systems

MDPI

Article

AI Agents for Supply Chain: Path Following Control Using Adaptive Kalman Filter Reinforcement Learning

Petru Stefan Vasian [1], Ionut Dancau [1], Catalin Dumitrescu [2,*], Eduard-Cristian Popovici [2] and Petrica Ciotirnae [3]

[1] Doctoral School of Electronics, Telecommunications, and Information Technology, National University of Science and Technology POLITEHNICA from Bucharest, 060042 Bucharest, Romania; petru_stefan.vasian@stud.trans.upb.ro (P.S.V.); alexandru.dancau@stud.etti.upb.ro (I.D.)
[2] Faculty of Electronics, Telecommunications, and Information Technology, National University of Science and Technology POLITEHNICA from Bucharest, 060042 Bucharest, Romania; eduard.popovici@upb.ro
[3] Communications Department of Military Technical Academy "Ferdinand I", 39-49 George Coșbuc Avenue, 050141 Bucharest, Romania; petrica.ciotirnae@mta.ro
* Correspondence: catalin.dumitrescu@upb.ro

Abstract

Path following control is a fundamental problem in robotics and autonomous systems, essential for the navigation of vehicles, drones, and mobile robots. The goal is to maintain a system on a predefined path despite external perturbations and complex system dynamics. Reinforcement learning (RL) offers a promising approach to overcome these limitations, allowing agents to learn control rules for following the optimal road route correlated with environmental conditions. Among RL algorithms, the Soft Actor-Critic (SAC) method stands out by combining the advantages of calculations made based on variables and rules. They will use an "actor" to select actions and a "critic" to evaluate the quality of these actions, facilitating efficient and stable learning. This paper proposes the integration of a Parallel Adaptive Kalman Filters (KF) with a Soft Actor-Critic (SAC) agent to create a highly robust system for path following control (PFC). This architecture combines optimal state estimation with maximum-entropy reinforcement learning to handle sensor noise, environmental disturbances, and continuous actuator controls.

Keywords: Soft Actor-Critic; Kalman filter; path control; reinforcement learning

1. Introduction

Currently, increasingly, efficient logistics is the winning element in attracting and retaining customers. Prestigious companies know that they can attract more customers, that they can retain them only if the distribution meets their demands. Second, logistics represents an important cost element in the activity of any company [1]. Logistics transport has rising costs, and companies must intensify their concerns for development of efficient logistics, to improve the efficiency of their physical distribution system, to visibly reduce their costs, with positive implications for both them and their customers. There are still serious gaps in this area, with modern decision-making tools currently being used quite little to coordinate transport activities, efficiently manage inventories, and efficiently store and handle goods [2].

The third consideration is based on developments in information technology that have a decisive influence on logistics, contributing greatly to physical conditioning distribution of products. Regarding this, the increasing use of control, electronic point-of-sale scanning

Check for updates

Academic Editor: Wen-Chyuan Chiang

Received: 16 June 2026
Revised: 5 July 2026
Accepted: 15 July 2026
Published: 17 July 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

systems, uniform product codes, satellite registration, electronic data interchange, and electronic fund transfers have enabled companies to create advanced systems for order processing, inventory control, and road supply chain planning and optimization. In the context of the globalization of the international economy and the expansion of e-commerce, the dynamics of supply chains have become an essential factor in ensuring organizational competitiveness. Among the available modes of transport, road transport remains the central pillar of global logistics, due to its unique flexibility, territorial penetration capacity and the possibility of making door-to-door deliveries. However, the major dependence on land infrastructure exposes this type of transport to a complex series of uncertainties and operational challenges [3].

Road supply route management is a strategic and operational set of planning, coordination and control of goods flows on the roads. It is not limited to establishing a route between point A and point B, but involves a structural analysis determined by the following variables: fleet configuration, legal tonnage restrictions, management of rest times for drivers, fuel price instability and traffic predictability. In an economic environment where customers demand ever shorter delivery times, the efficiency of the choice of supply routes directly influences the profitability of a company.

Today, digital transformation is redefining this field. The shift from manual, empirical planning to the use of artificial intelligence algorithms, transport management systems (TMS) and advanced telematics allows for real-time resource optimization. Reducing empty kilometers, choosing optimal transport routes and minimizing times at unloading points have become zero priorities. Moreover, in the current context of environmental regulations, road route management acquires a mandatory ecological dimension, with the direct aim of reducing the carbon footprint of transport [4]. In this context, the efficiency of transport and logistics networks directly depends on the ability to monitor and guide vehicles with mathematical precision. Path following control is a frontier discipline, located at the intersection of automatic systems engineering, robotics and intelligent logistics. This conceptual component aims at the ability of a system to guide a vehicle, be it a heavy-duty truck, a special vehicle or an autonomous vehicle, along a predefined trajectory, in conditions of maximum safety and efficiency. If route management determines which path to follow from a macroeconomic perspective, route tracking control solves the dynamic execution problem: how to keep the vehicle on that ideal path. This process involves continuous data collection using kinematic sensors, satellite navigation systems (GNSS/GPS) and advanced telematics technologies, followed by automatic or assisted adjustment of travel parameters (speed, position, steering angle, acceleration). The importance of this control is paramount in modern road transport, where environmental conditions such as road conditions, weather phenomena or the behavior of other road users can generate dangerous deviations from the route. The article aims to analyze a mathematical mechanism for controlling road route tracking. Through this analysis, the paper aims to demonstrate that intelligent routing is not just a measure to reduce operational costs, but an indispensable strategic component for the resilience and sustainability of the modern supply chain.

This study seeks to bridge that gap by addressing the following questions: (1) How can intelligent multi-agent systems enhance decision-making efficiency in supply chain? (2) What role do AI-driven predictive models play in optimizing transaction outcomes? (3) How does the integration of computational intelligence impact logistics optimization adaptability and automation? By mapping the methodology directly to these questions, the study ensures a structured approach that systematically analyzes the interplay between AI, multi-agent transactions, and route planning optimization [5].

Route tracking control is a fundamental issue in robotics and autonomous systems, essential for the navigation of vehicles, drones, and mobile robots. The goal is to keep a

system on a predefined path, despite external disturbances and complex system dynamics. The established algorithms for trajectory control currently used, proportional–integral–derivative (PID) [6], Kalman filter, predictive model control (MPC), often require detailed knowledge of the system's dynamic model and have limitations in unknown or changing (dynamic) environments [7,8].

Path following control by fusing reinforcement learning/deep reinforcement learning (RL/DRL) and Kalman filter (KF) is a cutting-edge field in autonomous systems engineering [9,10]. This hybrid approach solves a critical problem: RL algorithms provide excellent adaptability in nonlinear dynamic environments, while KF compensates for sensor noise, estimates hidden states, and recalibrates model-generated uncertainties in real time [11]. Recent literature review highlights three main ways in which RL and KF collaborate: RL for Kalman Filter Optimization (RL-driven KF): Algorithms such as DDPG (Deep Deterministic Policy Gradient) or PPO are used as agents to dynamically adjust the Q and R noise covariance matrices according to the current positioning errors. This aspect prevents the filter from diverging in nonlinear environments (e.g., low road grip for autonomous vehicles) [12,13], KF as a state estimator for the RL agent (Filter-Enhanced RL): The Kalman filter performs preprocessing of raw data from sensors (IMU, GPS, LiDAR). The current, noise-cleaned states and disturbance estimates are sent as inputs (observations) to the RL agent's policy network, massively reducing the search space and accelerating convergence [14,15]. Hierarchical Structures: KF estimates external disturbances at the dynamic level of the system, while RL adjusts the parameters of a predictive controller (MPC) or generates the nominal steering commands [16–18]. The main directions of applicability are autonomous vehicles and complex systems—research has been focused on maintaining vehicle stability at high speeds or for slippery road surfaces. The models use RL (e.g., TD3 or DDPG) combined with cubic or unscented Kalman adaptive filters to estimate the sideslip angle and lateral tire forces, thus achieving tracking error reductions of over 30–54% compared to classical methods [19,20]; drones (UAVs) in environments with interference and GPS jamming—in military or urban scenarios where the GPS signal is jammed, the recent literature introduces the use of EKF/AKF to estimate signal strength and flight path losses, the data are integrated into adaptive navigation policies based on Deep RL, allowing drones to avoid dynamic obstacles and correct their trajectory [21]; industrial robotics and logistics manipulators—in the case of robotic arms that manipulate objects on arbitrary trajectories, an AKF augmented with RL schemes (with reward shaping techniques) has proven essential for vibration attenuation and mechanical wear compensation, in this case the KF dynamically estimates nonlinear deformations and frictions, and the RL agent performs optimal trajectory planning at the millisecond level [22]. Other areas of applicability are autonomous underwater vehicles (AUVs) because underwater 3D trajectory tracking is affected by ocean currents. Recent studies combine Transformer neural networks with extended Kalman filters (A-KIT) to learn noise process online. RL algorithms solve nonlinear optimal control equations, improving positioning accuracy by up to 49% compared to conventional methods [23–27]. For the development of path tracking control strategies (RL + MPC/NMPC) the dominant trend in the last 3 years is no longer pure "end-to-end" RL control (which can generate errors for safety), but the hybrid hierarchical RL control as an optimizer for MPC, in this case agents based on Deep Deterministic Policy Gradient (DDPG), Twin Delayed DDPG (TD3) or Proximal Policy Optimization (PPO) are used to online optimize the cost functions and prediction horizon determined by MPC [28,29]; compensation of model errors: in applications using heavy logistics vehicles, where the dynamics change suddenly after loading/unloading, the RL agent learns the difference (mismatch) between the ideal kinematic model in MPC and the real vehicle, so RL applies correction commands directly on the steering angle (feedforward

