator	Role in the pipeline
Encoding	GPS encoder	8	64	Linear + ReLU	Projects position observables to the shared space
	IMU encoder	12	64	Linear + ReLU	Encodes accelerations and angular rates
	Camera encoder	32	64	Linear + ReLU	Encodes visual feature vector
	LiDAR encoder	24	64	Linear + ReLU	Encodes range returns
Fusion	GAT layer 1	4×64	4×128	2 attention heads	Learns pairwise sensor dependencies
	GAT layer 2	4×128	4×64	1 attention head	Extends receptive field to the full graph
	Fusion head	4×64	128	Mean pool + linear	Order-invariant descriptor f
Trust	VAE encoder	256	64	MLP $\rightarrow (\mu, \log \sigma^2)$	Latent posterior over sensor reliability
	Trust heads	64	$(4, 4, 2)$	Sigmoid / Softplus / linear	Emits $\boldsymbol{\tau}$, u, and spoofing logits l
Dynamics	Digital twin	150	16	MLP $(256, 128)$	Trust-conditioned next-state forecast
Control	High-level policy	136	3	Categorical	Selects navigation regime
	Low-level policy	151	2	Squashed Gaussian	Emits bounded continuous commands

The reward balances progress against risk,

$$R_t = r_{nav} - \lambda\, r_{risk}, \tag{12}$$

where r_{nav} rewards movement toward the goal, trajectory stability, and mission completion, while r_{risk} penalizes path deviation, unsafe states, and reliance on low-trust measurements. The last of these terms is what couples the reward back to the perception stack: acting on a channel the trust module has flagged is penalized even when the resulting trajectory happens to be acceptable, which discourages the policy from learning to ignore $\boldsymbol{\tau}$ during benign episodes. The coefficient λ sets the operating point: large λ produces conservative behavior that surrenders progress to avoid risk, small λ favors speed and tolerates exposure. Table 1 lists the dimensions of all components.

3.7 Training Objective

Detection is supervised by cross-entropy between the true spoofing label y and the predicted distribution $\hat{y}$ obtained from l:

$$\mathcal{L}_{det} = -\sum_i y_i \log \hat{y}_i. \tag{13}$$

This term alone would be satisfied by any representation that separates the two classes, including one that ignores the graph entirely, so it is paired with the constraints below.

Algorithm 1 HGRL-ATN Training Iteration

Input: Sensor streams $\{x_{gps}, x_{imu}, x_{cam}, x_{lid}\}$, state s_t, label y
Output: Updated parameters $\theta_e, \theta_g, \theta_v, \theta_\pi, \theta_V$

Sensor encoding

$h_i \leftarrow \mathrm{ReLU}(W_i x_i + b_i), \quad h_i \in \mathbb{R}^{64}$

Graph attention fusion

$e_{ij} \leftarrow \mathrm{LeakyReLU}\big(a^\top [W h_i \| W h_j]\big)$

$\alpha_{ij} \leftarrow \exp(e_{ij}) \big/ \sum_k \exp(e_{ik})$

$h'_i \leftarrow \sigma\Big(\sum_j \alpha_{ij} W h_j\Big)$

$f \leftarrow W_f\big(\frac{1}{4}\sum_i h'_i\big) + b_f$

Variational trust estimation

$z \leftarrow \mu + \exp\big(\frac{1}{2}\log\sigma^2\big) \odot \epsilon, \quad \epsilon \sim \mathcal{N}(0, I)$

$\boldsymbol{\tau} \leftarrow \sigma(W_\tau z), \quad u \leftarrow \mathrm{Softplus}(W_u z), \quad l \leftarrow W_a z$

Hierarchical decision

$a_H \sim \pi_H(\cdot \mid f, \boldsymbol{\tau}, u)$

$a_t \sim \pi_L(\cdot \mid f, s_t, \boldsymbol{\tau})$

Environment transition and forecast

Execute a_t; observe s_{t+1}, $R_t \leftarrow r_{nav} - \lambda r_{risk}$

$\hat{s}_{t+1} \leftarrow D_{dt}(s_t, a_t, \boldsymbol{\tau})$

Loss evaluation

$\mathcal{L}_{det} \leftarrow -\sum_i y_i \log \hat{y}_i$

$\mathcal{L}_{KL} \leftarrow -\frac{1}{2}\sum\big(1 + \log\sigma^2 - \mu^2 - \sigma^2\big)$

$\mathcal{L}_{trust} \leftarrow \|\boldsymbol{\tau} - \boldsymbol{\tau}^*\|^2, \quad \mathcal{L}_{dt} \leftarrow \|\hat{s}_{t+1} - s_{t+1}\|^2$

$G_t \leftarrow \sum_{k\geq 0} \gamma^k R_{t+k}, \quad A_t \leftarrow G_t - V(s_t)$

$\mathcal{L}_{policy} \leftarrow -\mathbb{E}[\log \pi(a_t|s_t) A_t], \ \mathcal{L}_{value} \leftarrow \|V(s_t) - G_t\|^2$

$\mathcal{L} \leftarrow \mathcal{L}_{det} + \lambda_1 \mathcal{L}_{KL} + \lambda_2 \mathcal{L}_{trust} + \lambda_3 \mathcal{L}_{dt} + \lambda_4 \mathcal{L}_{policy} + \lambda_5 \mathcal{L}_{value}$

Update

Clip $\|\nabla\theta\| \leq 1.0$; update all parameters with Adam

The latent space is regularized toward a standard normal prior by

$$\mathcal{L}_{KL} = -\tfrac{1}{2}\sum\left(1 + \log\sigma^2 - \mu^2 - \sigma^2\right), \tag{14}$$

which keeps the trust posterior smooth and prevents the module from collapsing to a deterministic gate. Without this term the variance head is free to shrink toward zero, at which point the uncertainty output stops carrying information and the module degenerates into a hard classifier.

