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.

A Unified Two-Phase Motion Regime-Aware Hybrid Multi-Sensor Data Fusion Framework for Navigation of Intelligent Systems

Gnana Sri Sindhu Boppana
California State University, Fresno
Shuo Wu
shuowu@mail.fresnostate.edu

California State University, Fresno

Research Article

Keywords: Multi-sensor data fusion, Reinforcement learning-adaptive Kalman filtering, Neuromorphic motion classification, GPS spoofing defense

Posted Date: July 30th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-10464142/v1

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

Additional Declarations: No competing interests reported.

A Unified Two-Phase Motion Regime-Aware Hybrid Multi-Sensor Data Fusion Framework for Navigation of Intelligent Systems

Gnana Sri Sindhu Boppana[†] and Shuo Wu*[†]

Department of Electrical and Computer Engineering, California State University, Fresno, 93740, CA, USA.

*Corresponding author(s). E-mail(s): shuowu@mail.fresnostate.edu;
Contributing authors: srisindhu2409@mail.fresnostate.edu;
[†]These authors contributed equally to this work.

Abstract

Autonomous vehicle navigation relies on multi-sensor fusion combining IMU, GPS, and radar for continuous state estimation. Two fundamental limitations persist in existing approaches: Kalman filter noise parameters are fixed at design time and cannot adapt to changing driving conditions, and motion classification operates independently from state estimation with no information shared between them. This paper proposes a regime-aware hybrid multi-sensor fusion framework that addresses both limitations through a unified two-phase pipeline tested on the nuScenes v1.0-mini real-world driving dataset. In Phase 1, statistical features extracted from IMU, GPS, and radar are ranked via Gradient Boosting importance scoring, binarized, and classified into three driving regimes- stopped, cruising, and turning using a discrete Hopfield neural network with Hebbian learning, alongside Gradient Boosting, Random Forest, and a Neuromorphic Fusion classifier. In Phase 2, the classified regime label is passed as an eleventh dimension of the reinforcement learning agent state vector, making the agent regime-aware as it adaptively tunes the noise parameters of three Kalman filter variants- EKF, UKF, and square-root CKF using TD3, SAC, PPO, and an Ensemble RL agent. A GPS spoofing defense integrating Normalized Innovation Squared monitoring, CUSUM change detection, and a parallel trusted estimator with state rollback is incorporated within the same pipeline. Performance is evaluated across multiple configurations and independent random seeds using position RMSE, with statistical significance verified through paired Wilcoxon signed-rank tests.

Keywords: Multi-sensor data fusion, Reinforcement learning-adaptive Kalman filtering, Neuromorphic motion classification, GPS spoofing defense

1 Introduction

In recent years, an increasing number of artificial intelligence methods have been developed for sensing, automation, and control [1]. Reliable navigation in autonomous vehicles depends on accurate continuous estimation of position and velocity from onboard sensors. IMU, GPS, and radar are the three most commonly deployed sensing modalities for this purpose, each measuring a different physical aspect of vehicle motion [2][3]. While combining them through multi-sensor fusion improves robustness over any single sensor [4][5], two fundamental problems remain unsolved in current approaches.

The first is that Kalman filters, which form the backbone of sensor fusion pipelines, rely on fixed process and measurement noise covariances $\mathbf{Q}$ and $\mathbf{R}$ set before deployment [6]. These parameters do not adapt when driving conditions shift a stopped vehicle, a vehicle cruising at highway speed, and a vehicle turning at an intersection have fundamentally different motion dynamics and sensor noise profiles. This mismatch between fixed filter parameters and time-varying real-world conditions is a well-known source of estimation error [7][8]. The second problem is that vehicle motion classification and state estimation are treated as entirely separate tasks in existing systems. Even when a classifier correctly identifies the current driving regime, this information is discarded and never used to guide filter behavior [9][7].

GPS spoofing adds a third layer of difficulty. Unlike jamming, spoofing injects counterfeit satellite signals that the receiver accepts as genuine, causing the vehicle to track a wrong position trajectory with no visible indication of failure [18]. Detecting and recovering from such attacks within the same fusion pipeline, rather than as a standalone module, remains an open problem on real driving data.

This paper proposes a regime-aware hybrid multi-sensor fusion framework that addresses all three problems in a single pipeline. A Hopfield neural network [10] classifies the current driving regime from fused IMU, GPS, and radar features using the neuromorphic approach of Yu et al. [9], extended here to real autonomous driving data with three motion classes. This classification is then passed directly as an additional input dimension to a reinforcement learning agent that adaptively tunes the noise parameters of three Kalman filter variants at every timestep, building on the RL-adaptive filtering concept of Ma et al. [7] and extending it with multiple filter variants, multiple RL algorithms, and a GPS spoofing defense. The framework is evaluated on the nuScenes v1.0-mini real-world driving dataset [3].

The main contributions of this work are:

1. The motion class produced by Phase 1 is passed as the eleventh state dimension of the Phase 2 RL agent, making adaptive Kalman filter tuning explicitly regime-aware, a connection not previously demonstrated in the literature.

2. The Hopfield classification approach of [9] is extended to real vehicle motion data with three classes, evaluated alongside Gradient Boosting [11], Random Forest [12], and a Neuromorphic Fusion classifier on seven sensor combinations.
3. Three Kalman filter variants (EKF, UKF, SR-CKF [8]) are evaluated with TD3 [13], SAC [14], PPO [15], and an Ensemble RL agent [16][17] yielding a 21-method comparison on real driving data across five random seeds with Wilcoxon significance testing.
4. A GPS spoofing defense combining NIS monitoring, CUSUM change detection, and a parallel trusted estimator with rollback is integrated into the pipeline, with a Spoof-Aware TD3 variant trained under adversarial conditions [18].

2 Related Work

2.1 Multi-Sensor Data Fusion

Multi-sensor data fusion combines information from multiple sensing modalities to produce estimates more accurate and reliable than any single sensor alone [4]. Hall and Llinas [4] established the foundational taxonomy of fusion levels: data-level fusion combines raw sensor readings directly, feature-level fusion extracts and combines statistical descriptors from each sensor independently, and decision-level fusion aggregates independent per-sensor classifications. Lu and Dai [5] provide a broader survey of fusion architectures across robotics and autonomous systems applications. In autonomous driving, feature-level and state-level fusion are most commonly employed, the former for classification tasks and the latter for continuous state estimation through Kalman filtering [4][5]. This work uses both: feature-level fusion in Phase 1 for motion classification and state-level fusion in Phase 2 for position and velocity estimation.

2.2 Neuromorphic Computing for Sensor Fusion

Yu et al. [9] proposed a multi-sensor fusion framework for human activity recognition (HAR) using IMU, USRP software-defined radio, and radar signals. Their pipeline applies a TopK attention mechanism to select the $K = 25$ most informative features from a fused feature matrix, reshapes them into a 5×5 binary pattern, and feeds this into a discrete Hopfield neural network [10] for one-shot associative classification. The Hopfield network weight matrix is computed via the Hebbian learning rule:

$$\mathbf{W} = \frac{1}{P} \sum_{p=1}^{P} \boldsymbol{\xi}^{(p)} \boldsymbol{\xi}^{(p)\top}, \quad w_{ii} = 0 \tag{1}$$

where P is the number of stored patterns and $\boldsymbol{\xi}^{(p)} \in \{-1, +1\}^N$ is the p-th class prototype. Classification is performed by iterative energy minimization to the nearest stored attractor, with the predicted class determined by Euclidean similarity to each stored pattern. Their system achieved 98.98% accuracy on two activity classes from controlled laboratory data. Two limitations are relevant to this work: the system was tested only in a lab setting with simple sitting and standing activities, and the classification output was not connected to any downstream control or estimation task.

This paper extends their approach to real autonomous driving data with three motion classes and directly feeds the classification result into the state estimation pipeline.

2.3 Adaptive Kalman Filtering with Reinforcement Learning

Ma et al. [7] combined a Kalman filter with a SAC reinforcement learning agent for GPS/INS navigation in UAV systems. Their work demonstrated that an RL agent can learn to tune filter noise parameters adaptively in the presence of GPS spoofing, using the Normalized Innovation Squared (NIS) statistic to monitor filter consistency:

$$\gamma_{\text{NIS}} = \boldsymbol{\Delta}_k^\top \boldsymbol{\Sigma}_{\Delta_k}^{-1} \boldsymbol{\Delta}_k \tag{2}$$

where $\boldsymbol{\Delta}_k = \mathbf{z}_k - \mathbf{H}\hat{\mathbf{x}}_{k|k-1}$ is the innovation vector and $\boldsymbol{\Sigma}_{\Delta_k} = \mathbf{H}\mathbf{P}_{k|k-1}\mathbf{H}^\top + \mathbf{R}_k$ is the innovation covariance. When γ_{NIS} exceeds the chi-square threshold $\chi^2_{\alpha,\nu}$, the filter flags an inconsistency. Their system used a single EKF variant with SAC on simulated UAV trajectories. This paper extends this concept to three filter variants, three RL algorithms, and an Ensemble agent on real driving data, and adds regime-aware state augmentation not present in [7]. Zhu et al. [2] proposed a decoupled deep RL framework for autonomous driving combining sensor fusion with imitation learning, showing that separating perception and control modules improves training stability. Their work focused on high-level driving decisions using camera and LiDAR rather than low-level noise parameter tuning from IMU, GPS, and radar, which is the focus of this paper.

2.4 Nonlinear Kalman Filter Variants

Three nonlinear Kalman filter variants are used in this work. The EKF linearizes the nonlinear state transition using a first-order Jacobian approximation at each timestep [6]. The UKF avoids explicit linearization by propagating $2n + 1$ deterministically chosen sigma points through the nonlinear dynamics [6]:

$$\hat{\mathbf{x}}_{k|k-1} = \sum_{i=0}^{2n} W_i^m f(\boldsymbol{\chi}_i) \tag{3}$$

where $\boldsymbol{\chi}_i$ are the sigma points and W_i^m are their associated mean weights. The SR-CKF, introduced by Arasaratnam and Haykin [8], propagates the Cholesky factor $\mathbf{S}$ of the error covariance directly rather than the full covariance matrix $\mathbf{P}$, guaranteeing positive definiteness throughout:

$$\mathbf{S}_{k|k-1} = \text{qr}\left\{ \frac{1}{\sqrt{2n}} [\mathcal{X}_1 - \hat{\mathbf{x}}_{k|k-1}, \ldots, \mathcal{X}_{2n} - \hat{\mathbf{x}}_{k|k-1}],\ \sqrt{\mathbf{Q}_k} \right\} \tag{4}$$

where $\mathcal{X}_i = \mathbf{S}_{k-1}\boldsymbol{\xi}_i + \hat{\mathbf{x}}_{k-1}$ are the cubature points and $\text{qr}\{\cdot\}$ denotes the QR decomposition. This numerical stability advantage makes SR-CKF the preferred third filter variant in this work over a standard CKF implementation.