steering) to cancel the stationary error in curves [30]. Unlike passenger vehicle applications, logistics-oriented applications mention that reward functions place particular emphasis on energy efficiency, reducing tire wear (by penalizing sudden steering movements), and maintaining rollover stability [31]. The comparison of the performances of the algorithms mentioned in the literature is presented in Table 1.

Table 1. Comparison of the performances of the algorithms.

	Classical Approach (MPC/PID)	Hybrid Approach (KF and RL)
Tracking accuracy	Low in non-Gaussian noise.	High; RMSEs are reduced by 30–60%.
Robustness to disturbances	Sensitive to model-reality mismatches.	Excellent; KF estimates uncertainties online.
Training time (RL)	Very long; vulnerable to rare rewards	Low; KF reduces noise in the state space.
Stability	Rigorous (via Lyapunov functions).	Hybrid; safety filters are used in the RL loop.

Reinforcement learning (RL) offers a promising approach to overcome these limitations, allowing agents to develop optimal rules for path following control in relation to environmental conditions. Among RL algorithms, the Soft Actor-Critical (SAC) method is distinguished by combining the advantages of mechanisms developed on variables and rules. Using an *actor* to select actions and a *critic* to assess efficiency of actions, facilitating effective and stable learning [32,33]. This article explores the applicability and benefits of the Kalman filters and Soft Actor-Critical filter architecture in the context of route tracking control.

Intelligent management of vehicle routes in logistics can reduce carbon emissions by up to 12–17%, becoming the central pillar of the green logistics strategy. Route optimization is no longer just a cost-cutting method, but an ecological necessity and compliance with European emission taxes (such as the CO_2-based Euro-vignette taxes). Figure 1 shows the statistics of CO_2 emissions generated by transport in the EU [34].

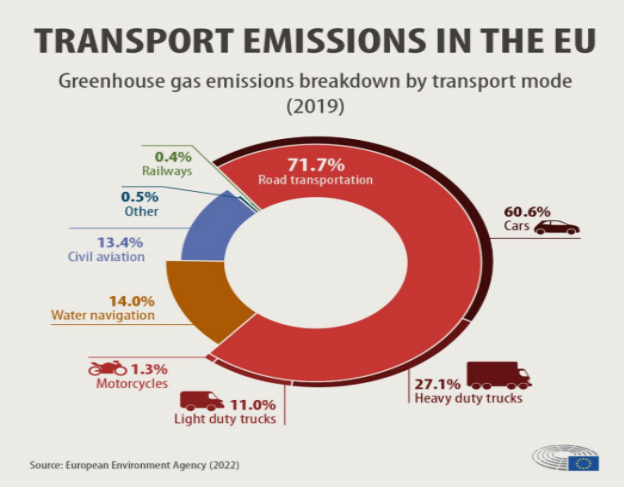


Figure 1. Statistics of CO_2 emissions generated by transport in the EU (source [1]).

Technological Optimization Strategies [35]

- Artificial Intelligence (AI)-based algorithms: Replace static route planning with dynamic models. They instantly analyze traffic history, weather forecasts and delivery windows requested by customers.
- Real-time congestion avoidance: Navigation software reroutes vehicles on the go to avoid traffic jams. This prevents engine idling, a major source of unnecessary emissions.
- Bypassing polluted areas: Integrating air quality APIs allows trucks to avoid congested urban centers or low emission zones (LEZs) during critical periods, protecting public health.

Efficient Fleet and Cargo Management

- Cargo consolidation: Maximizing the useful capacity of each vehicle reduces the total number of trips required.
- Minimizing empty trips: Planning return routes to pick up other goods (backhauling) reduces unnecessary mileage and CO2 emissions by almost 38% on the empty return segment.
- Predictive maintenance: Sensor-monitored fleets consume less fuel because they are running at optimal technical parameters (correct tire pressure, tuned engines).

The paper develops a multi-agent for path following control system using computational intelligence techniques to find the most efficient solutions for supply chain performed in logistics planning optimization. The paper has the following structure: Section 2 describes the architecture and algorithms used, to define intelligent agents and how they can influence path following control activity and a generative model of the Kalman filter and adaptive agent with the theoretical and architectural material and the Soft Actor-Critic (SAC) network that together form the generative mechanism of our agent. Section 3 presents the results from experiments conducted on an agent-based path following control architecture and in Section 4 the conclusions and future research directions are presented.

2. Materials and Methods

Our methodology proposes the integration of the Kalman filter with the hybrid Soft Actor-Critic (SAC) reinforcement learning (RL) technique for planning routes and routes in the case of logistics planning optimization. The aim is to increase the accuracy of the decision-making process for planning routes of vehicles that ensure the supply chain.

By combining the two techniques, we aimed to create an autonomous vehicle system that becomes efficient in terms of rapid adaptation to dynamic environments. Figure 2 illustrates the proposed architecture in which the Kalman filter processes data from the environment, acquired from LiDAR, video cameras or continuous actuator controls, and the SAC agent manages the execution, the system operating in a closed loop.

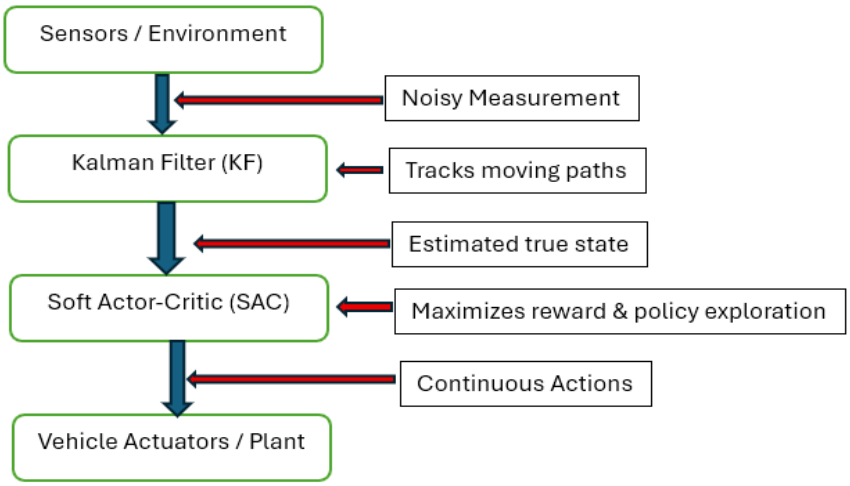


Figure 2. Architecture for path following control combining Kalman filter (KF) and Actor-Critic (SAC).