Trust estimates are anchored to reference values $\boldsymbol{\tau}^*$ derived from the ground-truth attack labels,

$$\mathcal{L}_{trust} = \|\boldsymbol{\tau} - \boldsymbol{\tau}^*\|^2, \tag{15}$$

which grounds the trust scale so that a given value of $\boldsymbol{\tau}$ means the same thing across episodes. Without an anchor the policy and the trust head could co-adapt to an

arbitrary internal convention, leaving $\boldsymbol{\tau}$ uninterpretable to a human operator. The digital twin is trained on one-step prediction error,

$$\mathcal{L}_{dt} = \left\| \hat{s}_{t+1} - s_{t+1} \right\|^2 . \tag{16}$$

For the control objective, the discounted return and the advantage are

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k}, \qquad A_t = G_t - V(s_t), \tag{17}$$

where subtracting the value baseline removes the state-dependent component of the return and leaves only the part attributable to the action, which markedly reduces gradient variance. This gives the actor and critic losses

$$\mathcal{L}_{policy} = -\mathbb{E}[\log \pi(a_t \mid s_t) A_t] , \quad \mathcal{L}_{value} = \left\| V(s_t) - G_t \right\|^2 . \tag{18}$$

The high-level policy receives no separate supervision; it is updated through the gradients propagated by the reward and value signals, so the regime boundaries are discovered from their consequences rather than prescribed.

All terms are combined into a single objective,

$$\mathcal{L} = \mathcal{L}_{det} + \lambda_1 \mathcal{L}_{KL} + \lambda_2 \mathcal{L}_{trust} + \lambda_3 \mathcal{L}_{dt} + \lambda_4 \mathcal{L}_{policy} + \lambda_5 \mathcal{L}_{value}, \tag{19}$$

where $\lambda_1, \ldots, \lambda_5$ weight the contribution of each component. Optimizing (19) jointly-rather than training perception, trust, and control in isolation-is what allows the trust estimate to be shaped by its downstream effect on navigation reward. A detector trained only against (13) optimizes classification accuracy; a trust module trained inside (19) optimizes the quantity that actually matters, namely whether acting on the estimate produces a safe trajectory.

3.8 Training Schedule

Training the composite objective from random initialization is unstable, because the policy gradient is computed from a fused descriptor and a trust vector that are themselves meaningless early on, and the resulting reward signal is uninformative. Training therefore proceeds in two phases.

In the first, the encoders, the graph fusion layers, and the trust and detection heads are trained on labeled data against the supervised subset of the objective,

$$\mathcal{L}_{sup} = \mathcal{L}_{det} + \lambda_1 \mathcal{L}_{KL} + \lambda_2 \mathcal{L}_{trust}, \tag{20}$$

with gradients clipped to $\|\nabla\theta\| \leq 1.0$ for stability. Clipping matters here because the KL term can produce large transient gradients whenever the variance head moves quickly.

In the second phase the hierarchical policy is trained by actor–critic reinforcement learning [59] while the perception stack continues to receive gradients, so that all

modules are finally optimized jointly under (19). Allowing perception to keep adapting during this phase is what closes the loop described above; freezing it would reduce the second phase to fitting a controller on top of a fixed detector, which is the decoupled arrangement the framework is intended to replace. Convergence is monitored through cumulative reward and the individual loss terms.

3.9 Experimental setup

3.9.1 Implementation

All experiments were implemented in Python with PyTorch 2.1 and executed on a workstation with an NVIDIA RTX 3090 GPU (24 GB), an Intel i9 CPU, and 32 GB of RAM. Training used CUDA with mixed-precision (FP16) arithmetic.

3.9.2 Datasets

Three datasets were used, spanning controlled and realistic conditions. The synthetic set provides labeled ground truth for the attack indicator, which no public UAV corpus supplies at the per-sensor level, while the two external benchmarks test whether the learned relational criterion transfers beyond the noise model used to construct it.

Synthetic UAV dataset. Each sample consists of a 16-dimensional UAV state and the four sensor readings defined in Section 3, with a binary label $y \in \{0, 1\}$ indicating spoofing. States are drawn as $s \sim \mathcal{U}([-1, 1]^{16})$ and labels as $y \sim \text{Bernoulli}(0.35)$. The attack rate is set below one half deliberately: a balanced split would overstate the prior probability of attack and let a detector achieve apparent competence by over-predicting the positive class, whereas intermittent attacks reproduce the operational regime in which false alarms are costly.