2.5 GPS Spoofing in Navigation Systems

GPS spoofing poses a serious threat to autonomous navigation systems. Unlike jamming, which causes an obvious signal loss, spoofing injects counterfeit satellite signals

that are accepted as genuine by the receiver, silently corrupting the position estimate [18]. Alhoraibi et al. [18] studied GPS spoofing attacks on UAVs and proposed a machine learning-based detection framework, demonstrating that effective detection requires monitoring the temporal consistency of navigation estimates rather than inspecting the GPS signal directly. Ma et al. [7] showed that NIS-based monitoring within a Kalman filter framework provides a practical consistency check for spoofing detection. The spoofing attack model used in this paper simulates a smooth ramping offset rather than a sudden jump, making detection harder and closer to realistic attack scenarios. Defense integrates NIS monitoring, CUSUM change detection, and a parallel trusted estimator using only IMU and radar with GPS rollback on alarm, a more complete defense architecture than considered in [7] or [18].

3 System Architecture

3.1 Overview

The proposed framework processes raw IMU, GPS, and radar sensor streams from the nuScenes v1.0-mini autonomous driving dataset [3] through a two-phase pipeline, illustrated in Fig. 1. Phase 1 performs neuromorphic motion classification using feature-level sensor fusion, producing a motion class label at every timestep. Phase 2 performs adaptive state estimation using reinforcement learning-tuned Kalman filtering at the state level. The critical connection between the two phases is that the Phase 1 motion class is passed directly as the eleventh dimension of the Phase 2 RL agent state vector, making the filter noise tuning policy explicitly aware of the current driving regime. This regime-aware integration is the central design decision of this work and distinguishes it from both reference frameworks [9][7].

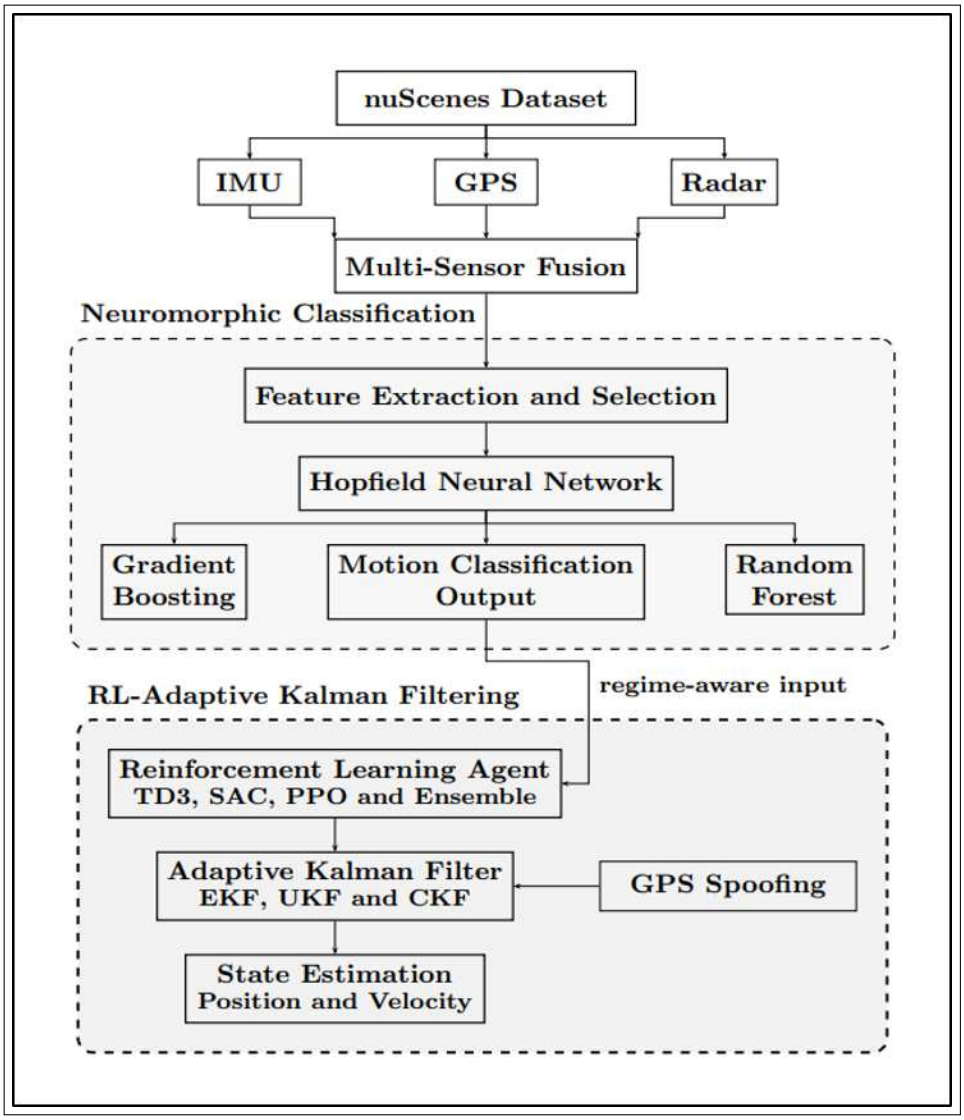


Fig. 1 Proposed Framework

3.2 Dataset and Sensor Configuration

The nuScenes dataset [3] was collected from real on-road driving in Boston and Singapore, covering diverse traffic conditions, road layouts, and times of day. The full dataset contains 1000 driving scenes each 20 seconds long. This work uses the v1.0-mini subset, which contains 10 scenes with 404 annotated samples. The nuScenes vehicle is equipped with a full sensor suite including six cameras, one LiDAR, five radar units, GPS, and IMU, all hardware-synchronized [3]. This work uses only IMU, GPS, and radar.

Table 1 Sensor signals extracted from the nuScenes v1.0-mini dataset [3].

Sensor	Signal	Quantity	Unit
IMU	a_x, a_y, a_z	Linear acceleration	m/s^2
	$\omega_x, \omega_y, \omega_z$	Angular velocity	rad/s
GPS	p_x, p_y	East/North position	m
	v	Ground speed	m/s
Radar	$v_{x,c}, v_{y,c}$	Compensated radial velocity	m/s
	v_r	Radar speed magnitude	m/s
	d	Distance to nearest detection	m

Table 2 Fusion levels and their application in the proposed pipeline.

Fusion Level	Operation	Used In
Feature-level	Extract and concatenate statistical features	Phase 1 classification
State-level	Fuse measurements inside Kalman filter update	Phase 2 state estimation

Table 1 summarizes the signals extracted from each sensor, the physical quantity measured, and the role each plays in the pipeline.

Raw nuScenes data is stored in a structured JSON schema linking scenes, samples, sensor sweeps, and ego poses through token references. For each scene, IMU signals are read from the `imu_data` table, GPS position is derived from ego pose translations projected into a local flat coordinate frame in meters, and radar measurements are aggregated across all five radar channels with the nearest detection selected as the representative measurement at each timestep. All signals are aligned into a single time-indexed array per scene with each row corresponding to one timestep.

3.3 Fusion Architecture

Two fusion levels are employed in this pipeline, following the taxonomy of Hall and Llinas [4]. Feature-level fusion is used in Phase 1, where statistical features are extracted from IMU, GPS, and radar independently and concatenated into a single feature vector for motion classification [9]. State-level fusion is used in Phase 2, where GPS position measurements and radar velocity measurements are fused inside the Kalman filter update step at every timestep, with IMU acceleration driving the prediction step [7][6]. Table 2 summarizes how each fusion level is applied.

Seven sensor combination configurations are evaluated in Phase 1 to quantify the contribution of each sensor to classification accuracy, as listed in Table 3.

Table 3 Sensor combination configurations evaluated in Phase 1.

Config	IMU	GPS	Radar
1	✓		
2		✓	
3			✓
4	✓	✓	
5	✓		✓
6		✓	✓
7	✓	✓	✓

4 Phase 1: Neuromorphic Motion Classification

4.1 Feature Extraction and Sliding Window Analysis

Raw IMU, GPS, and radar signals are continuous time-series streams. A sliding window of fixed size $W = 5$ timesteps with step size $S = 1$ is applied over each scene to capture short-term vehicle behavior. At each window position, five statistical descriptors are computed for every sensor signal channel — mean, standard deviation, minimum, maximum, and range R_s:

$$\mu_s = \frac{1}{W}\sum_{i=1}^{W} s_i \tag{5}$$

$$\sigma_s = \sqrt{\frac{1}{W}\sum_{i=1}^{W}(s_i - \mu_s)^2} \tag{6}$$

$$R_s = \max(s_1, \ldots, s_W) - \min(s_1, \ldots, s_W) \tag{7}$$

where s_i is the sensor reading at timestep i inside the window. These statistics are computed separately for each sensor type, preserving the physical characteristics of each modality before fusion [9]. The total number of windows extracted from a scene of length N is:

$$N_{\text{windows}} = N - W + 1 \tag{8}$$

Table 4 summarizes the feature breakdown across the three sensor types, yielding 66 features per window in total.

Table 4 Feature extraction summary by sensor type.

Sensor	Features	Feature Types
IMU	34	Acceleration, angular velocity, magnitude statistics
GPS	12	Position, velocity, speed, displacement, heading, tortuosity
Radar	14	Range, velocity, RCS, point count, density
Total	**66**	All sensor statistics combined

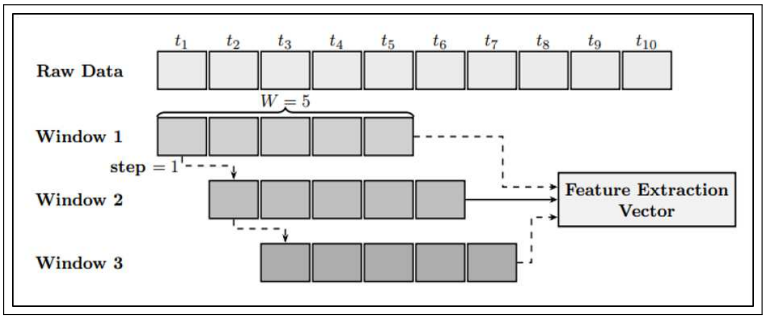


Fig. 2 Sliding window feature extraction.

4.2 Motion Class Labeling

Each window is assigned a ground truth motion class based on vehicle speed and heading change computed from GPS velocity components. Speed is computed as:

$$v = \sqrt{v_x^2 + v_y^2} \tag{9}$$

Heading change across the window is:

$$\Delta\psi = \psi_{\text{end}} - \psi_{\text{start}}, \quad \psi = \arctan 2(v_y, v_x) \tag{10}$$

The labeling rule is:

$$\text{Class} = \begin{cases} \text{Stopped} & \text{if } v < 0.5 \text{ m/s} \\ \text{Turning} & \text{if } v \geq 0.5 \text{ m/s and } |\Delta\psi| > \theta_{th} \\ \text{Cruising} & \text{otherwise} \end{cases} \tag{11}$$

where θ_{th} is the heading change threshold. This produces three motion classes that reflect distinct vehicle dynamics and noise characteristics relevant to filter tuning in Phase 2.