2.1. Parallel Kalman Filter Architectures for High-Accuracy Vehicle Trajectory Tracking

In the theory of guidance systems for autonomous and intelligent vehicles, the estimation of kinematic and dynamic states requires absolute precision and resilience. Modeling a vehicle's motion using a single Kalman filter (KF) has severe structural limitations when the vehicle is operating in transient driving regimes (e.g., sudden transitions from straight-line driving on a highway to spontaneous obstacle avoidance maneuvers). To overcome this drawback, architecture based on parallel Kalman filters has been developed. This mathematical approach uses multiple filters running in parallel simultaneously, each optimized for a specific kinematic/dynamic model or fed with data from different sensor subsets. This subchapter reviews the mathematical foundations of parallel structures, with a focus on the Interacting Multiple Model (IMM) algorithm and decentralized sensor fusion architectures [36,37].

The most common use of parallel filters for trajectory tracking is the use of competing kinematic models. The vehicle does not follow a single mathematical template throughout its journey; therefore, a set of M parallel Kalman filters is configured, also called the multimode approach: the Interacting Multiple Model (IMM) algorithm, each corresponding to a specific driving mode (e.g., uniform straight-line driving, constant-rate cornering, high acceleration) [38].

The vehicle transition between the M driving modes is modeled as a discrete Markov chain, governed by a transition probability matrix Π:

$$\Pi = \begin{bmatrix} p_{11} & \cdots & p_{1M} \\ \vdots & \ddots & \vdots \\ p_{M1} & \cdots & p_{MM} \end{bmatrix} \quad (1)$$

where $p_{ij} = P(m_k = j | m_{k-1} i)$ represents the probability that the vehicle switches from model i at time k − 1 to model j at time k.

Parallel IMM algorithm steps

At each time step k, the M filters interact recursively through the following mathematical structure:

Step A: Input Interaction and Mixing

Before prediction, the estimated states $\hat{x}^i_{k-1}$ and the covariances P^i_{k-1} of the filters are combined based on the conditional transition probabilities, generating mixed inputs ($\hat{x}^{0j}_{k-1}$ și P^{0j}_{k-1}) for each filter j:

$$\hat{x}^{0j}_{k-1} = \sum_{i=1}^{M} \hat{x}^i_{k-1} \mu^{(i|j)}_{(k-1|k-1)} \quad (2)$$

$$P^{0j}_{k-1} = \sum_{i=1}^{M} \mu^{(i|j)}_{(k-1|k-1)} \left[P^i_{k-1} + \left(\hat{x}^i_{k-1} - \hat{x}^{0j}_{k-1} \right) \left(\hat{x}^i_{k-1} - \hat{x}^{0j}_{k-1} \right)^T \right] \quad (3)$$

where $\mu^{(i|j)}_{(k-1|k-1)}$ are the mixing probabilities calculated from the probability vector of μ_{k-1} models.

Step B: Parallel Filtering (Prediction and Update)

Each filter j in the parallel structure independently executes its standard Extended Kalman Filter (EKF) or Unscented Kalman Filter (UKF) equations, using its own equations of motion $f^j(\cdot)$ and measurement $g^j(\cdot)$:

$$\hat{x}^{j,-}_k = f^j \left(\hat{x}^{0j}_{k-1}, u_k \right) \quad (4)$$

$$P^{j,-}_k = A^j_k P^{0j}_{k-1} \left(A^j_k \right)^T + Q^j \quad (5)$$

$$\hat{x}_k^j = \hat{x}_k^{j,-} + K_k^j + \left[z_k - g^j\left(\hat{x}_k^{j,-}\right)\right] \tag{6}$$

Step C: Updating Model Probability

The plausibility (likelihood) Λ_k^j of each filter is calculated, assuming a Gaussian distribution of the measurement residuals (contribution, note $v_k^j = z_k - g^j\left(\hat{x}_k^{j,-}\right)$):

$$\Lambda_k^j = \frac{1}{\sqrt{\left|2\pi S_k^j\right|}} exp\left(-\frac{1}{2}\left(v_k^j\right)^T\left(S_k^j\right)^{-1}v_k^j\right) \tag{7}$$

where S_k^j is the covariance matrix of the innovation of filter j. The model probability is updated Bayesian:

$$\mu_k^j = \frac{\Lambda_k^j \sum_{i=1}^M p_{ij}\mu_{k-1}^i}{\sum_{m=1}^M \Lambda_k^m \sum_{i=1}^M p_{im}\mu_{k-1}^i} \tag{8}$$

Step D: Combining States (System Output)

The final output sent to the trajectory controller is a weighted sum of the states of all filters:

$$\hat{x}_k = \sum_{j=1}^M \hat{x}_k^j \mu_k^j \tag{9}$$

$$P_k = \sum_{j=1}^M \mu_k^j \left[P_k^j + \left(\hat{x}_k^j - \hat{x}_k\right)\left(\hat{x}_k^j - \hat{x}_k\right)^T\right] \tag{10}$$

Decentralized Parallel Data Fusion (Sensor-Level Parallelism)

A second parallel architecture is used to asynchronously process massive data streams from different sensors (GNSS, LiDAR, Cameras, IMU, Odometry), avoiding blocking the central processor. Figure 3 shows the architecture for decentralized parallel data fusion.

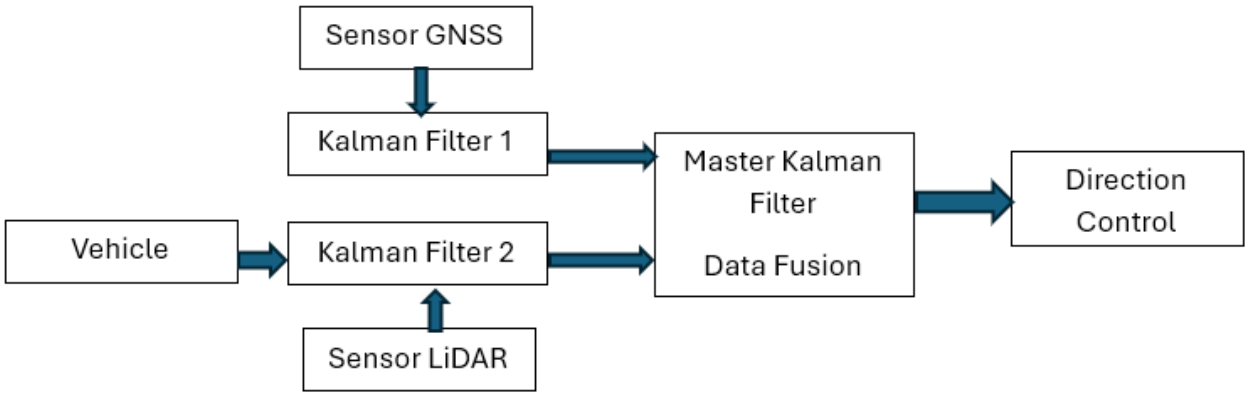


Figure 3. Computational architecture for decentralized parallel data fusion.

In this configuration, each sensor sends its raw data to its own local Kalman filter. The local filters estimate the state in parallel and send only the optimized state vectors $\hat{x}_i$ and covariance matrices P_i to a Central Fusion (Master KF).

The optimal fusion at the central level is mathematically achieved by the Millman intersection of covariances algorithm or by inverse covariance weighting:

$$P_{central}^{-1} = P_1^{-1} + P_2^{-1} + \cdots + P_N^{-1} \tag{11}$$

$$\hat{x}_{central} = P_{central}\left(P_1^{-1}\hat{x}_1 + P_2^{-1}\hat{x}_2 + \cdots + P_N^{-1}\hat{x}_N\right) \tag{12}$$

The advantages of Parallel Execution in Trajectory Control [39,40] are:

- *Elimination of Sensor Latency*: If a LiDAR sensor provides data at 20 Hz and a camera system at 60 Hz, the filters running in parallel process the information independently, at the sensors' own pace. The central fuser asynchronously retrieves the latest valid estimates, eliminating phase delays in the calculation of the steering angle.
- *Fault Tolerance:* If the GNSS sensor is affected by noise or massive reflections in the urban environment, the associated local filter will generate a P_{GNSS} covariance matrix with very large values. The central fuser will automatically downgrade the weight of this $P_{GNSS}^{-1} \rightarrow 0$ filter, relying exclusively on the parallel filters fed by the IMU and odometry to keep the vehicle on the trajectory.
- *Dynamic Adaptability*: During an emergency maneuver, the plausibility of the rectilinear model decreases instantly, and the μ weight of the parallel filter configured for tight cornering dynamic equations increases towards 1. This allows the trajectory controller to receive a faithful estimate of the lateral forces on the tires, without the delay specific to a single rigid filter.