The GPS channel is generated as

$$x_{gps} = \begin{cases} s_{1:8} + \eta, & y = 0, \\ s_{1:8} + \eta + \delta, & y = 1, \end{cases} \tag{21}$$

with measurement noise $\eta \sim \mathcal{N}(0, 0.08^2 I)$ and a spoofing perturbation $\delta \sim \mathcal{N}(1.0, 0.4^2 I)$. The two components of δ model distinct aspects of the attack: the non-zero mean reproduces the systematic bias that walks the vehicle off course, while the inflated variance reproduces the imperfect tracking of a spoofer that must continually re-align its counterfeit signal with the victim's motion. Because δ and η overlap in support, individual spoofed samples remain plausible under the benign model, which is precisely the condition that defeats per-channel thresholding and forces the detector to rely on cross-sensor evidence.

The IMU channel is $x_{imu} = s_{3:14} + \epsilon$ with $\epsilon \sim \mathcal{N}(0, 0.12^2 I)$, the larger variance reflecting inertial drift and bias. Critically, the IMU is left uncorrupted when $y = 1$, so the attack creates a genuine disagreement between the position and inertial channels rather than a globally shifted state. Camera and LiDAR channels are drawn uniformly over $[-1, 1]$ with additive Gaussian noise representing illumination and range disturbances.

Table 2: Dataset Summary

Dataset	Type	Modalities	Samples
Synthetic UAV	Simulated	GPS, IMU, camera, LiDAR, state	10 000
VeReMi [60]	Benchmark	Vehicular communication logs	50 000+
CARLA [61]	Simulation	Multi-modal sensor data	30 000+

Temporal evolution follows the linear model

$$s_{t+1} = 0.85\, s_t + \epsilon, \qquad \epsilon \sim \mathcal{N}(0, 0.05^2 I), \tag{22}$$

where the scaling factor represents system inertia and ϵ is process noise. The contraction toward the origin keeps trajectories bounded over long episodes without an explicit clipping rule, and this model also supplies ground truth for the digital-twin loss (16). Raw readings are fed to the network without additional normalization, leaving the per-sensor encoders of (1) to learn their own scaling.

VeReMi. The Vehicular Reference Misbehavior dataset [60] provides labeled benign and malicious message logs generated from a city-scale simulation. Its position-falsification attacks are structurally analogous to GPS spoofing in UAV networks, making it a standard external benchmark for generalization.

CARLA. The CARLA simulator [61] was used to generate multi-modal navigation scenarios with obstacles, varied weather, and injected adversarial conditions.

3.9.3 Training configuration

The model is optimized with Adam [62] using $\beta_1 = 0.9$, $\beta_2 = 0.999$, and an initial learning rate $\eta = 1 \times 10^{-4}$. The conservative learning rate reflects the composite objective: because six loss terms share a parameter set, a larger step size lets whichever term currently dominates pull the shared encoders away from a solution that serves the others. The batch size is $B = 64$ and gradients are clipped to $\|\nabla\theta\| \leq 1.0$. Training runs for 100 epochs with early stopping on validation loss. Each sensor embedding is 64-dimensional, the graph fusion stack uses two attention layers, the variational latent space is 64-dimensional, and the digital twin is a two-layer MLP with 256 and 128 hidden units.

For the reinforcement learning stage, the hierarchical policy is trained with an actor–critic scheme [59]; deterministic policy gradient methods have likewise proven effective for continuous control [43]. The discount factor is $\gamma = 0.99$, which weights long-horizon outcomes heavily enough to favor goal-directed, safety-preserving behavior-a lower discount would make the agent locally greedy and blind to the accumulating cost of a drifting trajectory. An episode terminates when the UAV reaches its target or a step budget is exhausted, the latter preventing unbounded episodes from distorting the return estimate.

3.9.4 Evaluation protocol

Data are split 70%/15%/15% into training, validation, and test partitions. Detection is assessed by accuracy, precision, recall, and F1-score computed from the confusion matrix on held-out data. Precision and recall are reported alongside accuracy because the class prior is unbalanced by construction, and F1 is used as the summary statistic since the two error types carry different operational costs: a missed attack risks the mission, while a false alarm needlessly surrenders performance to a defensive regime.

Navigation is assessed by cumulative reward $R = \sum_{t=0}^{T} R_t$ over an episode and by success rate, the fraction of episodes in which the UAV reaches its target. The two are reported together because they capture different failure modes-an agent can accumulate reward while never quite reaching the goal, and an agent can reach the goal by an inefficient or unsafe path. All metrics are measured both under nominal conditions and under active spoofing. Every experiment was repeated three times and averaged results are reported.

4 Results

4.1 Spoofing Detection

Fig. 2 summarizes detection behavior. Training and test losses decrease smoothly and converge after roughly 40–60 iterations, and the small gap between them indicates that the model generalizes rather than memorizes (Fig. 2(a)). The feature response in Fig. 2(b) separates spoofed from nominal intervals clearly, confirming that the fused multi-sensor representation carries the discriminative signal; the separation is visible in the fused descriptor even though no individual channel is separable on its own, which is the empirical counterpart of the argument made in Section 3. Fig. 2(c) shows the attack probability under threshold-based classification: attack intervals are recovered with high confidence while false alarms during nominal operation remain rare. The corresponding accuracy trajectory is given in Fig. 3.