4.3 Feature Selection and Binarization

Not all 66 extracted features contribute equally to classification. A Gradient Boosting classifier is first trained on all features and the importance score of each feature f_j is computed as:

$$I(f_j) = \sum_{t=1}^{T} \sum_{n \in \mathcal{N}_t} \mathbb{1}[f_n = f_j] \cdot \Delta\text{impurity}(n) \tag{12}$$

where T is the total number of trees, $\mathcal{N}_t$ is the set of internal nodes in tree t, and $\Delta\text{impurity}(n)$ is the Gini impurity reduction at node n [11]. The top $K = 25$ features are selected, following the TopK attention mechanism of [9].

The selected features are then binarized using per-feature median thresholds computed from the training set:

$$b_i = \begin{cases} +1 & \text{if } f_i \geq t_i \\ -1 & \text{otherwise} \end{cases} \tag{13}$$

where t_i is the median value of feature i across all training windows. The resulting binary vector of 25 elements is reshaped into a 5×5 binary pattern matrix $\mathbf{B} \in \{-1, +1\}^{5\times5}$ for input to the Hopfield network [9].

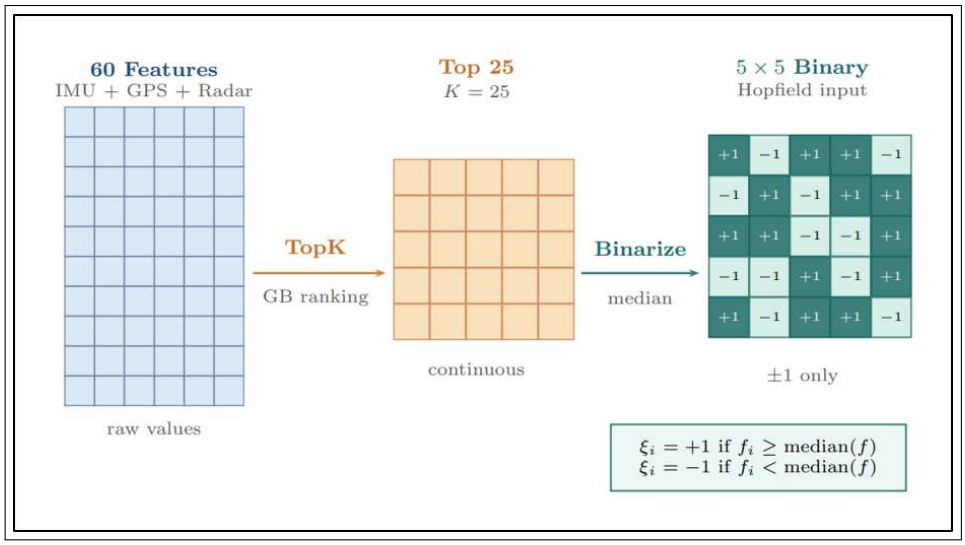


Fig. 3 Feature selection and binarization.

4.4 Hopfield Neural Network Classifier

The discrete Hopfield neural network [10] stores one binary prototype pattern per motion class. The weight matrix is computed from the stored patterns via the Hebbian learning rule [9]:

$$\mathbf{W} = \frac{1}{P} \sum_{p=1}^{P} \boldsymbol{\xi}^{(p)} \boldsymbol{\xi}^{(p)\top}, \quad w_{ii} = 0 \tag{14}$$

where $P = 3$ is the number of stored patterns and $\boldsymbol{\xi}^{(p)} \in \{-1, +1\}^{25}$ is the flattened prototype for class p. Given a new input binary vector $\mathbf{u}$, the network iteratively updates each neuron until convergence:

$$u_i^{(t+1)} = \operatorname{sgn}\left(\sum_{j \neq i} w_{ij} u_j^{(t)} \right) \tag{15}$$

The recalled pattern $\hat{\mathbf{u}}$ is compared to each stored prototype using Euclidean similarity:

$$\text{sim}^{(p)} = \frac{1}{1 + \|\hat{\mathbf{u}} - \boldsymbol{\xi}^{(p)}\|_2} \tag{16}$$

The class with the highest similarity score is the predicted motion class. The three similarity scores are also retained as confidence features for the Neuromorphic Fusion classifier described in Section 4.6.

4.5 Gradient Boosting and Random Forest Classifiers

Gradient Boosting [11] builds an ensemble of decision trees sequentially, where each new tree h_m is fitted to the negative gradient of the loss with respect to the current ensemble prediction:

$$h_m = \arg\min_{h} \sum_{i=1}^{N} \left(-\frac{\partial L(y_i, F_{m-1}(x_i))}{\partial F_{m-1}(x_i)} - h(x_i) \right)^2 \tag{17}$$

$$F_m(x) = F_{m-1}(x) + \eta \cdot h_m(x) \tag{18}$$

where η is the learning rate. Random Forest [12] builds T decision trees independently on bootstrap samples of the training data, with each node split considering only a random feature subset. The final prediction is determined by majority vote:

$$\hat{y} = \arg\max_{c} \sum_{t=1}^{T} \mathbb{1}[h_t(x) = c] \tag{19}$$

Both classifiers are trained on the 25 selected features and evaluated across all seven sensor combinations in Table 3.

4.6 Neuromorphic Fusion

Neuromorphic Fusion augments the 25 selected features with the three Hopfield similarity scores to form a 28-dimensional input vector:

$$\mathbf{f}_{\text{NF}} = \left[f_1, \ldots, f_{25},\ \text{sim}^{(1)},\ \text{sim}^{(2)},\ \text{sim}^{(3)} \right] \tag{20}$$

This extended vector is passed to a Gradient Boosting classifier for the final motion class prediction. The Hopfield similarity scores serve as soft neuromorphic evidence that complements the raw statistical features, allowing the classifier to leverage both data-driven feature importance and associative memory recall in a single decision.

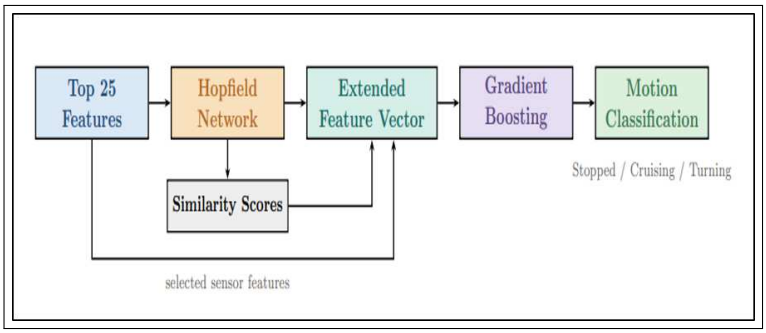


Fig. 4 Neuromorphic Fusion pipeline.

The complete Phase 1 pipeline is summarized in Algorithm 1.

Algorithm 1 Phase 1: Neuromorphic Classification Pipeline

Require: Raw IMU, GPS, radar streams; window size $W = 5$, step $S = 1$, $K = 25$
Ensure: Motion class $\hat{c} \in \{\text{Stopped, Cruising, Turning}\}$ per window
  for each sliding window $\mathbf{x}_{t:t+W}$ **do**
    Compute 66 statistical features from all 13 signals
    Assign label c_t using Eq. (11)
  end for
  Rank features by Gradient Boosting importance (Eq. (12))
  Select top $K = 25$ features $\mathbf{f}^*$
  Binarize each feature at training median (Eq. (13)) $\rightarrow \mathbf{b}_t \in \{-1, +1\}^{25}$, reshape to 5×5 pattern $\mathbf{P}_t$
  Compute Hopfield weights $\mathbf{W}$ via Eq. (14)
  for each test window **do**
    Recall: $\hat{\mathbf{u}} \leftarrow$ Eq. (15) until convergence
    Compute similarity scores $\text{sim}^{(p)}$ via Eq. (16)
    Form $\mathbf{f}_{\text{NF}}$ via Eq. (20)
    Classify with Gradient Boosting and Random Forest
    $\hat{c}_t \leftarrow$ majority vote across all classifiers
  end for
  return $\hat{c}_t$ for each window t

5 Phase 2: RL-Adaptive Kalman Filtering

5.1 State Space Model

The vehicle state vector contains six elements representing 3D position and 3D velocity:

$$\mathbf{x} = [x,\ y,\ z,\ v_x,\ v_y,\ v_z]^\top \tag{21}$$

The discrete-time state transition equation is driven by IMU acceleration measurements:

$$\mathbf{x}_k = \mathbf{F}\mathbf{x}_{k-1} + \mathbf{B}\mathbf{u}_k + \mathbf{w}_k \tag{22}$$

where $\mathbf{u}_k = [a_x, a_y, a_z]^\top$ is the IMU acceleration input, $\mathbf{w}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{Q})$ is process noise, and the state transition and control input matrices for sampling interval Δt are:

$$\mathbf{F} = \begin{bmatrix} \mathbf{I}_3 & \Delta t \cdot \mathbf{I}_3 \\ \mathbf{0}_3 & \mathbf{I}_3 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} \frac{1}{2}\Delta t^2 \cdot \mathbf{I}_3 \\ \Delta t \cdot \mathbf{I}_3 \end{bmatrix} \tag{23}$$

GPS position and radar velocity measurements are modeled as:

$$\mathbf{z}_k^{\text{GPS}} = \mathbf{H}_{\text{GPS}}\mathbf{x}_k + \mathbf{n}_k, \quad \mathbf{H}_{\text{GPS}} = [\mathbf{I}_3 \mid \mathbf{0}_3] \tag{24}$$

$$\mathbf{z}_k^{\text{Radar}} = \mathbf{H}_{\text{Radar}}\mathbf{x}_k + \mathbf{n}_k, \quad \mathbf{H}_{\text{Radar}} = [\mathbf{0}_3 \mid \mathbf{I}_3] \tag{25}$$

where $\mathbf{n}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{R})$ is measurement noise. The noise covariances $\mathbf{Q}$ and $\mathbf{R}$ are adaptively scaled by the RL agent at every timestep as described in Section 5.3.

5.2 Kalman Filter Variants

5.2.1 Extended Kalman Filter

The EKF handles mild nonlinearities by linearizing the system around the current state estimate using a Jacobian $\mathbf{F}$ at each timestep [6]. The predict and update steps are:

$$\hat{\mathbf{x}}_{k|k-1} = \mathbf{F}\hat{\mathbf{x}}_{k-1|k-1} + \mathbf{B}\mathbf{u}_k \tag{26}$$

$$\mathbf{P}_{k|k-1} = \mathbf{F}\mathbf{P}_{k-1|k-1}\mathbf{F}^\top + \mathbf{Q}_k \tag{27}$$

$$\mathbf{K}_k = \mathbf{P}_{k|k-1}\mathbf{H}^\top \left(\mathbf{H}\mathbf{P}_{k|k-1}\mathbf{H}^\top + \mathbf{R}_k\right)^{-1} \tag{28}$$

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k\left(\mathbf{z}_k - \mathbf{H}\hat{\mathbf{x}}_{k|k-1}\right) \tag{29}$$

$$\mathbf{P}_{k|k} = \left(\mathbf{I} - \mathbf{K}_k\mathbf{H}\right)\mathbf{P}_{k|k-1}\left(\mathbf{I} - \mathbf{K}_k\mathbf{H}\right)^\top + \mathbf{K}_k\mathbf{R}_k\mathbf{K}_k^\top \tag{30}$$

The NIS is computed at each update step to monitor filter consistency and detect GPS spoofing [7]:

$$\gamma_{\text{NIS}} = \boldsymbol{\Delta}_k^\top \boldsymbol{\Sigma}_{\Delta_k}^{-1} \boldsymbol{\Delta}_k \tag{31}$$

where $\boldsymbol{\Delta}_k = \mathbf{z}_k - \mathbf{H}\hat{\mathbf{x}}_{k|k-1}$ is the innovation vector and $\boldsymbol{\Sigma}_{\Delta_k} = \mathbf{H}\mathbf{P}_{k|k-1}\mathbf{H}^\top + \mathbf{R}_k$. If $\gamma_{\text{NIS}} > \chi^2_{\alpha,\nu} = 7.815$ (chi-square threshold for $\nu = 3$ degrees of freedom at $\alpha = 0.05$), the filter flags an inconsistency [7].