Parallel Kalman filter-based architecture represents the ideal mathematical solution for handling the hybrid uncertainty (both geometric and behavioral) encountered in vehicle trajectory tracking. By distributing computational effort and combining estimates probabilistically or algebraically, these systems provide superior trajectory smoothing, transforming an autonomous vehicle from a reactive system into a fully predictive and safe system.

2.2. Reinforcement Learning (RL) and Soft Actor-Critic (SAC) Agent Algorithms

Soft Actor-Critic (SAC) is a reinforcement learning (RL) algorithm without policy rules and models, which provides high stability and efficiency for managing continuous action spaces. Unlike classical algorithms that only seek to maximize long-term rewards, a SAC agent is based on the Maximum Entropy Framework, simultaneously optimizing both the reward received and the degree of diversity (entropy) for determining the strategy (policy) used.

Reinforcement learning (RL) offers a promising approach to overcome these limitations, allowing agents to learn optimal rules for control by interacting with environment. Among RL algorithms, the Actor-Critical method is distinguished by combining the advantages of variable-based and rule-based implementations. For this use an actor to select actions and a critic to assess the quality of the rules necessary for actions, facilitating effective and stable learning [41]. This paper explores the applicability and advantages of the Actor-Critical architecture in the context of route tracking control.

Fundamentals of the Actor-Critical Algorithm

The Actor-Critic algorithms are a class of reinforcement learning methods that combine two main components: the actor and the critic.

Actor: Action Generation

The actor is responsible for nominating actions in correlation with the current state of the environment with which he interacts. This represents agent policy, i.e., a mapping from statuses to actions. The policy can be stochastic (the exit is probabilities for each action) or deterministic (the exit is directly the action). In route tracking control, the actor would receive information about the current state of the vehicle (e.g., position, orientation, speed, distance from the route) and generate control signals (e.g., steering angle, acceleration/deceleration) to keep the vehicle on the route. The actor's parameters are adjusted to obtain the maximum reward throughout the action. Adjustment is based on the policy gradient, which indicates the direction in which parameters need to be changed to improve performance.

The Critic: The Evaluation of Values

The critic has the role of estimating the value function (V(s)) or the action-state value function (Q(S, A)). Action state value is expressed by:

$$Q_k(S, A^c, \phi_k) \tag{13}$$

where for each parameter ϕ_k, the observation S and the action part are considered A^c. For each critic corresponds $Q_k(S, A^c, \phi_k)$ to a critical target action $Q_k(S, A^c, \phi_{tk})$ that will have the same structure and parameterization.

It evaluates how good a certain state or action taken for a state is. The critic's information is used to guide the actor's learning. Without criticism, the actor would have to explore the environment more to understand the impact of his actions, which would slow down the learning process.

The critic is trained to minimize the error obtained from the estimated value and the reward received because of the action, adjusted by the value of the next state (TD error). This error is then used to update both the actor and the critic [42].

The critical function in the SAC estimates the value of a stock in each state. To optimize itself, the algorithm uses the mean square error of the time lag based on the modified Bellman equation for entropy, by the formula:

$$y = r + \gamma(1 - d)\left[\min Q_{\theta_{targ,i}}(s', a') - \propto \log \pi_\Phi (a'|s')\right] \tag{14}$$

where

r is rewards.

γ is the discount factor.

d is the terminal status (done) indicator.

$\propto log\pi_\phi(a'|s')$ is the entropy bonus that encourages exploration.

Actor–Critic Interaction

The interaction between actor and critic is cyclical:

- The actor observes the state of the environment and chooses action.
- The environment executes action, evolution toward a new state, using grants a reward.
- The critic obtains the new status and reward and calculates the TD error, which is a measure of how surprising the agent's action was compared to expectations.
- Critic updates its own value function based on this error.
- The actor receives the TD error from the critic and adjusts his policy in the direction that led to higher rewards (or positive TD errors).

This collaboration allows for quick and a lot of stable learning compared to pure policy methods (where variance can be high) or pure value-based methods (which may struggle with continuous action spaces) [43].

Critical Actor Application in Route Tracking Control

For route tracking control, an Actor-Critic-based system would be configured as follows:

➢ ***Definition of State and Action***

For the parallel hybrid KF-SAC model to be implemented and scientifically validated, the state and action spaces cannot only be described qualitatively but must be mathematically formalized as compact multidimensional vectors. In the context of a logistics fleet, the states combine the kinematic parameters from the Kalman filter (KF) with the operational variables of the supply chain. The action space models the continuous control decisions of the vehicle. The fleet management system is modeled as a Continuous Markov Decision Process (MDP).

Formalization of the State Space and Action Space

Mathematical definition of state space ($\mathcal{S}$)

At each discrete time instant t, the complete state of a vehicle in the fleet is represented by a multidimensional vector formed by concatenating three sub-vectors: kinematic, logistic, and environmental. The state space is defined as a continuous space $\mathcal{S} \subseteq \mathbb{R}^{10}$:

$$s_t = \left[s_{cin,t}^T, s_{log,t}^T, s_{env,t}^T \right]^T \tag{15}$$

The components of the state vector are formed as follows:

the kinematic vector ($s_{cin,t} \in \mathbb{R}^4$): represents the optimal kinematic state of the vehicle, determined directly from the a posteriori estimation of KF $\left(\hat{x}_{k|k} \right)$:

$$s_{cin,t}^T = \left[\hat{p}_{x,t}, \hat{p}_{y,t}, \hat{v}_t, \hat{\theta}_t \right]^T \tag{16}$$

where $\hat{p}_{x,t}, \hat{p}_{y,t}$ are the smoothed plane coordinates (UTM projection), $\hat{v}_t$ is the estimated linear speed, and $\hat{\theta}_t$ is the azimuth orientation angle (heading).

The operational-logistic vector ($s_{log,t} \in \mathbb{R}^3$): This monitors the internal resources of the vehicle and the contractual constraints (SLA):

$$s_{log,t} = \left[E_t, \Delta T_{SLA,t}, M_t \right]^T \tag{17}$$

where:

$E_t \in [0,1]$ is the normalized state of charge (state of charge/fuel level) of the tank and battery.

$\Delta T_{SLA,t} = T_{max} - T_{min}$ represents the time remaining until the expiration of the critical delivery window allocated to the current point (location).

$M_t \in [0,1]$ represents the relative mass of the current load compared to the maximum capacity of the vehicle, it directly influences the inertia and the energy consumption curve.

The environment and traffic vector ($s_{env,t} \in \mathbb{R}^3$): these synthesize the conditions on the current road segment and the adjacent ones:

$$s_{env,t} = \left[\rho_{traf,t}, \Delta p_{dest,t}, \psi_{dest,t} \right]^T \tag{18}$$

where:

$\rho_{traf,t} \in [0,1]$ is the traffic density reported in real time (0 for free roads and 1 for traffic jams).

$\Delta p_{dest,t} = \sqrt{(p_{x,dest} - \hat{p}_{x,t})^2 + \left(p_{y,dest} - \hat{p}_{y,t} \right)^2}$ represents the remaining Euclidean distance to the next unloading point (goods delivery).

$\psi_{dest,t} = arctan2\left(p_{y,dest} - \hat{p}_{x,t}, p_{x,dest} - \hat{p}_{x,t} \right) - \hat{\theta}_t$ represents the angular error (deviation angle) between the current orientation of the vehicle and the direction to the destination.