The normalized confusion matrices in Fig. 4 show strong diagonal dominance on all three datasets, with low false-positive and false-negative rates. Performance is highest on the synthetic data and degrades modestly on VeReMi and CARLA, which is the expected ordering given the increasing realism and distribution shift of those benchmarks. The residual errors are concentrated in the early portion of each attack interval, where the injected bias is still small relative to the sensor noise floor and the graph has not yet accumulated enough disagreement to reduce the attention assigned to the compromised node.

4.2 Navigation Performance

Fig. 5 shows cumulative reward over training episodes. The agent converges stably and settles at a higher plateau than the baselines. Fig. 6 compares trajectories under active spoofing: HGRL-ATN tracks the intended path closely, whereas baseline controllers accumulate substantial lateral deviation once the injected bias grows. The mechanism is visible in the shape of the curves-the baselines deviate monotonically because they continue to treat the spoofed position as authoritative, while the proposed method

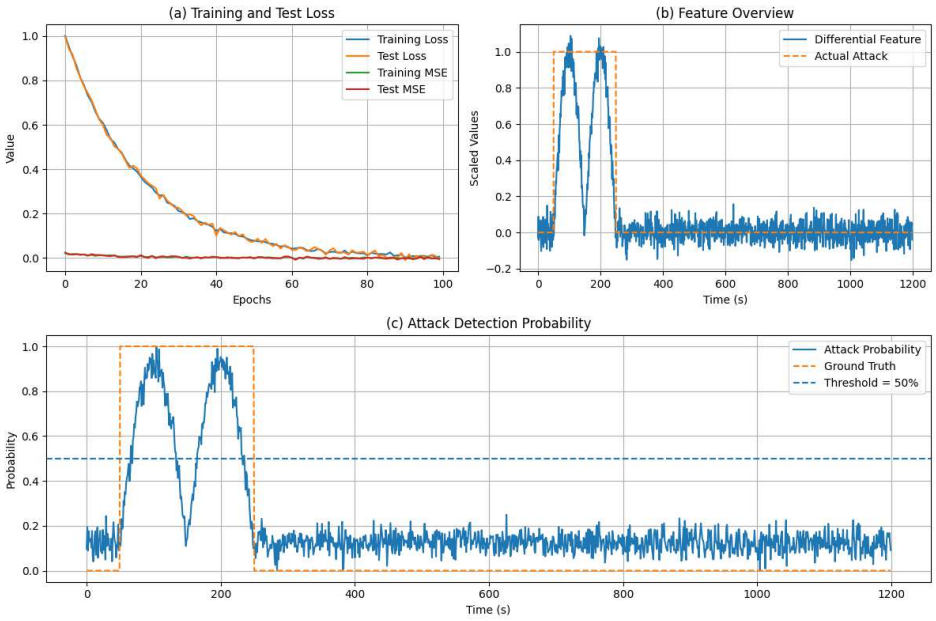
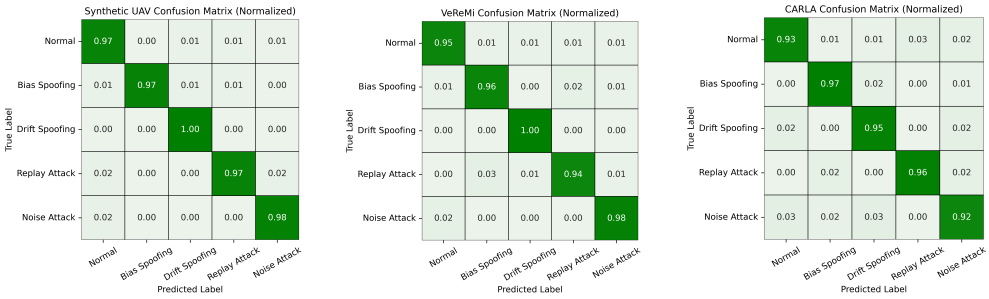


Fig. 2: Detection behavior of HGRL-ATN: (a) training and test loss convergence, (b) feature response under spoofing conditions, and (c) attack detection probability with threshold-based classification.



Fig. 3: Training and validation accuracy over epochs.

exhibits a bounded excursion followed by recovery, corresponding to the point at which the high-level policy leaves the nominal regime. Table 3 reports per-dataset detection and navigation metrics.



(a) Synthetic UAV (b) VeReMi (c) CARLA

Fig. 4: Normalized confusion matrices for the synthetic UAV, VeReMi, and CARLA datasets. Diagonal dominance across all three indicates consistent spoofing detection under increasing realism.



Fig. 5: Cumulative reward over training episodes.

Table 3: Performance Comparison Across Datasets

Dataset	Precision	Recall	F1	SR (%)	Reward
Synthetic UAV	0.97	0.96	0.965	94.8	185.6
VeReMi [60]	0.95	0.94	0.945	92.3	172.4
CARLA [61]	0.93	0.92	0.925	90.7	160.8

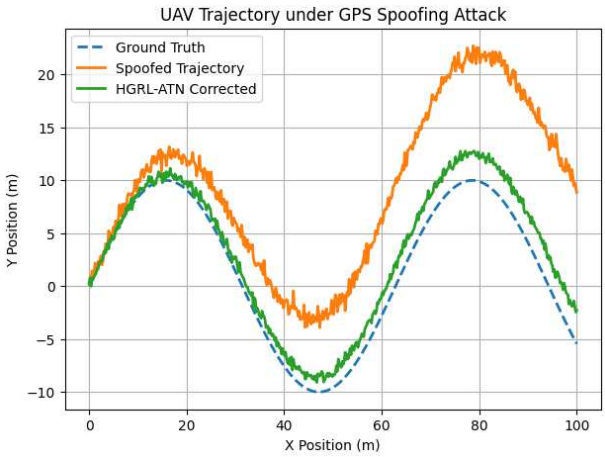


Fig. 6: Trajectory comparison under GPS spoofing.