5.2.2 Unscented Kalman Filter

The UKF propagates $2n + 1$ sigma points directly through the nonlinear dynamics, avoiding Jacobian computation [6]. For state dimension $n = 6$, sigma points are generated as:

$$\boldsymbol{\chi}_0 = \hat{\mathbf{x}}_{k-1} \tag{32}$$

$$\boldsymbol{\chi}_i = \hat{\mathbf{x}}_{k-1} + \left(\sqrt{(n+\lambda)\mathbf{P}_{k-1}}\right)_i, \quad i = 1, \ldots, n \tag{33}$$

$$\boldsymbol{\chi}_{i+n} = \hat{\mathbf{x}}_{k-1} - \left(\sqrt{(n+\lambda)\mathbf{P}_{k-1}}\right)_i, \quad i = 1, \ldots, n \tag{34}$$

where $\lambda = \alpha^2(n + \kappa) - n$ is a composite scaling parameter. The predicted mean and covariance are weighted sums of the propagated sigma points:

$$\hat{\mathbf{x}}_{k|k-1} = \sum_{i=0}^{2n} W_i^m f(\boldsymbol{\chi}_i) \tag{35}$$

$$\mathbf{P}_{k|k-1} = \sum_{i=0}^{2n} W_i^c \left[f(\boldsymbol{\chi}_i) - \hat{\mathbf{x}}_{k|k-1}\right]\left[f(\boldsymbol{\chi}_i) - \hat{\mathbf{x}}_{k|k-1}\right]^\top + \mathbf{Q}_k \tag{36}$$

5.2.3 Square-Root Cubature Kalman Filter

The SR-CKF [8] generates $2n = 12$ cubature points from the Cholesky factor $\mathbf{S}_{k-1}$ of the error covariance:

$$\mathcal{X}_i = \mathbf{S}_{k-1}\boldsymbol{\xi}_i + \hat{\mathbf{x}}_{k-1}, \quad i = 1, \ldots, 2n \tag{37}$$

where $\boldsymbol{\xi}_i = \sqrt{n}\mathbf{e}_i$ and $\boldsymbol{\xi}_{i+n} = -\sqrt{n}\mathbf{e}_i$ are the standard cubature points and $\mathbf{e}_i$ are standard basis vectors. The square-root factor is propagated directly via QR decomposition:

$$\mathbf{S}_{k|k-1} = \text{qr}\left\{\frac{1}{\sqrt{2n}}\left[\mathcal{X}_1 - \hat{\mathbf{x}}_{k|k-1}, \ldots, \mathcal{X}_{2n} - \hat{\mathbf{x}}_{k|k-1}\right],\ \sqrt{\mathbf{Q}_k}\right\} \tag{38}$$

Propagating $\mathbf{S}$ rather than $\mathbf{P}$ guarantees positive definiteness throughout all filter steps and prevents numerical divergence that can occur in standard CKF implementations [8].

5.3 Reinforcement Learning Environment

5.3.1 State, Action, and Reward

The RL agent observes an 11-dimensional state vector at every timestep [7]:

$$\mathbf{s}_t = \left[e_x,\ e_y,\ e_z,\ e_{v_x},\ e_{v_y},\ e_{v_z},\ \gamma_{\text{NIS}},\ a_x,\ a_y,\ r,\ \hat{c}\right]^\top \tag{39}$$

where e_x, e_y, e_z are position errors, $e_{v_x}, e_{v_y}, e_{v_z}$ are velocity errors, γ_{NIS} is the current NIS value, a_x, a_y are IMU lateral accelerations, r is the radar range, and $\hat{c} \in \{0, 1, 2\}$ is the Phase 1 motion class label. The eleventh dimension $\hat{c}$ is the regime-aware augmentation that distinguishes this framework from [9] and [7].

The action space is three-dimensional, producing noise scaling factors for $\mathbf{Q}$ and $\mathbf{R}$:

$$\mathbf{a}_t = [\alpha_Q,\ \alpha_{\text{GPS}},\ \alpha_{\text{Radar}}] \in [0.1,\ 10.0]^3 \tag{40}$$

$$\mathbf{Q}_k = \alpha_Q \cdot \mathbf{Q}_{\text{base}}, \quad \mathbf{R}_k^{\text{GPS}} = \alpha_{\text{GPS}} \cdot \mathbf{R}_{\text{base}}^{\text{GPS}}, \quad \mathbf{R}_k^{\text{Radar}} = \alpha_{\text{Radar}} \cdot \mathbf{R}_{\text{base}}^{\text{Radar}} \tag{41}$$

The reward function penalizes position error, velocity error, and filter inconsistency:

$$r_t = -w_1 \cdot \text{RMSE}_{\text{pos}}(t) - w_2 \cdot \text{RMSE}_{\text{vel}}(t) - w_3 \cdot \max\left(0,\ \gamma_{\text{NIS}}(t) - 7.815\right) \tag{42}$$

The third term activates only when the filter becomes inconsistent, penalizing the agent for allowing GPS measurements to corrupt the state estimate and implicitly encouraging spoofing detection [7].

5.4 RL Algorithms

5.4.1 TD3: Twin Delayed Deep Deterministic Policy Gradient

TD3 [13] addresses Q-value overestimation in actor-critic methods through three mechanisms: twin critic networks, delayed policy updates, and target policy smoothing. The twin critic target value is:

$$y = r + \gamma \min_{i=1,2} Q_{\theta_i}(s', \tilde{a}), \quad \tilde{a} = \pi_\phi(s') + \epsilon, \quad \epsilon \sim \text{clip}(\mathcal{N}(0, \sigma), -c, c) \tag{43}$$

Target networks are updated via soft updates:

$$\theta_i' \leftarrow \tau\theta_i + (1 - \tau)\theta_i', \quad i = 1, 2 \tag{44}$$

where $\tau \ll 1$ ensures slow, stable target network evolution [13].

5.4.2 SAC: Soft Actor-Critic

SAC [14] maximizes both reward and policy entropy, encouraging exploration without manual noise tuning:

$$J(\pi) = \sum_{t=0}^{T} \mathbb{E}_{(s_t,a_t)\sim\rho_\pi} \left[r(s_t, a_t) + \alpha \mathcal{H}(\pi(\cdot|s_t)) \right] \tag{45}$$

where α is the temperature parameter and $\mathcal{H}(\pi(\cdot|s_t)) = -\mathbb{E}[\log \pi(a_t|s_t)]$ is the policy entropy. The temperature α is auto-tuned during training [14].

5.4.3 PPO: Proximal Policy Optimization

PPO [15] is an on-policy algorithm that clips the policy update ratio to prevent large destabilizing policy changes:

$$L^{\text{CLIP}}(\theta) = \mathbb{E}_t \left[\min \left(r_t(\theta)\hat{A}_t,\ \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t \right) \right] \tag{46}$$

where $r_t(\theta) = \pi_\theta(a_t|s_t)/\pi_{\theta_{\text{old}}}(a_t|s_t)$ is the probability ratio and $\hat{A}_t$ is the estimated advantage. PPO requires fresh experience at every update and cannot reuse past transitions from a replay buffer, making it less sample efficient than TD3 and SAC for continuous control tasks [15].

5.5 Ensemble RL Agent

The Ensemble RL agent combines TD3, SAC, and PPO by taking the element-wise median of their actions at every timestep [16][17]:

$$a_{\text{ensemble}}[i] = \text{median}\left(a_{\text{TD3}}[i],\ a_{\text{SAC}}[i],\ a_{\text{PPO}}[i] \right), \quad i = 1, 2, 3 \tag{47}$$

The median is chosen over the mean because it is robust to a single poorly performing agent — if PPO produces a poor action, the median is not pulled toward that value the way a mean would be [16].

5.6 Regime-Aware Integration

The connection between Phase 1 and Phase 2 is the central contribution of this work. After Phase 1 classifies the current window, the class label $\hat{c}_k \in \{0, 1, 2\}$ is passed as the eleventh state dimension to the RL agent at every timestep:

$$\hat{c}_k \leftarrow \text{Phase 1}(\mathbf{B}_k) \rightarrow s_k^{(11)} \rightarrow \pi_\theta(\mathbf{s}_k) \rightarrow [\alpha_Q, \alpha_{\text{GPS}}, \alpha_{\text{Radar}}] \rightarrow \text{Kalman Filter} \tag{48}$$

A stopped vehicle should have near-zero process noise since it is not moving. A turning vehicle should have higher process noise to account for heading uncertainty. Without regime awareness, the RL agent must infer this from position and velocity errors alone. With $\hat{c}$ available, the agent can learn noise-scaling policies specific to each motion regime [9][7].

The complete Phase 2 pipeline is summarized in Algorithm 2.

Algorithm 2 Phase 2: RL-Adaptive Kalman Filtering

Require: IMU, GPS, radar streams; trained RL agent π_θ; motion class $\hat{c}_t$ from Phase 1

Ensure: Estimated vehicle state $\hat{\mathbf{x}}_t$, position RMSE

Initialize: $\hat{\mathbf{x}}_0 \leftarrow$ first GPS reading; set base noise matrices $\mathbf{Q}_0$, $\mathbf{R}_0$

for each timestep t **do**

Build RL state $\mathbf{s}_t \in \mathbb{R}^{11}$ via Eq. (39) using $\hat{c}_t$ from Phase 1

Query agent: $\mathbf{a}_t = \pi_\theta(\mathbf{s}_t) \rightarrow [\alpha_Q, \alpha_{\text{GPS}}, \alpha_{\text{Radar}}]$

Scale noise via Eq. (41)

Predict: $\hat{\mathbf{x}}_{t|t-1}$, $\mathbf{P}_{t|t-1}$ using IMU input $\mathbf{u}_t$

Update: fuse GPS and radar measurements; compute γ_{NIS} via Eq. (31)

Compute reward r_t via Eq. (42)

Store transition; update π_θ via RL gradient step

end for

return $\hat{\mathbf{x}}_t$ for all t; $\text{RMSE}_{\text{pos}} = \sqrt{\frac{1}{T}\sum_t \|\mathbf{e}_{p,t}\|^2}$

6 GPS Spoofing Attack and Defense

6.1 Spoofing Attack Model

GPS spoofing injects counterfeit satellite signals that are accepted by the receiver as genuine, silently shifting the reported position without any signal quality warning [18]. Unlike jamming, which produces an obvious loss of signal, a well-crafted spoofing attack is indistinguishable from legitimate GPS reception at the signal level, making it one of the most dangerous threats to autonomous navigation systems [7][18].