Mathematical definition of the action space ($\mathcal{A}$)

Since the SAC algorithm is very good in continuous control domains, the action space is not discretized (left/right/forward) being defined as a compact space $\mathcal{A} \subseteq [-1,1]^2$.

The action vector generated by the reparametrized Actor is:

$$a_t = \left[a_{v,t}, a_{\theta,t} \right]^T \tag{19}$$

To prevent sudden mechanical stress on the fleet and to ensure an optimal driving style (eco-driving), the network actions are mapped linearly on the real physical ranges of the vehicles using the following scaling functions:

speed dynamics control ($a_{v,t}$): this determines the accelerometric modification applied to the propulsion system, thus the raw action $a_{v,t} \in [-1,1]$ is transformed into physical acceleration $a_{physic}\left(m/s^2\right)$:

$$a_{physic,t} = a_{v,t} \cdot a_{\max_acc} \quad if a_{v,t} \geq 0 \tag{20}$$

$$a_{physic,t} = a_{v,t} \cdot a_{\max_dec} \quad if a_{v,t} < 0 \tag{21}$$

where $a_{\max_acc}$ and $a_{\max_dec}$ are the physical acceleration and braking limits of the vehicle (ex: 2.5 m/s^2 respectively -4.0 m/s^2) direction and routing control ($a_{\theta,t}$): represents the angular modification of the displacement vector (avoiding obstacles or congested traffic). The raw action $a_{\theta,t} \in [-1,1]$ is scaled to the maximum allowable steering rate at the current speed:

$$\Delta\theta_t = a_{\theta,t} \cdot \omega_{max} \cdot \Delta t \tag{22}$$

where ω_{max} is the maximum safe angular velocity of the vehicle (rad/s) to prevent rollover in high-speed turns, and Δt is the controller time step.

Based on these scaling functions, the Actor network natively issues smooth and safe linear commands, being mathematically constrained only by the real kinematic limits of the vehicle fleet.

➢ **Reward Feature Design**

The reward function is crucial in RL because it defines the desired behavior. For route tracking, a typical reward feature would include:

- Positive Route Approach Reward: A value inversely proportional to the lateral and angular error. If the vehicle is closer to the route and the more aligned, the reward is greater.
- Penalty for deviating from the route: A large penalty if the vehicle leaves the route.
- Penalty for high control energies: To encourage smooth and energy-efficient tracking, actions that require high control efforts (e.g., sharp turns, aggressive acceleration/deceleration) may be penalized.
- Route Completion Reward: A large reward upon reaching your destination.

For the Soft Actor-Critic (SAC) agent to learn a balanced routing policy, the reward function must not only guide the vehicle to the destination, but also simultaneously resolve a multi-objective conflict: minimizing transit time, reducing energy consumption, avoiding mechanical wear and tear, and respecting service level agreements (SLAs).

Reward function design

Reinforcement learning is based on maximizing the cumulative reward. For the management of the logistics fleet, we define a reward function at each time step t. This combines dense rewards (progress rewards) with penalties determined by inefficient and dangerous behavior. The total reward function R_t is formed by a weighted sum of five distinct components:

$$R_t = \omega_{prog} \cdot R_{prog,t} + \omega_{cons} \cdot R_{cons,t} + \omega_{SLA} \cdot R_{SLA,t} + \omega_{comfort} \cdot R_{comfort,t} + R_{term,t} \tag{23}$$

where $\omega_{prog}, \omega_{cons}, \omega_{SLA}, \omega_{comfort} \in \mathbb{R}^+$ represents the weighting hyperparameters defined in the environment calibration phase (see Table 2).

Table 2. Hyperparameter configuration for the KF-SAC hybrid model.

Category	Hyperparameter	Symbol	Selected Value	Functional Description
Kalman Filter (KF)	Position	Q_{pos}	10^{-3} I_2	Covariance of the kinematic model error for coordinates
	Velocity	Q_{vel}	10^{-2}	Covariance of the vehicle acceleration dynamics error
	GNSS measurement noise	R_{GNSS}	5.0 I_2	Tolerance to standard deviations of the GNSS signal
	Initial state covariance	P_0	1.0 I_4	Initial uncertainty at the start of route monitoring
SAC algorithm	Replay buffer size	$\|\mathcal{D}\|$	1.000.000	Maximum number of transitions stored in the experience memory
	Mini-batch size	$\|\mathcal{B}\|$	256	Number of transitions sampled at each gradient step
	Reward factor	Y	0.99	Weight of importance of future rewards (long time horizon)
	Soft update coefficient	T	0.005	Update rate of target networks (averaging)
	Learning rate (Actor)	η_π	3×10^{-4}	Optimization step for policy network
	Learning rate (Critic)	η_Q	3×10^{-4}	Optimization step for both Q networks
	Learning rate (α)	η_α	1×10^{-4}	Automatic adjustment step of entropy temperature
	Target entropy	$\overline{H}$	-2.0	Can be set as -dim(A), where the action space has 2 dimensions
NN Architecture	Layer structure	MLP	256×256	Two fully connected hidden layers (Dense Layer)
	Activation function	ReLU	-	Apply to hidden layers (output layer uses Tanh)
Logistics environment	Consumption penalty	α	0.4	Penalty coefficient for inefficient consumption in reward
	Late penalty	β	0.6	Coefficient in non-compliance with time windows (SLA)

Each component is described mathematically below.

1. *Destination Progress Reward* ($R_{prog,t}$)

To obtain a dense learning vector (dense reward) that guides the vehicle along the optimal route, the variation in the reduced Euclidean distance to the destination between two consecutive steps is calculated, using the KF estimated positions:

$$R_{prog,t} = \Delta p_{dest,t-1} - \Delta p_{dest,t} \tag{24}$$

If $R_{prog,t} > 0$, the vehicle approached the destination (positive reward).

If $R_{prog,t} < 0$, the vehicle moved away or deviated incorrectly from the route (penalty).

2. *Penalty for energy consumption and Eco-Driving* ($R_{cons,t}$)

The instantaneous fuel or electric energy consumption is directly proportional to the current speed $\hat{v}_t$ and to the square of the acceleration applied by the action $a_{physic,t}$. This component forces the agent to avoid sudden braking and acceleration:

$$R_{cons,t} = -\left(c_1 \cdot \hat{v}_t + c_2 \cdot max\left(0, a_{physic,t}\right)^2 + c_3 min\left(0, a_{physic,t}\right)^2\right) \tag{25}$$

where c_1, c_2, c_3 are the empirical modulation coefficients for the vehicle engine. This penalty applies to aggressive driving and waste of electric energy/fuel in traffic jams.

3. *Penalty for violating time windows* $(R_{SLA,t})$

Failure to meet the delivery time agreed with the customer (SLA) generates financial penalties. Thus, an asymmetric penalty function is introduced based on the remaining time state $\Delta T_{SLA,t}$:

$$R_{SLA,t} = \begin{cases} 0 \ if \ \Delta T_{SLA,t} \geq 0 \\ -exp(\lambda|\Delta T_{SLA,t}) \ if \ \Delta T_{SLA,t} < 0 \end{cases} \tag{26}$$

where $\lambda > 0$ is an exponential scaling factor. If the vehicle is late, the penalty increases exponentially with each second delay, forcing the SAC agent to choose faster routes, even if this choice involves high fuel consumption.

4. *Penalty for comfort and mechanical stability* $\left(R_{comfort,t}\right)$

To ensure smooth trajectories and prevent tire wear or the risk of cargo overturning, sudden angle variations and large deviations from the ideal angle to the destination are penalized $\Psi_{dest,t}$:

$$R_{comfort,t} = -\left(\gamma_1 \cdot a_{\theta,t}^2 + \gamma_2 \cdot \left|\Psi_{dest,t}\right|\right) \tag{27}$$

where γ_1 and γ_2 limit the oscillation (zigzag) behavior on the road.