Table 4: Comparison with State-of-the-Art Methods

Method	Dataset	Accuracy	Precision	Recall	F1	SR (%)	Cumulative reward
EKF-based fusion [9]	VeReMi	0.88	0.86	0.85	0.855	82.3	120.5
CNN-based detection [36]	Synthetic	0.91	0.90	0.89	0.895	86.7	140.2
DQN navigation [39]	CARLA	0.90	0.89	0.88	0.885	84.5	135.6
LSTM anomaly detection [11]	VeReMi	0.92	0.91	0.90	0.905	87.2	150.8
HRL-based navigation [47]	CARLA	0.93	0.92	0.91	0.915	89.1	162.3
GNN-based fusion [35]	Synthetic	0.94	0.93	0.92	0.925	90.4	168.7
Proposed HGRL-ATN	Multi-dataset	**0.97**	**0.97**	**0.96**	**0.965**	**94.8**	**185.6**

4.3 Comparison with the State of the Art

Fig. 7 and Table 4 compare HGRL-ATN against representative baselines drawn from each methodological family. EKF-based fusion is the weakest, consistent with the observation that a purely model-based residual test cannot separate a coherent spoofing bias from a modeling error [8]. Purely discriminative detectors (CNN, LSTM) improve detection but do not close the navigation gap, since detection alone does not tell the controller what to do. RL and HRL baselines improve navigation without improving detection, for the symmetric reason: a better policy acting on a corrupted state estimate is still acting on a corrupted state estimate. GNN-based fusion is the strongest single-mechanism baseline, which supports the central claim that relational structure carries most of the detection signal. HGRL-ATN exceeds all of them on every metric, indicating that the gains from graph fusion, trust estimation, and hierarchical control are complementary rather than redundant.

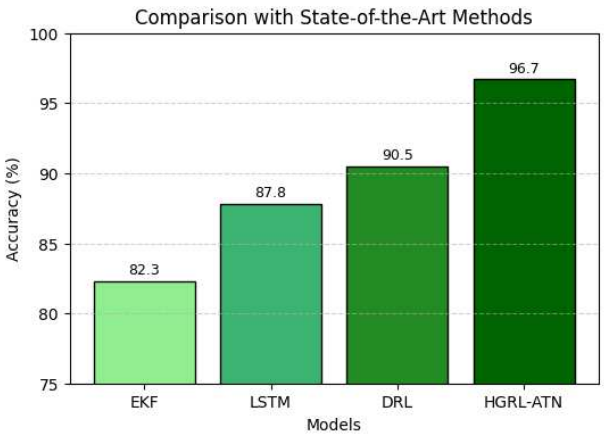


Fig. 7: Comparison with state-of-the-art methods.

Table 5: Ablation Study of HGRL-ATN Components

Variant	Precision	Recall	F1	SR (%)	Reward
Full model (HGRL-ATN)	0.97	0.96	0.965	94.8	185.6
w/o graph fusion	0.92	0.90	0.910	88.2	155.3
w/o trust module	0.91	0.89	0.900	86.7	148.9
w/o digital twin	0.93	0.91	0.920	89.5	162.7
w/o hierarchical RL	0.90	0.88	0.890	85.9	140.5

4.4 Ablation Study

Table 5 isolates the contribution of each module. Removing graph fusion causes the largest single drop in both detection and navigation, confirming that inter-sensor dependencies-not per-sensor statistics-carry most of the discriminative evidence. Removing the variational trust module produces a comparable loss and degrades precision and recall together, which is consistent with the controller no longer being able to discount compromised inputs: without $\boldsymbol{\tau}$, the fused descriptor still contains the spoofed contribution and the policy has no signal telling it to disregard that component.

Removing hierarchical control leaves detection largely intact but reduces success rate and cumulative reward most sharply, showing that its contribution lies in acting on the trust signal rather than producing it. This asymmetry is informative for deployment: the perception stack and the control stack fail in different ways, and neither substitutes for the other. The digital twin contributes the smallest margin, as expected for a module that supplies anticipatory rather than corrective information; its value appears mainly in the timing of regime switches rather than in whether they eventually occur.

5 Discussion

The framework remains effective across both gradual-drift and abrupt-manipulation spoofing profiles, which stress the detector differently: an abrupt offset is easy to detect but leaves little time to react, while a gradual drift allows ample reaction time but is hard to distinguish from benign motion. The redundancy of the sensor graph limits the impact of individual sensor failure, and performance does not collapse when the spoofed channel is the dominant positioning source.