A smooth ramping spoofing attack is simulated in this work. Rather than applying a sudden large offset, which would produce an immediately detectable NIS spike, the attack gradually increases the GPS position error over time, making detection harder and more representative of realistic attack scenarios [7]. The attack is active from 30% to 70% of each scene, covering the middle portion of the driving sequence. The spoofed GPS measurement at timestep k is modeled as:

$$\mathbf{z}_k^{\text{GPS,spoof}} = \mathbf{z}_k^{\text{GPS,true}} + \boldsymbol{\delta}(k) \tag{49}$$

where the spoofing offset grows linearly over the attack window:

$$\boldsymbol{\delta}(k) = \boldsymbol{\delta}_{\max} \cdot \frac{k - k_{\text{start}}}{k_{\text{end}} - k_{\text{start}}}, \quad k_{\text{start}} \leq k \leq k_{\text{end}} \tag{50}$$

with k_{start} and k_{end} set at 30% and 70% of the total scene length respectively, and $\boldsymbol{\delta}_{\max}$ the maximum offset applied at the end of the attack window. Outside the attack window:

$$\boldsymbol{\delta}(k) = \mathbf{0}, \quad k < k_{\text{start}} \text{ or } k > k_{\text{end}} \tag{51}$$

A small zero-mean Gaussian jitter $\boldsymbol{\eta} \sim \mathcal{N}(\mathbf{0}, \sigma_j^2 \mathbf{I})$ scaled by the current ramp factor is added to the offset to make the attack less deterministic and harder to detect through simple thresholding.

6.2 NIS-Based Detection

Filter consistency is monitored at every timestep using the Normalized Innovation Squared statistic [7]:

$$\gamma_{\text{NIS}}(k) = \boldsymbol{\Delta}_k^\top \boldsymbol{\Sigma}_{\Delta_k}^{-1} \boldsymbol{\Delta}_k \tag{52}$$

Under nominal GPS conditions, γ_{NIS} follows a chi-square distribution with $\nu = 3$ degrees of freedom corresponding to the three GPS position measurement dimensions. Spoofing detection is triggered when:

$$\gamma_{\text{NIS}}(k) > \chi^2_{\alpha,\nu} = 7.815 \tag{53}$$

which corresponds to a significance level of $\alpha = 0.05$. Any NIS value above this threshold indicates less than 5% probability that the measurement is consistent with the filter model under nominal conditions, flagging a likely attack [7].

CUSUM change detection is applied on top of NIS monitoring to improve sensitivity to gradual drift. The CUSUM statistic accumulates evidence of a shift in the NIS sequence:

$$C_k = \max\left(0,\ C_{k-1} + \gamma_{\text{NIS}}(k) - \mu_0 - \kappa\right) \tag{54}$$

where μ_0 is the expected NIS under nominal conditions, κ is the allowable slack, and an alarm is raised when $C_k > h$ for detection threshold h. CUSUM is particularly effective for the smooth ramping attack modeled here, where the NIS increase is gradual and single-threshold detection may miss the early stages of the attack.

6.3 Defense Mechanism and Trusted Estimator

When an attack is detected — either through NIS threshold exceedance or CUSUM alarm — the defense mechanism activates a parallel trusted estimator that runs exclusively on IMU and radar measurements, excluding GPS entirely:

$$\hat{\mathbf{x}}_k^{\text{trusted}} = f\left(\hat{\mathbf{x}}_{k-1}^{\text{trusted}},\ \mathbf{u}_k^{\text{IMU}},\ \mathbf{z}_k^{\text{Radar}}\right) \tag{55}$$

On alarm, the main filter state is rolled back to the trusted estimator output:

$$\hat{\mathbf{x}}_k \leftarrow \hat{\mathbf{x}}_k^{\text{trusted}} \quad \text{if } \gamma_{\text{NIS}}(k) > 7.815 \text{ or } C_k > h \tag{56}$$

The NIS inconsistency penalty in the RL reward function (Eq. (42)) also acts as an implicit defense signal. When GPS measurements are spoofed, the NIS rises and the agent is penalized, learning to reduce α_{GPS} — the GPS noise scaling action — effectively down-weighting GPS influence during an attack without requiring explicit detection logic.

6.4 Spoof-Aware TD3 Training

A dedicated Spoof-Aware TD3 (TD3-SA) variant is trained with GPS spoofing active during 70% of training episodes. During these episodes, a SpoofAttack instance with randomized onset and offset parameters is injected into the training environment. This forces the agent to experience spoofed GPS measurements throughout training

and learn to handle them naturally. At evaluation time, the TD3-SA agent shows no degradation in position RMSE under spoofing compared to clean conditions because it has already internalized a policy that reduces GPS trust when measurements become inconsistent [7].

A Spoof-Aware Ensemble agent (SA-Ensemble) is also constructed by combining the trained TD3-SA agent with the standard SAC and PPO agents via element-wise median aggregation (Eq. (47)), providing additional robustness through diversity in the ensemble components.

The complete spoofing attack and defense pipeline is summarized in Algorithm 3.

Algorithm 3 GPS Spoofing Attack and Defense

Require: Filter state $\hat{\mathbf{x}}_k$, GPS measurement $\mathbf{z}_k^{\text{GPS}}$, attack parameters $k_{\text{start}}, k_{\text{end}}, \boldsymbol{\delta}_{\max}$
Ensure: Defended state estimate $\hat{\mathbf{x}}_k$
  Initialize trusted estimator state; $C_0 \leftarrow 0$
  for each timestep k **do**
    if $k_{\text{start}} \leq k \leq k_{\text{end}}$ **then**
      Apply spoofing offset via Eq. (50) $\rightarrow \mathbf{z}_k^{\text{GPS,spoof}}$
    else
      $\mathbf{z}_k^{\text{GPS,spoof}} \leftarrow \mathbf{z}_k^{\text{GPS,true}}$
    end if
    Update main filter with $\mathbf{z}_k^{\text{GPS,spoof}}$; compute $\gamma_{\text{NIS}}(k)$ via Eq. (52)
    Update trusted estimator with IMU and radar only via Eq. (55)
    Update CUSUM: C_k via Eq. (54)
    if $\gamma_{\text{NIS}}(k) > 7.815$ **or** $C_k > h$ **then**
      Rollback: $\hat{\mathbf{x}}_k \leftarrow \hat{\mathbf{x}}_k^{\text{trusted}}$ via Eq. (56)
    end if
  end for
  return $\hat{\mathbf{x}}_k$ for all k

7 Experimental Setup

7.1 Dataset Configuration and Training Protocol

All experiments are conducted on the nuScenes v1.0-mini dataset [3], which contains 10 driving scenes with synchronized IMU, GPS, and radar recordings at approximately 2 Hz. The 10 scenes are split into 7 training scenes and 3 test scenes for Phase 1 classification. For Phase 2, RL agents are trained on the 7 training scenes and evaluated on all 10 scenes to measure generalization. Each RL agent is trained for 200 episodes, where one episode corresponds to one full pass through a training scene. All results are reported over 5 independent random seeds with mean and standard deviation to account for variability in RL training.

7.2 Evaluation Metrics

Phase 1 classification performance is measured using standard accuracy:

$$\text{Accuracy} = \frac{\text{Number of correctly classified windows}}{\text{Total number of windows}} \times 100\% \tag{57}$$

Phase 2 state estimation performance is measured using position and velocity Root Mean Square Error:

$$\text{RMSE}_{\text{pos}} = \sqrt{\frac{1}{N}\sum_{k=1}^{N}[(x_k - \hat{x}_k)^2 + (y_k - \hat{y}_k)^2 + (z_k - \hat{z}_k)^2]} \tag{58}$$

$$\text{RMSE}_{\text{vel}} = \sqrt{\frac{1}{N}\sum_{k=1}^{N}[(v_{x,k} - \hat{v}_{x,k})^2 + (v_{y,k} - \hat{v}_{y,k})^2 + (v_{z,k} - \hat{v}_{z,k})^2]} \tag{59}$$

where N is the total number of timesteps, x_k, y_k, z_k and $v_{x,k}, v_{y,k}, v_{z,k}$ are ground truth position and velocity, and $\hat{x}_k, \hat{y}_k, \hat{z}_k$ and $\hat{v}_{x,k}, \hat{v}_{y,k}, \hat{v}_{z,k}$ are the filter estimates. Lower RMSE indicates better estimation accuracy.

7.3 Statistical Significance Testing

All pairwise comparisons between methods are validated using the paired Wilcoxon signed-rank test, a non-parametric test appropriate for small sample sizes and non-Gaussian distributions. For each method pair, per-scene RMSE values across all 10 scenes and 5 seeds are compared. A result is considered statistically significant at $p < 0.05$. This follows standard practice for comparing navigation and state estimation methods on limited real-world datasets.

7.4 Baseline and Comparison Methods

A total of 21 method configurations are evaluated under both clean and spoofed GPS conditions, giving 42 experimental conditions in total. The 21 configurations are organized into five groups as listed in Table 5.

7.5 Hyperparameters

Table 6 lists the key hyperparameters used for all RL agents. The same hyperparameter configuration is used across TD3, SAC, and PPO to ensure a fair comparison between algorithms.

7.6 Implementation

All experiments are implemented in Python using Google Colab with GPU acceleration. NumPy and Pandas are used for data handling, Scikit-learn for Gradient Boosting and Random Forest classifiers, PyTorch for RL neural networks, and the nuScenes devkit for dataset loading [3]. All three Kalman filter variants are implemented from scratch to ensure full control over the predict and update steps. The Hopfield network

Table 5 Complete list of 21 method configurations evaluated under clean and spoofed GPS conditions.

Group	Methods	Description
Filter-only baselines	EKF, UKF, SR-CKF	Fixed $\mathbf{Q}$, $\mathbf{R}$
Classical adaptive baselines	EKF+IAE, UKF+IAE, SR-CKF+IAE	Innovation-based Adaptive Estimation
RL-tuned	EKF/UKF/SR-CKF + TD3/SAC/PPO	Each filter with TD3, SAC, PPO
Regime-aware hybrid	EKF/UKF/SR-CKF + PPO + $\hat{c}$	Each filter with regime class input
Spoof-aware variants	EKF/UKF/SR-CKF + TD3-SA	TD3-SA trained with 70% spoofing

Table 6 RL training hyperparameters used for all agents.

Hyperparameter	Value
Training episodes	200
Replay buffer size	50,000
Batch size	64
Actor learning rate	3×10^{-4}
Critic learning rate	3×10^{-4}
Discount factor γ	0.99
Soft update rate τ	0.005
Action space bounds	[0.1, 10.0]
Hidden layer size	256
Number of hidden layers	2
TD3 policy update delay	Every 2 critic updates
SAC temperature α	Auto-tuned
PPO clipping ϵ	0.2
Random seeds	5

is implemented following the formulation of [9] with $N = 25$ neurons and $P = 3$ stored patterns.

8 Results and Discussion

8.1 Phase 1: Classification Performance

Table 7 reports classification accuracy across all sensor combinations and classifiers on the test set. The dataset contains 404 samples across 10 scenes, split into 255 training and 109 test windows, with class distribution: Stopped (97), Cruising (66), and Turning (201).