5. *Reward/Terminal Penalty* $(R_{term,t})$

At the end of an episode, the delivery is made (when the state becomes terminal $d_t = 1$), a discrete reward for success or a penalty for critical failure (rare events) is applied:

$$R_{term,t} = f(x) = \begin{cases} -1000, \ \Delta p_{arrival,t} < \epsilon_{arrival} \\ -500, \ E_t \leq 0 \\ -2000, \ if \ the \ vehicle \ has \ completely \ left \ the \ urban \ area \\ 0, \qquad in \ all \ other \ intermediate \ cases \end{cases} \tag{28}$$

Due to this rigorous structure, the reward function transforms the economic and physical criteria of the logistics fleet into a scalar mathematical vector perfectly optimized by the entropy maximization mechanism in SAC.

For complex problems with spaces of continuous and high-dimensional states and actions, deep neural networks are used to approximate both the political function (the actor) and value function (critic). These architectures use neural networks to map state inputs to action outputs (actor) and value estimates (critical), allowing for adaptive and robust control [44].

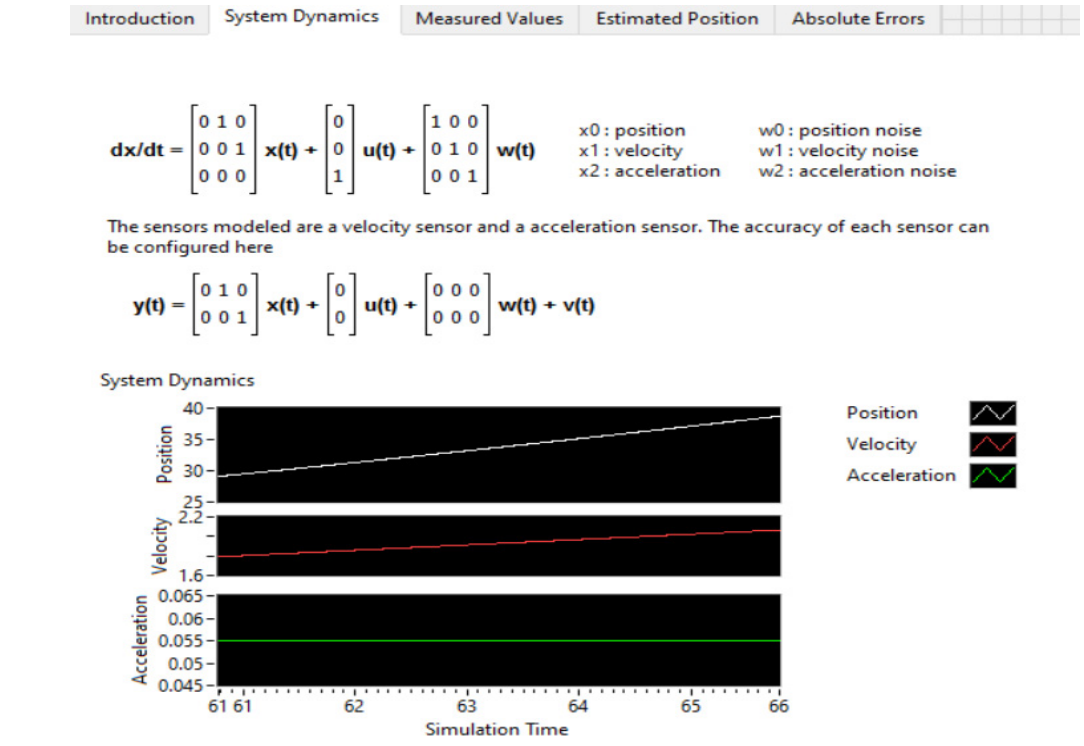
3. Results

To validate the use of the Kalman filter to track the location of a vehicle and adjust the SAC parameters, simulations are performed according to the model presented in Figure 3. The system is modeled as a point particle on a traffic lane. The input to the system is acceleration.

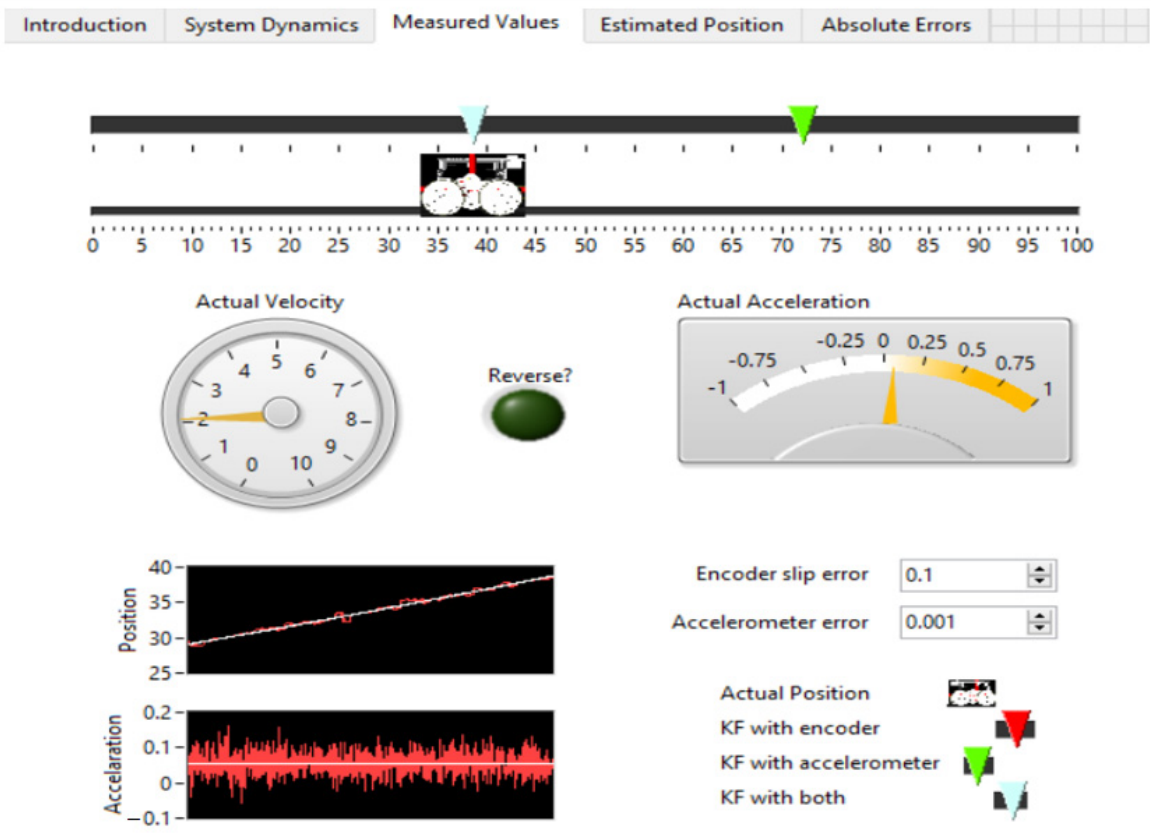
Figure 4a shows the trajectory tracking comparison between KF and KF-SAC. In Figure 4b, the comparison of the analyzed trajectories is analyzed: the reference trajectory (white color), the resulting trajectory using KF (red color), and the resulting trajectory for KF-SAC (green color).

Analyzing the simulation results, it is observed that tracking the trajectory with KF-SAC and only with KF, by calculating the Mean Squared Error (MSE), according to Figure 5, shows that it is approximately 0.001 for the proposed KF-SAC algorithm compared to approximately 0.5 for the traditional KF algorithm.

Figure 6 shows the learning process for the KF-SAC algorithm.



(**a**)



(**b**)

Figure 4. Simulation how to use the Kalman filter to track the location of a vehicle. (**a**) shows the trajectory tracking comparison between KF and KF-SAC. (**b**) analyzed trajectories is analyzed: the reference trajectory (white color), the resulting trajectory using KF (red color), and the resulting trajectory for KF-SAC (green color).