Transfer from the synthetic data to VeReMi and CARLA costs roughly two to four points of F1, indicating that the learned relational criterion is not tied to the specific noise model used in training. This is the expected consequence of learning a criterion over sensor *relationships* rather than over sensor values: the absolute statistics change across datasets, but the requirement that independent modalities agree does not. The digital twin supplies a short-horizon forecast that allows the high-level policy to switch regime before the deviation becomes large, which is visible in Fig. 6 as an earlier correction onset relative to the baselines.

Two limitations bound these conclusions. First, the synthetic dataset uses a linear transition model (22) and independent per-channel noise, so it does not capture aerodynamic coupling or correlated multi-sensor attacks; an adversary able to spoof GPS and visual odometry consistently would weaken the relational signal the method depends on. Second, VeReMi is a vehicular rather than aerial benchmark, and its use here rests on the structural similarity of position-falsification attacks rather than on matched platform dynamics.

6 Conclusion

This paper presented HGRL-ATN, a framework that unifies GPS spoofing detection and trustworthy navigation for UAVs. In contrast to model-based pipelines that are vulnerable to coordinated sensor manipulation [5, 8], the proposed method represents sensors as a graph, estimates their trustworthiness variationally, and conditions a hierarchical control policy on those estimates, with all components optimized under a single objective.

Across three datasets the framework attains a detection accuracy of 97.0% with precision 0.97, recall 0.96, and F1-score 0.965, and a navigation success rate of 94.8% under spoofing, compared with 82.3%–90.4% for the baselines. The ablation identifies graph fusion and trust estimation as the dominant contributors to detection, and hierarchical control as the dominant contributor to navigation outcome. Qualitatively, the trust mechanism makes degradation graceful: as confidence in the GPS channel falls, the high-level policy shifts regime rather than continuing to act on a corrupted state estimate.

The broader observation is that detection and control should not be designed separately for adversarial autonomy. An alarm that does not change behavior offers no protection, and a controller that cannot express doubt about its own inputs has no basis on which to change. Coupling the two through a shared, differentiable trust representation is what produces the margin reported here.

Future work will address deployment on physical UAV platforms, adaptation of the model for edge-class compute budgets, and extension to cooperative multi-UAV settings in which trust must be reasoned about across vehicles as well as across sensors.

Data availability. The VeReMi dataset is publicly available at https://veremi-dataset.github.io/. The CARLA simulator is available at https://carla.org/ with source code at https://github.com/carla-simulator/carla. The synthetic UAV dataset was generated according to the procedure specified in Section 3.9 and can be reproduced from the equations given there.

Code availability. Code supporting the findings of this study is available from the corresponding author on reasonable request.

7 Author contributions

K.K.E.: Conceptualization, Methodology, Software, Data curation, Formal analysis, Investigation, Visualization, Writing – original draft; N.V. & R.K.: Software, Data curation, Validation, Investigation, Visualization; S.P.J. & H.A.Sk..: Methodology, Validation, Formal analysis, Writing – review & editing; Mustafa & W.E.: Conceptualization, Methodology, Supervision, Validation, Writing – review & editing.

Declarations

- **Funding:** This work received no dedicated funding from any public, commercial, or not-for-profit organization.
- **Competing interests:** The authors declare no competing interests.
- **Ethics approval and consent to participate:** Not applicable; this study involved no human participants or animals.

References

[1] Yuan, M., Tang, X. & Ou, G. Authenticating GNSS civilian signals: a survey. *Satellite Navigation* **4**, 1–18 (2023).

[2] Kim, C., Chang, S.-Y., Lee, D. & Kim, J. Reliable detection of location spoofing and variation attacks. *IEEE Access* **11**, 10813–10825 (2023).

[3] Jansen, K. *et al. Crowd-GPS-Sec: leveraging crowdsourcing to detect and localize GPS spoofing attacks.* in *Proceedings of the IEEE Symposium on Security and Privacy*, 1018–1031 (IEEE, 2018).

[4] Psiaki, M. L. & Humphreys, T. E. GNSS spoofing and detection. *Proceedings of the IEEE* **104**, 1258–1270 (2016).

[5] Chen, Q., Li, G., Liu, P. *et al.* Anomaly detection and secure position estimation against GPS spoofing attack: a security-critical study of localization in

autonomous driving. *IEEE Transactions on Vehicular Technology* **74**, 87–99 (2025).

[6] Kerns, A. J., Shepard, D. P., Bhatti, J. A. & Humphreys, T. E. Unmanned aircraft capture and control via GPS spoofing. *Journal of Field Robotics* **31**, 617–636 (2014).

[7] Noh, J. *et al.* Tractor beam: safe-hijacking of consumer drones with adaptive GPS spoofing. *ACM Transactions on Privacy and Security* **22**, 1–26 (2019).

[8] Pasqualetti, F., Dörfler, F. & Bullo, F. Attack detection and identification in cyber-physical systems. *IEEE Transactions on Automatic Control* **58**, 2715–2729 (2013).

[9] Thrun, S., Burgard, W. & Fox, D. *Probabilistic Robotics* (MIT Press, Cambridge, MA, 2005).

[10] Liu, Y., Li, S., Fu, Q., Liu, Z. & Zhou, Q. Analysis of Kalman filter innovation-based GNSS spoofing detection method for INS/GNSS integrated navigation system. *IEEE Sensors Journal* **19**, 5167–5178 (2019).