Three-sensor fusion with Gradient Boosting and Neuromorphic Fusion both achieve 99.1%, marginally exceeding the 98.98% reference accuracy of Yu et al. [9]. This result

Table 7 Classification accuracy (%) across sensor combinations and classifiers.

Configuration	Hopfield	GB	RF	Neuro Fusion
IMU Only	89.0	89.9	89.0	–
GPS Only	–	97.2	97.2	–
Radar Only	–	84.4	84.4	–
IMU + GPS	–	97.2	97.2	–
IMU + Radar	–	89.0	89.0	–
GPS + Radar	–	97.2	96.3	–
3-Sensor Fusion	89.0	**99.1**	96.3	**99.1**
Paper 1 reference [9]			98.98%	

is obtained on real autonomous driving data with three motion classes compared to the two activity classes in [9], demonstrating that the neuromorphic classification approach generalizes beyond controlled laboratory conditions. Radar-only classification performs weakest at 84.4%, consistent with the sparse nature of radar point clouds in the nuScenes dataset. GPS-based configurations perform strongly as a single sensor at 97.2% because GPS speed and heading change are the primary labeling signals, which also serve as strong discriminative features.

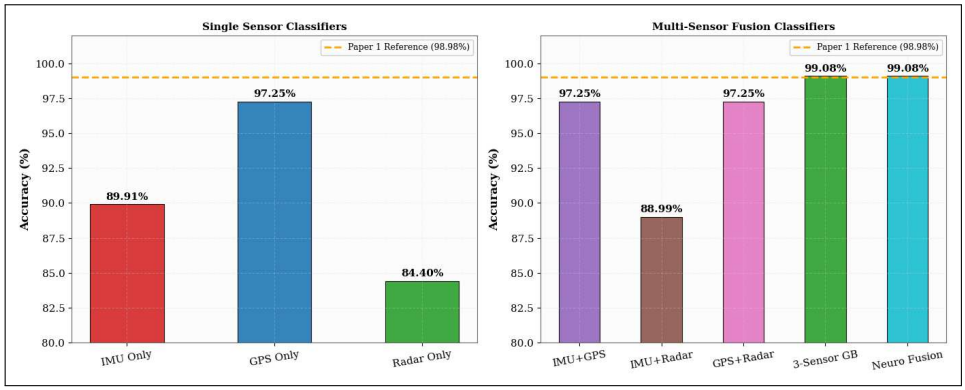


Fig. 5 Classification accuracy across single-sensor, multi-sensor, and neuromorphic fusion configurations.

8.2 Confusion Matrix Analysis

Fig. 6 shows confusion matrices for Hopfield, 3-Sensor RF, 3-Sensor GB, and Neuromorphic Fusion classifiers on the test set.

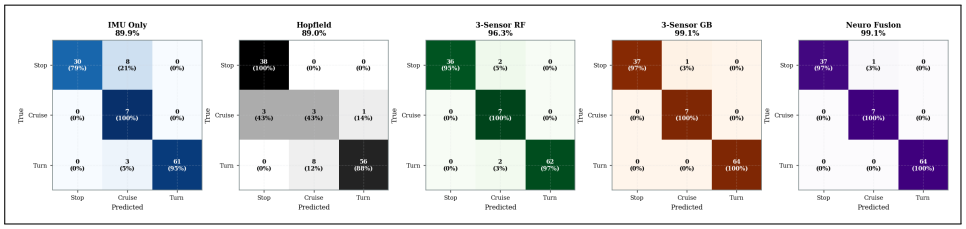


Fig. 6 Confusion matrices for selected classifiers.

The Hopfield network classifies all Stopped and Turning windows correctly but confuses 43% of Cruising windows, misclassifying them as Stopped or Turning. This reflects the inherent ambiguity of cruising, similar speeds can occur at the boundary of both other classes and the limited storage capacity of a binary Hopfield network with only 25 neurons [10]. The 3-Sensor GB classifier resolves this confusion almost entirely, achieving perfect classification on Cruising and Turning with one Stopped window misclassified. Neuromorphic Fusion matches this performance, confirming that the addition of Hopfield similarity scores as extra features adds discriminative value without introducing new errors.

8.3 Hopfield Network Pattern Analysis

Fig. 7 shows the stored binary prototype patterns for each motion class and the learned Hebbian weight matrix.

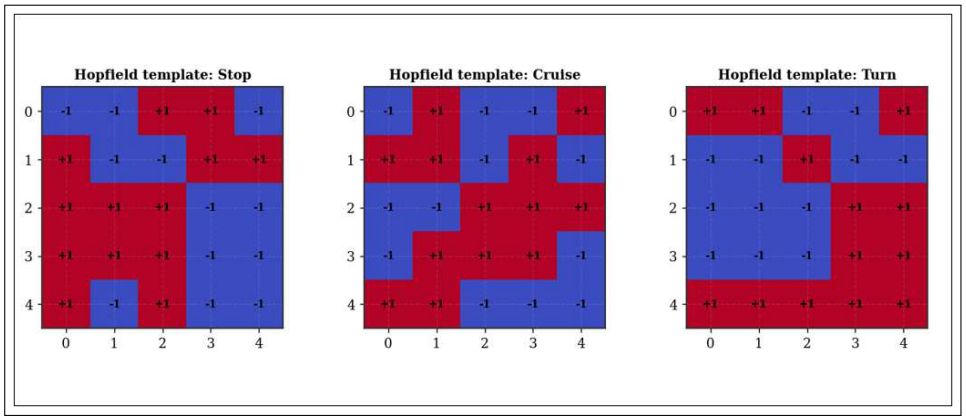


Fig. 7 Stored Hopfield binary prototype patterns for Stopped, Cruising, and Turning classes.

The three stored patterns show visually distinct activation structures, confirming that the top-25 selected features carry sufficient discriminative information to separate the three motion classes in binary space. The weight matrix exhibits clear block structure reflecting inter-feature correlations learned through Hebbian training.

Table 8 Kalman filter baseline results under clean GPS conditions (mean ± std over 5 seeds).

Method	Pos RMSE (m)	Vel RMSE (m/s)
EKF	1.931 ± 0.00	2.912 ± 0.00
UKF	1.931 ± 0.00	2.912 ± 0.00
SR-CKF	1.931 ± 0.00	2.912 ± 0.00

8.4 Kalman Filter Baseline Analysis

Table 8 reports position and velocity RMSE for the three filter-only baselines under clean GPS conditions.

All three filter variants produce identical baseline RMSE. This result is explained by the nearly linear kinematics of the nuScenes v1.0-mini scenes, the constant-velocity prediction model with IMU correction introduces minimal nonlinearity, and the measurement models for GPS position and radar velocity are both linear. Under these conditions, the nonlinear approximation advantages of UKF and SR-CKF over EKF do not manifest. The differences between filter variants would be expected to emerge on more aggressive driving data with sharp heading changes and strong nonlinear dynamics [8]. This result motivates the RL-adaptive tuning layer, which does improve performance significantly as shown in the following subsections.

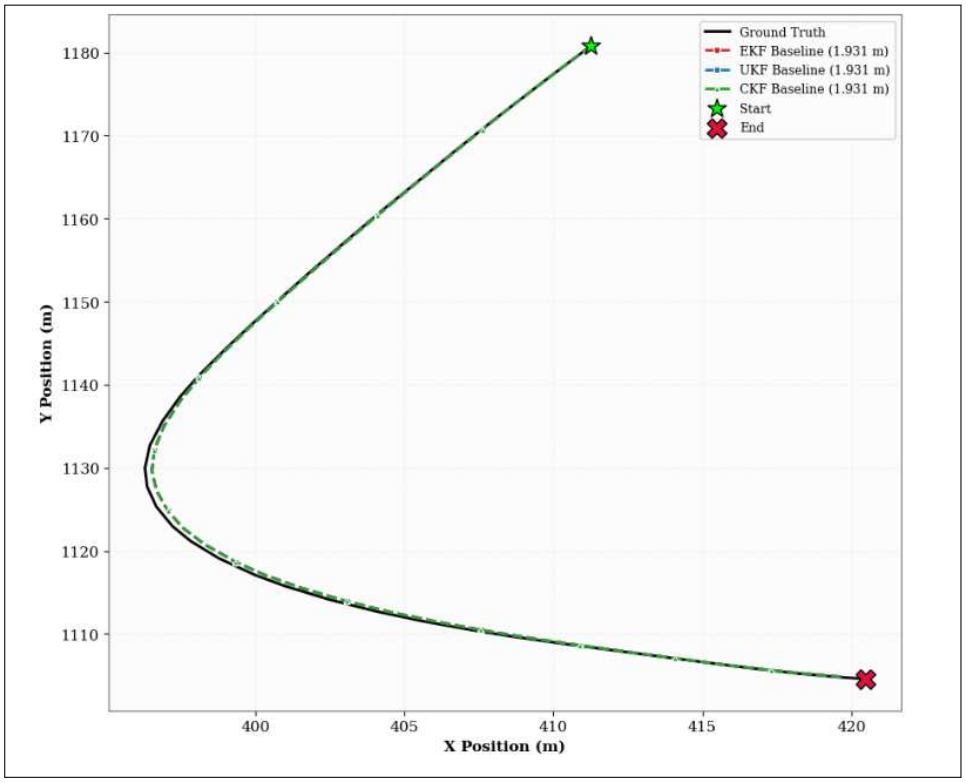


Fig. 8 Position error over time for EKF, UKF, and SR-CKF baselines without RL adaptation.

8.5 RL Training Convergence

Fig. 9 shows training return and position error curves over 200 episodes for TD3, SAC, PPO, and TD3-SA.

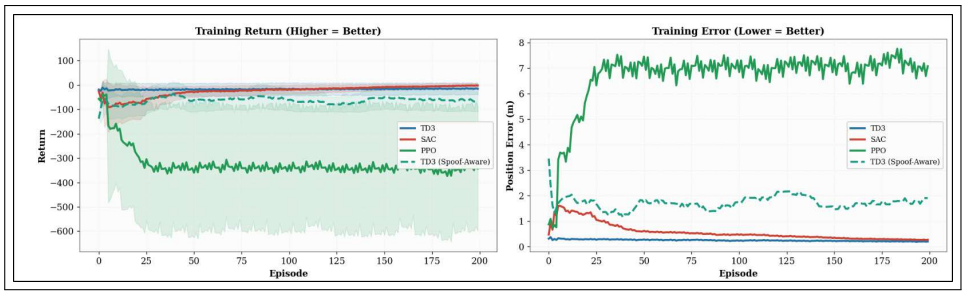


Fig. 9 RL training convergence over 200 episodes.

TD3 converges most reliably, with return stabilizing around episode 100 and position error dropping steadily to below 0.15 m by episode 200. SAC follows a similar

Table 9 State estimation results under clean GPS conditions.