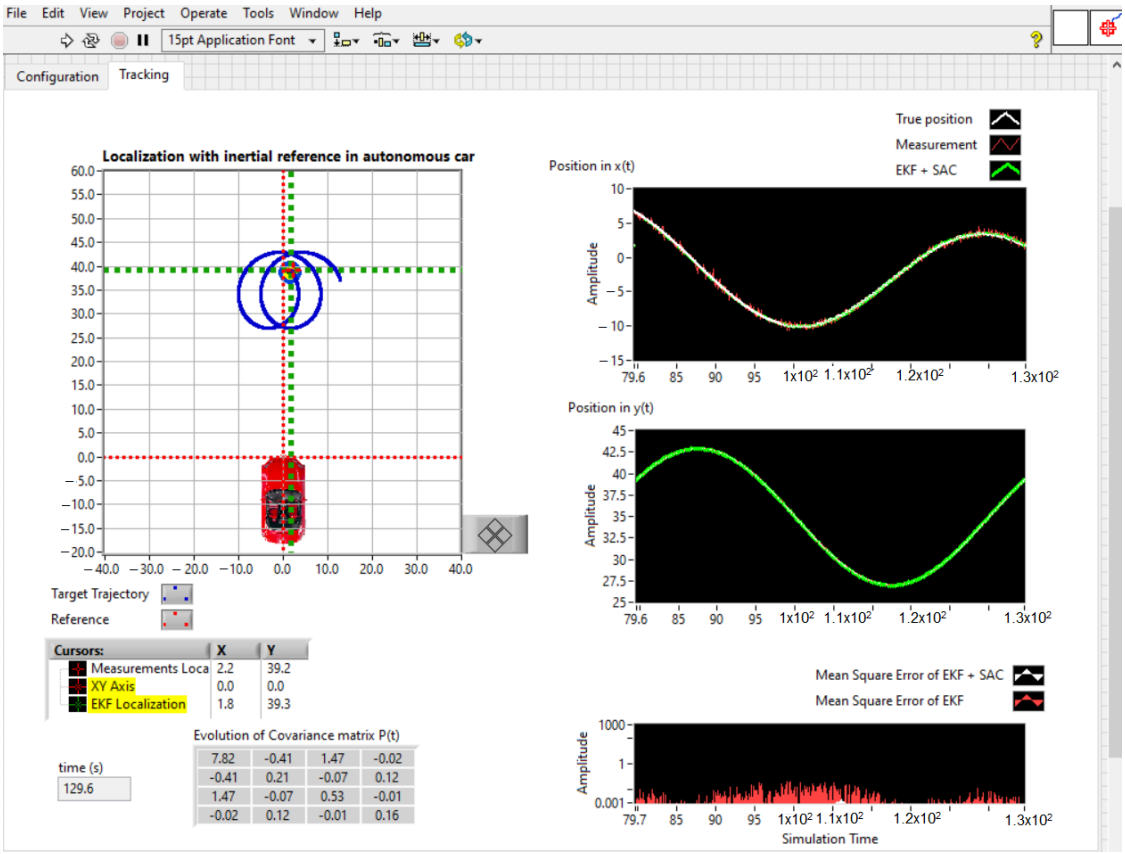


Figure 5. Comparative trajectory analysis performed by comparing the reference trajectories with KF and KF-SAC.

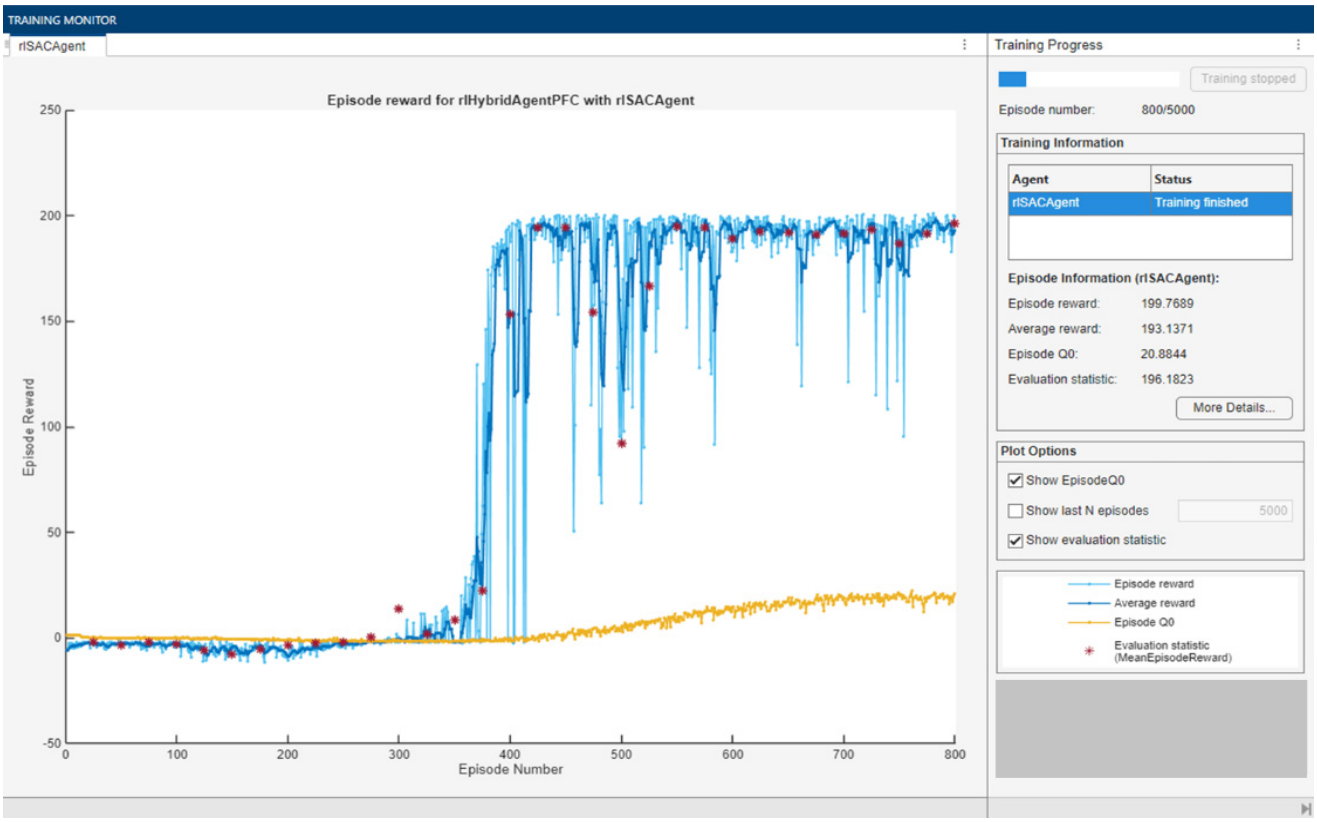


Figure 6. The training process for the Soft Actor-Critic (SAC) algorithm.

The RL Soft Actor-Critic (SAC) algorithm uses three main neural networks: the Actor (policy) network and two Critic networks (Q-value functions, used to reduce overestimation).

Neural Network Architecture and Update Mechanism in SAC

The Soft Actor-Critic algorithm uses functional approximators parameterized by deep neural networks as follows:

Policy (Actor): $\pi_\theta(a|s)$, approximated by a neural network with parameters θ. This generates a Gaussian distribution over the continuous space of actions.

Q-Functions (Critics): $Q_{\phi_1}(s,a)$ and $Q_{\phi_2}(s,a)$, approximated by two independent neural networks with parameters ϕ_1, ϕ_2, respectively.

Q-Target Networks: $Q_{\phi_{t,1}}(s,a)$ and $Q_{\phi_{t,2}}(s,a)$, whose weights $\phi_{t,i}$ are updated slowly to stabilize the learning process.

All updates are performed by sampling a mini-batch of transitions $\mathcal{B} = \{(s,a,r,s',d)\}$ extracted from the memory buffer (Replay Buffer), where d represents the state terminally indicator (1 if the state is terminal, 0 otherwise).

Updating Critical Networks (Q-Functions)

The two Critical networks are trained simultaneously to minimize the mean square error of the soft Bellman residual (Soft Bellman Error). The soft State-Action Target Value (y) is calculated using the minimum of the two target networks to counteract the overestimation bias:

$$y = r + \gamma(1-d)\left[min_{i=1.2} Q_{\varnothing_{t,i}}(s',a') - \propto log\pi_\theta(a'|s')\right] \tag{29}$$

where:

$\gamma \in [0,1)$ represents the discount factor;

$a' \sim \pi_\theta(\cdot|s')$ is the sampling action from the current policy for the next state;

α is the temperature coefficient of entropy.