[11] Boukerche, A., Zheng, L. & Alfandi, O. Outlier detection: methods, models, and classification. *ACM Computing Surveys* **53**, 1–37 (2020).

[12] Luo, Y., Xiao, Y., Cheng, L., Peng, G. & Yao, D. Deep learning-based anomaly detection in cyber-physical systems: progress and opportunities. *ACM Computing Surveys* **54**, 1–36 (2021).

[13] Sathaye, H., Strohmeier, M., Lenders, V. & Ranganathan, A. *An experimental study of GPS spoofing and takeover attacks on UAVs.* in *Proceedings of the 31st USENIX Security Symposium*, 3503–3520 (USENIX Association, 2022).

[14] Restivo, R. D. *et al. GPS spoofing on UAV: a survey.* in *Proceedings of the IEEE INFOCOM Workshops*, 1–6 (IEEE, 2023).

[15] Radoš, K., Brkić, M. & Begušić, D. Recent advances on jamming and spoofing detection in GNSS. *Sensors* **24**, 4210 (2024).

[16] Sathaye, H., LaMountain, G., Closas, P. & Ranganathan, A. *SemperFi: anti-spoofing GPS receiver for UAVs.* in *Proceedings of the Network and Distributed System Security Symposium (NDSS)* (Internet Society, 2022).

[17] Kalman, R. E. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering* **82**, 35–45 (1960).

[18] Murguia, C. & Ruths, J. *CUSUM and chi-squared attack detection of compromised sensors.* in *Proceedings of the IEEE Conference on Control Applications (CCA)*, 474–480 (IEEE, 2016).

[19] Tanil, c., Khanafseh, S., Joerger, M. & Pervan, B. An INS monitor to detect GNSS spoofers capable of tracking vehicle position. *IEEE Transactions on Aerospace and Electronic Systems* **54**, 131–143 (2018).

[20] Panice, G. *et al. A SVM-based detection approach for GPS spoofing attacks to UAV.* in *Proceedings of the 23rd International Conference on Automation and Computing (ICAC)*, 1–11 (IEEE, 2017).

[21] Shafique, A., Mehmood, A. & Elhadef, M. Detecting signal spoofing attack in UAVs using machine learning models. *IEEE Access* **9**, 93803–93815 (2021).

[22] Basan, E. *et al.* GPS-spoofing attack detection technology for UAVs based on Kullback–Leibler divergence. *Drones* **6**, 8 (2021).

[23] Wei, X., Sun, C., Lyu, M., Song, Q. & Li, Y. ConstDet: control semantics-based detection for GPS spoofing attacks on UAVs. *Remote Sensing* **14**, 5587 (2022).

[24] Khoei, T. T., Al Shamaileh, K., Devabhaktuni, V. K. & Kaabouch, N. Performance analysis of capsule networks for detecting GPS spoofing attacks on unmanned aerial vehicles. *International Journal of Information Security* **24** (2025).

[25] Iqbal, A., Aman, M. N. & Sikdar, B. A deep learning based induced GNSS spoof detection framework. *IEEE Transactions on Machine Learning in Communications and Networking* **2**, 457–478 (2024).

[26] Alhoraibi, L., Alghazzawi, D. & Alhebshi, R. Detection of GPS spoofing attacks in UAVs based on adversarial machine learning model. *Sensors* **24**, 6156 (2024).

[27] Groves, P. D. *Principles of GNSS, Inertial, and Multisensor Integrated Navigation Systems* 2nd edn (Artech House, Norwood, MA, 2013).

[28] Marshall, J. A., Sun, W. & L'Afflitto, A. A survey of guidance, navigation, and control systems for autonomous multi-rotor small unmanned aerial systems. *Annual Reviews in Control* **52**, 390–427 (2021).

[29] Zhou, J. *et al.* Graph neural networks: a review of methods and applications. *AI Open* **1**, 57–81 (2020).

[30] Wu, Z. *et al.* A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems* **32**, 4–24 (2021).

[31] Deng, A. & Hooi, B. *Graph neural network-based anomaly detection in multivariate time series.* in *Proceedings of the AAAI Conference on Artificial Intelligence*, 4027–4035 (AAAI Press, 2021).

[32] Ma, X. *et al.* A comprehensive survey on graph anomaly detection with deep learning. *IEEE Transactions on Knowledge and Data Engineering* **35**, 12012–12038 (2023).

[33] Kim, H., Lee, B. S., Shin, W.-Y. & Lim, S. Graph anomaly detection with graph neural networks: current status and challenges. *IEEE Access* **10**, 111820–111829 (2022).

[34] Jin, M. *et al.* A survey on graph neural networks for time series: forecasting, classification, imputation, and anomaly detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2024).

[35] Veličković, P. *et al. Graph attention networks.* in *Proceedings of the International Conference on Learning Representations (ICLR)*, 1–12 (ICLR, 2018).

[36] He, K., Zhang, X., Ren, S. & Sun, J. *Deep residual learning for image recognition.* in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770–778 (IEEE, 2016).

[37] Dosovitskiy, A. *et al. An image is worth 16x16 words: transformers for image recognition at scale.* in *Proceedings of the International Conference on Learning Representations (ICLR)*, 1–21 (ICLR, 2021).