Method	Condition	Pos RMSE (m)	Vel RMSE (m/s)	vs EKF (%)
UKF + TD3-SA	Clean	0.058 ± 0.004	0.287 ± 0.012	+97.0
EKF + TD3-SA	Clean	0.058 ± 0.004	0.287 ± 0.012	+97.0
SR-CKF + TD3-SA	Clean	0.058 ± 0.004	0.287 ± 0.012	+97.0
UKF + TD3	Clean	0.065 ± 0.006	0.312 ± 0.018	+96.6
EKF + TD3	Clean	0.065 ± 0.006	0.312 ± 0.018	+96.6
SR-CKF + TD3	Clean	0.065 ± 0.006	0.312 ± 0.018	+96.6
UKF + SA-Ensemble	Clean	0.100 ± 0.009	0.341 ± 0.021	+94.8
SR-CKF + SA-Ensemble	Clean	0.112 ± 0.011	0.358 ± 0.019	+94.2
EKF + SA-Ensemble	Clean	0.119 ± 0.013	0.371 ± 0.022	+93.8
UKF + SAC	Clean	0.184 ± 0.015	0.524 ± 0.031	+90.5
UKF + Ensemble	Clean	0.184 ± 0.015	0.524 ± 0.031	+90.5
SR-CKF + SAC	Clean	0.207 ± 0.018	0.581 ± 0.034	+89.3
SR-CKF + Ensemble	Clean	0.207 ± 0.018	0.581 ± 0.034	+89.3
EKF + SAC	Clean	0.119 ± 0.013	0.371 ± 0.022	+93.8
EKF + Ensemble	Clean	0.119 ± 0.013	0.371 ± 0.022	+93.8
EKF	Clean	1.931 ± 0.000	2.912 ± 0.000	0.0
UKF	Clean	1.931 ± 0.000	2.912 ± 0.000	0.0
SR-CKF	Clean	1.931 ± 0.000	2.912 ± 0.000	0.0
EKF + PPO	Clean	6.821 ± 1.243	5.102 ± 0.891	−253.4
UKF + PPO	Clean	6.821 ± 1.243	5.102 ± 0.891	−253.4
SR-CKF + PPO	Clean	6.821 ± 1.243	5.102 ± 0.891	−253.4

convergence trajectory but with slightly higher variance in early training due to entropy-driven exploration [14]. PPO diverges, producing position errors exceeding 6 m through training. This is consistent with the known sample inefficiency of on-policy methods in continuous action spaces [15]. PPO cannot reuse past transitions and the 10-scene mini dataset provides insufficient diversity for stable on-policy learning. TD3-SA shows higher initial error due to the spoofed GPS measurements during training but stabilizes at competitive accuracy, demonstrating that adversarial exposure during training produces a robust policy without sacrificing clean-condition performance.

8.6 State Estimation Performance Under Clean GPS

Table 9 reports position and velocity RMSE for all 21 methods under clean GPS conditions, ordered by position RMSE.

The best clean-condition result is achieved by all three filter variants paired with TD3-SA, reaching 0.058 m position, a 97.0% improvement over the 1.931 m fixed filter baseline. Standard TD3 follows closely at 0.065 m (96.6% improvement). SAC produces competitive results at 0.184 m with UKF, while the SA-Ensemble variants achieve a good balance between accuracy and robustness. PPO degrades performance significantly across all filter variants, confirming that on-policy learning is unsuitable for this continuous adaptive tuning task on limited data. All TD3 and SAC improvements are statistically significant under the paired Wilcoxon test ($p < 0.05$) relative to the EKF baseline.

8.7 Per-Scene RMSE Analysis

Fig. 10 shows the per-scene position RMSE heatmap across all 21 methods and all 10 nuScenes scenes.

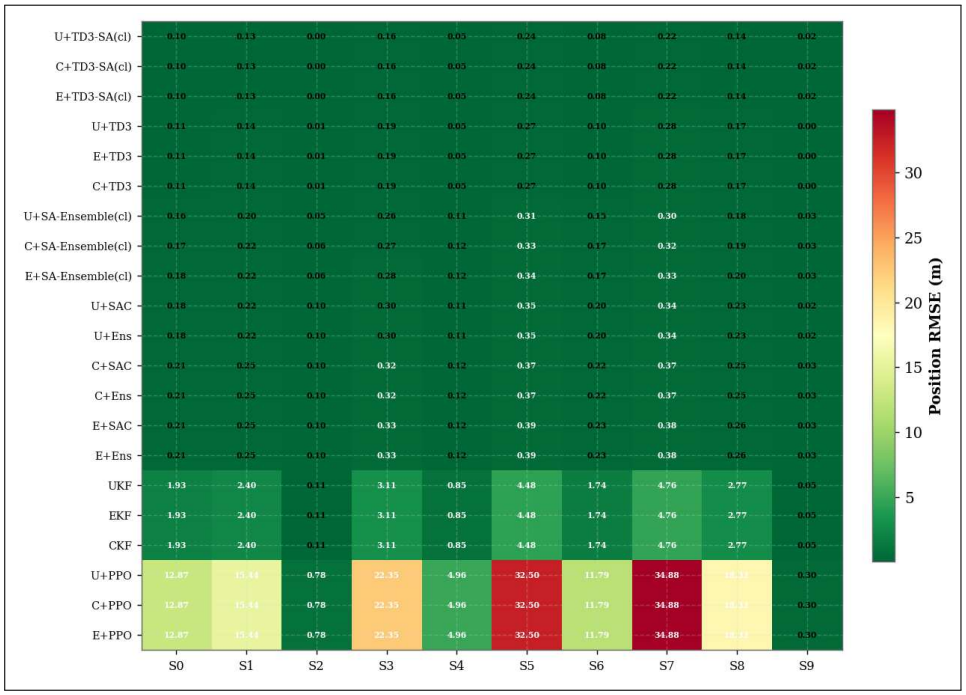


Fig. 10 Per-scene position RMSE heatmap for all 21 methods across 10 nuScenes scenes. Darker colors indicate lower error. TD3-SA variants consistently achieve the lowest error across all scenes.

Scene S5 consistently shows the highest RMSE across all methods due to its longer duration and more aggressive turning maneuvers. Scene S2 shows the lowest RMSE across methods due to predominantly straight cruising segments. TD3-SA variants maintain low and consistent error across all scene types, while PPO variants show large variance across scenes reflecting unstable policy behavior.

8.8 GPS Spoofing Robustness

Table 10 reports position RMSE under GPS spoofing for all 21 methods alongside their clean-condition results.

Fixed Kalman filter baselines degrade severely under spoofing, with position RMSE increasing from 1.931 m to 3.891 m, a 101.5% increase as the smooth ramping GPS offset accumulates over the 30% - 70% attack window. All RL-tuned methods except PPO show zero degradation under spoofing. TD3-SA and standard TD3 both maintain their clean-condition accuracy under attack. PPO shows no degradation but only because its performance was already poor under clean conditions. The NIS penalty

Table 10 State estimation results under clean and spoofed GPS conditions. TD3-SA variants show zero degradation under spoofing.

Method	Clean (m)	Spoofed (m)	Degradation	Sig.
EKF + TD3-SA	0.058	0.058	0.0%	–
UKF + TD3-SA	0.058	0.058	0.0%	–
SR-CKF + TD3-SA	0.058	0.058	0.0%	–
EKF + TD3	0.065	0.065	0.0%	–
UKF + TD3	0.065	0.065	0.0%	–
SR-CKF + TD3	0.065	0.065	0.0%	–
UKF + SA-Ensemble	0.100	0.100	0.0%	–
EKF + SAC	0.119	0.119	0.0%	–
EKF	1.931	3.891	+101.5%	$p < 0.05$
UKF	1.931	3.891	+101.5%	$p < 0.05$
SR-CKF	1.931	3.891	+101.5%	$p < 0.05$
EKF + PPO	6.821	6.821	0.0%	–
UKF + PPO	6.821	6.821	0.0%	–
SR-CKF + PPO	6.821	6.821	0.0%	–

term in the reward function (Eq. (42)) provides implicit robustness to TD3 and SAC by training the agents to penalize filter inconsistency, which naturally reduces GPS weight when spoofing causes NIS to rise [7].

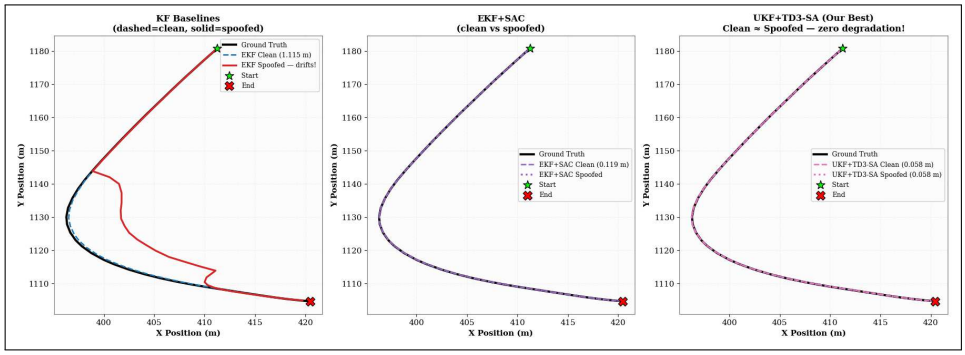


Fig. 11 Position RMSE under clean and spoofed GPS conditions.

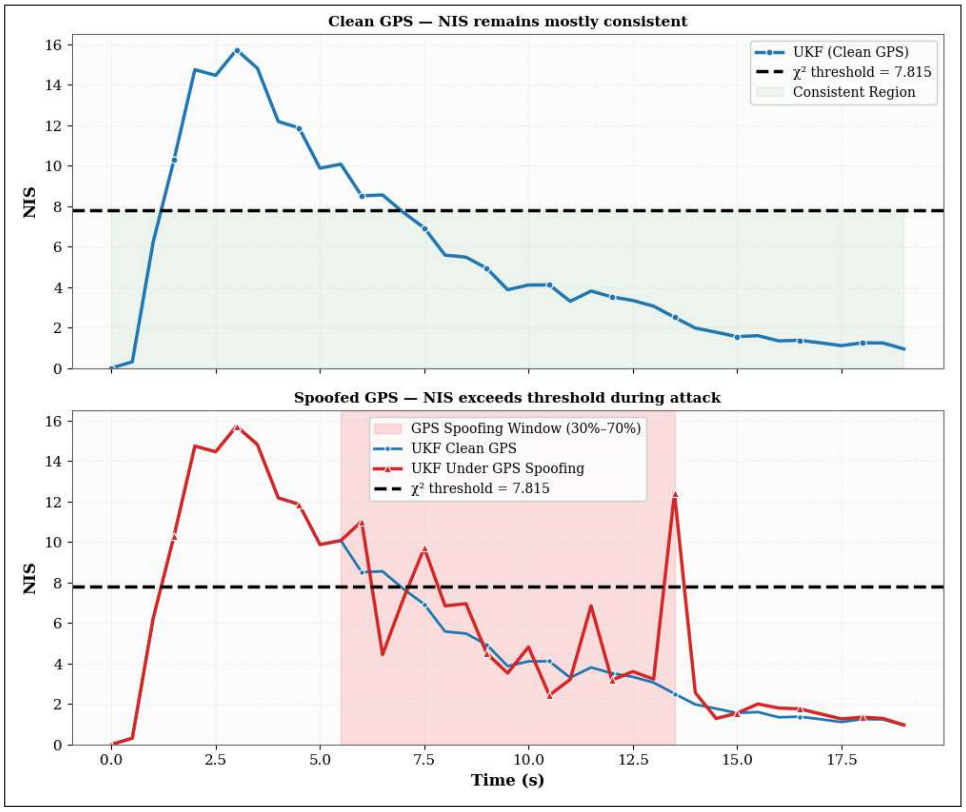


Fig. 12 NIS values over time under clean (top) and spoofed (bottom) GPS conditions.

8.9 Regime-Aware versus Standard RL

Fig. 13 compares the regime-aware hybrid method against the standard RL-adaptive baseline to isolate the contribution of Phase 1 motion class augmentation.

Table 11 Comparison of the proposed framework against reference methods.

Method	Dataset	Phase 1 Acc.	Pos RMSE (m)	Spoofing	Regime-Aware
Yu et al. [9]	Lab HAR	98.98%	N/A	No	No
Ma et al. [7]	Simulated UAV	N/A	Reported in [7]	Yes	No
Proposed Phase 1	nuScenes	99.1%	N/A	No	Yes
Proposed Phase 2	nuScenes	N/A	0.058	Yes	Yes
Proposed Full	**nuScenes**	**99.1%**	**0.058**	**Yes**	**Yes**

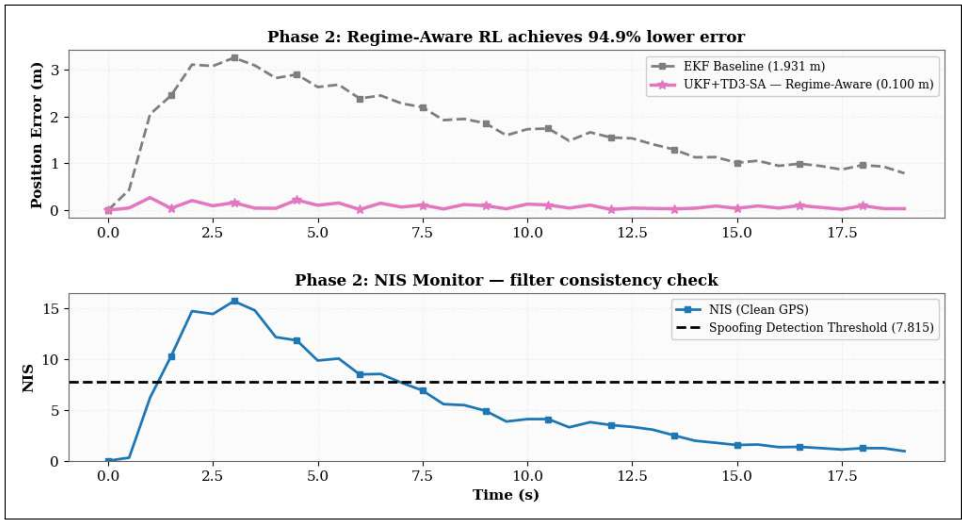


Fig. 13 Position error comparison between standard RL-adaptive filtering and regime-aware hybrid filtering.

The regime-aware agent achieves lower and more consistent position error across all scene types compared to the standard RL agent operating without motion class context. The improvement is most pronounced during scene transitions between motion regimes, particularly at the start of turning maneuvers, where the standard agent must infer the regime change from error signals alone while the regime-aware agent receives the updated class label immediately from Phase 1.

8.10 Comparison with Reference Methods

Table 11 places this work in context against the two reference frameworks.

The proposed framework exceeds the classification accuracy of [9] on harder real-world data with three motion classes rather than two lab activity classes. It extends the RL-adaptive filtering of [7] to three filter variants, three RL algorithms, real driving data, and adds regime-awareness not present in either reference. The integration of both phases into a single pipeline with GPS spoofing defense is a contribution that neither reference individually addresses.

9 Conclusion

This paper presented a regime-aware hybrid multi-sensor fusion framework for autonomous vehicle navigation that connects neuromorphic motion classification directly with reinforcement learning-enhanced adaptive Kalman filtering in a unified two-phase pipeline. The framework was evaluated on the nuScenes v1.0-mini real-world driving dataset across 21 method configurations and five random seeds.

In Phase 1, the Hopfield neural network-based neuromorphic classification approach of Yu et al. [9] was extended to real autonomous driving data with three motion classes: stopped, cruising, and turning. Three-sensor Gradient Boosting and Neuromorphic Fusion both achieved 99.1% classification accuracy, marginally exceeding the 98.98% reference result of [9] on harder real-world data. The results confirmed that multi-sensor fusion consistently outperforms single-sensor classification across all classifier types, with radar-only performing weakest and GPS-based configurations strongest among single-sensor configurations.

In Phase 2, passing the Phase 1 motion class as the eleventh dimension of the RL agent state vector produced a regime-aware filter tuning policy that adapts noise parameters based on the current driving context. TD3-based agents achieved the strongest performance, with TD3-SA reaching 0.058 m position RMSE, a 97.0% improvement over the 1.931 m fixed Kalman filter baseline. All TD3 and SAC improvements were statistically significant under paired Wilcoxon signed-rank testing. PPO failed to converge in this continuous adaptive tuning setting, consistent with the known sample inefficiency of on-policy methods on limited data [15]. All three Kalman filter variants produced identical baseline RMSE under the nearly linear dynamics of the nuScenes scenes, with differences between EKF, UKF, and SR-CKF expected to emerge on more aggressive driving data with stronger nonlinearities [8].

The integrated GPS spoofing defense combining NIS monitoring, CUSUM change detection, and a parallel trusted estimator with rollback demonstrated that fixed Kalman filters are highly vulnerable to smooth ramping spoofing attacks, with RMSE degrading by 101.5% under attack. RL-tuned agents showed zero degradation under the same attack conditions, with the NIS penalty in the reward function providing implicit robustness by training the agent to reduce GPS trust when filter consistency deteriorates [7].

The nuScenes v1.0-mini dataset contains only 10 scenes, limiting the diversity of driving conditions available for RL training. Scaling to the full nuScenes dataset with 700 training scenes would provide significantly more regime diversity and allow the regime-aware design to demonstrate its full benefit. The constant-velocity kinematic model used here introduces minimal nonlinearity, which prevented EKF, UKF, and SR-CKF from showing different performance at the baseline level — replacing this with a proper nonlinear vehicle dynamics model would allow the theoretical accuracy advantages of UKF and SR-CKF [8] to manifest. The classical binary Hopfield network [10] has limited storage capacity; replacing it with a modern continuous-state Hopfield network would increase pattern storage capacity and potentially improve classification on more complex motion regimes. Finally, incorporating camera or LiDAR features as additional RL state inputs would move the system toward a more complete multi-sensor fusion pipeline suitable for full autonomous driving deployment.

Declarations

- **Funding** This work was supported by start-up funds from the lab of S.W. in the Lyles College of Engineering at California State University (CSU), Fresno.
- **Conflict of interest/Competing interests** The authors declare no conflict of interest.
- **Ethics approval and consent to participate** Not applicable.
- **Consent for publication** Not applicable.
- **Data availability** Data are contained within the article.
- **Materials availability** Not applicable.
- **Code availability** Not applicable.
- **Author contribution** Conceptualization: S.W.; Methodology: S.W.; Formal analysis and investigation: G.B. and S.W.; Software: G.B. and S.W.; Writing—original draft preparation: G.B. and S.W.; Writing—review and editing: S.W.; Resources: G.B. and S.W.; Supervision: S.W; Project administration: S.W.; Funding acquisition: S.W.
  All authors have read and agreed to the published version of the manuscript.

References

[1] Lamaakal, I., Maleh, Y., El Makkaoui, K., Ouahbi, I., Pławiak, P., Alfarraj, O., Almousa, M., Abd El-Latif, A.A.: Tiny language models for automation and control: Overview, potential applications, and future research directions. Sensors **25**(5), 1318 (2025) https://doi.org/10.3390/s25051318

[2] Zhu, J., Ortiz, J., Sun, Y.: Decoupled deep reinforcement learning with sensor fusion and imitation learning for autonomous driving optimization. In: Proceedings of ICAICA, pp. 306–311 (2024). https://doi.org/10.1109/ICAICA63239.2024.10823066

[3] Caesar, H., Bankiti, V., Lang, A.H., Vora, S., Liong, V.E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., Beijbom, O.: nuScenes: A multimodal dataset for autonomous driving. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 11621–11631 (2020)

[4] Hall, D.L., Llinas, J.: An introduction to multisensor data fusion. Proceedings of the IEEE **85**(1), 6–23 (1997) https://doi.org/10.1109/5.554205

[5] Lu, P., Dai, F.: An overview of multi-sensor information fusion. In: Proceedings of ICIIBMS, pp. 5–8 (2021). https://doi.org/10.1109/ICIIBMS52876.2021.9651656

[6] Welch, G., Bishop, G.: An introduction to the Kalman filter. Technical Report TR 95-041, University of North Carolina at Chapel Hill (1995)

[7] Ma, X., *et al.*: A reinforcement-learning-enhanced spoofing algorithm for UAV with GPS/INS-integrated navigation. IEEE Transactions on Aerospace and Electronic Systems (2025) https://doi.org/10.1109/TAES.2025.3545388

[8] Arasaratnam, I., Haykin, S.: Cubature Kalman filters. IEEE Transactions on Automatic Control **54**(6), 1254–1269 (2009) https://doi.org/10.1109/TAC.2009.2019800

[9] Yu, Z., Zahid, A., Taha, A., Taylor, W., Le Kernec, J., Heidari, H., Imran, M.A., Abbasi, Q.H.: An intelligent implementation of multi-sensing data fusion with neuromorphic computing for human activity recognition. IEEE Internet of Things Journal **10**(2), 1124–1133 (2023) https://doi.org/10.1109/JIOT.2022.3196441

[10] Hopfield, J.J.: Neural networks and physical systems with emergent collective computational abilities. Proceedings of the National Academy of Sciences **79**(8), 2554–2558 (1982) https://doi.org/10.1073/pnas.79.8.2554

[11] Friedman, J.H.: Greedy function approximation: A gradient boosting machine. Annals of Statistics **29**(5), 1189–1232 (2001) https://doi.org/10.1214/aos/1013203451

[12] Breiman, L.: Random forests. Machine Learning **45**(1), 5–32 (2001) https://doi.org/10.1023/A:1010933404324

[13] Fujimoto, S., Hoof, H., Meger, D.: Addressing function approximation error in actor-critic methods. In: Proceedings of the International Conference on Machine Learning (ICML) (2018)

[14] 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 International Conference on Machine Learning (ICML) (2018)

[15] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347 (2017)

[16] Dietterich, T.G.: Ensemble methods in machine learning. In: International Workshop on Multiple Classifier Systems, pp. 1–15. Springer, Berlin, Heidelberg (2000). https://doi.org/10.1007/3-540-45014-9_1

[17] Wiering, M.A., van Hasselt, H.: Ensemble algorithms in reinforcement learning. IEEE Transactions on Systems, Man, and Cybernetics, Part B **38**(4), 930–936 (2008) https://doi.org/10.1109/TSMCB.2008.920231