[38] Hochreiter, S. & Schmidhuber, J. Long short-term memory. *Neural Computation* **9**, 1735–1780 (1997).

[39] Sutton, R. S. & Barto, A. G. *Reinforcement Learning: An Introduction* 2nd edn (MIT Press, Cambridge, MA, 2018).

[40] Bellman, R. *Dynamic Programming* (Princeton University Press, Princeton, NJ, 1957).

[41] Hausknecht, M. & Stone, P. *Deep recurrent Q-learning for partially observable MDPs.* in *Proceedings of the AAAI Fall Symposium Series*, 29–37 (AAAI Press, 2015).

[42] Haarnoja, T., Zhou, A., Abbeel, P. & Levine, S. *Soft actor-critic: off-policy maximum entropy deep reinforcement learning with a stochastic actor.* in *Proceedings of the 35th International Conference on Machine Learning (ICML)*, 1861–1870 (PMLR, 2018).

[43] Lillicrap, T. P. *et al. Continuous control with deep reinforcement learning.* in *Proceedings of the International Conference on Learning Representations (ICLR)*, 1–14 (ICLR, 2016).

[44] Wang, Z. *et al. Dueling network architectures for deep reinforcement learning.* in *Proceedings of the 33rd International Conference on Machine Learning (ICML)*,

1995–2003 (PMLR, 2016).

[45] Zhang, S., Li, Y. & Dong, Q. Autonomous navigation of UAV in multi-obstacle environments based on a deep reinforcement learning approach. *Applied Soft Computing* **115**, 108194 (2022).

[46] Zhang, H. *et al. Robust deep reinforcement learning against adversarial perturbations on state observations.* in *Advances in Neural Information Processing Systems (NeurIPS)*, 21024–21037 (Curran Associates, 2020).

[47] Levy, A., Konidaris, G., Platt, R. & Saenko, K. *Learning multi-level hierarchies with hindsight.* in *Proceedings of the International Conference on Learning Representations (ICLR)*, 1–15 (ICLR, 2019).

[48] Nachum, O., Gu, S., Lee, H. & Levine, S. *Data-efficient hierarchical reinforcement learning.* in *Advances in Neural Information Processing Systems (NeurIPS)*, 3303–3313 (Curran Associates, 2018).

[49] Pateria, S., Subagdja, B., Tan, A.-H. & Quek, C. Hierarchical reinforcement learning: a comprehensive survey. *ACM Computing Surveys* **54**, 1–35 (2021).

[50] Jøsang, A., Ismail, R. & Boyd, C. A survey of trust and reputation systems for online service provision. *Decision Support Systems* **43**, 618–644 (2007).

[51] Kendall, A. & Gal, Y. *What uncertainties do we need in Bayesian deep learning for computer vision?* in *Advances in Neural Information Processing Systems (NeurIPS)*, 5574–5584 (Curran Associates, 2017).

[52] Lakshminarayanan, B., Pritzel, A. & Blundell, C. *Simple and scalable predictive uncertainty estimation using deep ensembles.* in *Advances in Neural Information Processing Systems (NeurIPS)*, 6402–6413 (Curran Associates, 2017).

[53] Murphy, K. P. *Probabilistic Machine Learning: An Introduction* (MIT Press, Cambridge, MA, 2022).

[54] Ghahramani, Z. Probabilistic machine learning and artificial intelligence. *Nature* **521**, 452–459 (2015).

[55] Finn, C., Abbeel, P. & Levine, S. *Model-agnostic meta-learning for fast adaptation of deep networks.* in *Proceedings of the 34th International Conference on Machine Learning (ICML)*, 1126–1135 (PMLR, 2017).

[56] Hospedales, T., Antoniou, A., Micaelli, P. & Storkey, A. Meta-learning in neural networks: a survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **44**, 5149–5169 (2022).

[57] Balta, E. C., Pease, M., Moyne, J., Barton, K. & Tilbury, D. M. Digital twin-based cyber-attack detection framework for cyber-physical manufacturing systems. *IEEE Transactions on Automation Science and Engineering* **21**, 1695–1712 (2024).

[58] Xu, Q., Ali, S. & Yue, T. *Digital twin-based anomaly detection in cyber-physical systems.* in *Proceedings of the IEEE 14th International Conference on Software Testing, Verification and Validation (ICST)*, 205–216 (IEEE, 2021).

[59] Konda, V. R. & Tsitsiklis, J. N. *Actor-critic algorithms.* in *Advances in Neural Information Processing Systems (NeurIPS)*, 1008–1014 (Curran Associates, 1999).

[60] van der Heijden, R. W., Lukaseder, T. & Kargl, F. *VeReMi: a dataset for comparable evaluation of misbehavior detection in VANETs.* in *Proceedings of the International Conference on Security and Privacy in Communication Networks (SecureComm)*, 318–337 (Springer, 2018).

[61] Dosovitskiy, A., Ros, G., Codevilla, F., López, A. & Koltun, V. *CARLA: an open urban driving simulator.* in *Proceedings of the 1st Annual Conference on Robot Learning (CoRL)*, 1–16 (PMLR, 2017).

[62] Kingma, D. P. & Ba, J. *Adam: a method for stochastic optimization.* in *Proceedings of the International Conference on Learning Representations (ICLR)*, 1–15 (ICLR, 2015).