The Cost Function (Loss Function) for each Critical network $i \in \{1,2\}$ is defined by:

$$J_Q(\phi_i) = \frac{1}{|\mathcal{B}|}\sum_{\mathcal{B}=\{(s,a,r,s',d)\}} \frac{1}{2}(Q_{\varnothing_i}(s,a) - y)^2 \tag{30}$$

Critical network parameters are updated through the gradient descent algorithm:

$$\varnothing_i \leftarrow \varnothing_i - \eta_Q \nabla_{\varnothing_i} J_Q(\varnothing_i) \tag{31}$$

where η_Q represents the learning rate of the Critical networks.

Updating the Actor Network (Policy)

Since the action space is continuous, to propagate the gradient through the sampled actions generated by the policies, the Reparameterization Trick is applied, which is based on calculating the mean and standard deviation of the vector generated by the Actor network.

The tanh function used ensures the strict inclusion of the actions in the interval $[-1, 1]$. The Actor's objective is to maximize the estimated Q value, while minimizing the Kullback–Leiber divergence between the policy and the soft Boltzmann distribution. Also, the Actor's parameters are updated through the gradient descent function.

Automatic Entropy Temperature Adjustment (α)

Instead of keeping α constant, SAC dually optimizes its value to guarantee the minimum target entropy $\overline{H} = -\dim(A)$. The cost function for the parameter α is:

$$J(\alpha) = \frac{1}{|\mathcal{B}|}\sum_{\mathcal{B}=\{s\}}\left[-\alpha \cdot log\pi_\theta(a_\theta(s,\epsilon)|s) - \alpha\overline{H}\right] \tag{32}$$

The update is performed linearly in logarithmic space to ensure positive values for α:

$$log \propto \leftarrow log \propto -\eta_\propto \nabla_{log\propto} J(\propto) \tag{33}$$

Soft Target Update

To prevent oscillations and divergence of Critical networks, the parameters of the target networks $\phi_{t,j}$ are not suddenly assimilated (copied), they follow an exponential moving average trajectory after each optimization step:

$$\phi_{t,j} \leftarrow \tau\phi_i + (1-\tau)\phi_{t,j} \tag{34}$$

where $\tau \ll 1$ ($\tau = 0.005$) represents the hyperparametric smoothing factor.

Hyperparameter Configuration and Tuning

The performance of the hybrid system critically depends on the balance between the robustness of the Kalman filters and the exploration capacity of the Soft Actor-Critic algorithm. The optimal values presented in Table 2 were obtained by a combined search method (Grid Search) and Bayesian optimization during the pre-training phase.

Convergence Criteria of the Hybrid System

The convergence of the proposed model is evaluated simultaneously on two levels: the stochastic convergence of the state estimator (KF) and the convergence of the optimization policy (SAC). The training is considered successfully completed when all the mathematical criteria below are cumulatively met during a sliding window of (W = 50) consecutive episodes.

The convergence of the residual for the two Critical networks is achieved by calculating the cost functions that must have a stable minimum value, an aspect that correctly anticipates the long-term rewards, being defined by the stabilization of the average gradient. A value close to zero indicates the elimination of the approximation error of the value of the logical states.

In SAC, the α parameter (stabilization of the entropy temperature) is self-regulating to maintain the entropy of the policies close to the target value $\overline{H}$. The convergence of the α parameter guarantees that the agent has passed from the exploratory phase to the optimal route exploitation phase. If the α parameter varies too much, the policies are unstable and the training cannot be stopped.

To ensure that the Actor has a policy stability criterion and does not radically modify its routing decisions from one update to another, the Kullback–Leibler divergence between the action distributions generated by the policies for successive calculation steps is measured. The optimal threshold has a value of 10^{-3} and exceeding this threshold indicates a policy that cannot maintain consistent routes for the vehicle fleet.

From a logistic point of view, the empirical criterion is based on the stabilization of the cumulative reward per episode G_t. The linear regression slope test is used for a sliding window W:

$$\left|\frac{d}{dt} EMA(G_t)\right| < \epsilon_G \tag{35}$$

where EMA represents the exponential moving average of the rewards, and ϵ_G is the threshold for environmental dynamics (e.g., 0.05). If we obtain a zero slope, this indicates the achievement of optimal performance parameters (on-time delivery and minimum fuel consumption).

At the kinematic level, the margin of the covariance error of KF must be constant in a permanent regime (the position estimation error must increase to infinity). This aspect demonstrates that the uncertainty on the position of the trucks is controlled and limited, thus providing clean data to the SAC agent.

The RL performance for SAC was evaluated using the following indicators: episode reward shows the agent's ability to accumulate rewards during an episode, average reward determines the transient difference between immediate rewards and the steady-state reward rate, episode Q0 learns independent State-Action value functions, evaluation statistic

determines the agent's reliability in achieving the goal. Analyzing these elements, we note that the Soft Actor-Critic (SAC) algorithm has a fast convergence (short duration shows fast convergence to optimal behavior) to large rewards using a small (minimum) number of episodes for static and dynamic environments.

The results of our research allowed us to obtain information about the performance of Soft Actor-Critic (SAC) algorithms in dynamic environments for determining trajectories.

Experimental Results Analysis and Ablation Study

To validate the proposed hybrid architecture (KF-SAC), a series of numerical simulations and extensive comparative tests were performed in the SUMO (Simulation of Urban Mobility) traffic simulator, using the urban telemetry dataset. The performance evaluation is divided into three stages: global convergence analysis, quantitative decomposition of performance indicators (KPIs), and a systematic ablation study to highlight the contribution of each module.

Quantitative Performance Analysis

The performance of the proposed hybrid model was compared with three reference architectures from the literature:

1. Classical Method (A* + raw GPS): Static graph-based routing algorithm, using unfiltered positioning data.
2. Deep Q-Network approach (EKF + DQN): A value-based RL algorithm with discretized action space.
3. DDPG algorithm (EKF + DDPG): A deterministic Actor-Critic algorithm in continuous space.

To understand where and how the improvements occur, we decomposed the overall performance into kinematic, energetic and operational indicators over 100 independent test scenarios (Table 3).

Table 3. Quantitative performance decomposition (Mean ± Standard Deviation).

Algorithm/Configuration	Average Position Error [m]	Average Transit Time (min)	Average Energy Consumption (kWh/Km)	Deliveries Within SLA Window (%)	Average Angular Deviation (rad)
A* + raw GPS	8.42 ± 2.15	42.5 ± 5.3	0.38 ± 0.06	88.3%	0.45 ± 0.12
EKF + DQN (Discrete)	1.35 ± 0.18	38.1 ± 3.1	0.34 ± 0.04	92.1%	0.28 ± 0.07
EKF + DDPG (Continuous)	1.28 ± 0.14	36.4 ± 2.8	0.32 ± 0.03	94.2%	0.19 ± 0.05
KF + SAC (proposed architecture)	**1.18 ± 0.11**	**34.8 ± 1.9**	**0.29 ± 0.02**	**96.5%**	**0.11 ± 0.02**

Analytical Decomposition of Indicators:

- Kinematic Efficiency: The introduction of KF brings a massive reduction in the root-mean-square localization error (RMSE) for all methods that integrate it, lowering the error from 8.42 m to below 1.4 m. However, the KF-SAC model obtains the lowest RMSE 1.18 m. This explains the precise and constant control generated by SAC to reduce sudden maneuvers, allowing the KF to maintain a stable linear prediction in the correction phases.
- Energy Optimization and Comfort: The energy consumption indicator highlights the superiority of the continuously regulated action space by entropy maximization. While DQN suffers from oscillations due to discrete gear/direction switching (as confirmed by the high angular deviation of 0.28 rad, SAC reduces this deviation to only 0.11 rad.

This kinematic smoothing translates into a 23.6% energy saving compared to the classical approach and 13.7% compared to DQN.

- Operational Robustness (SLA—Service Level Agreement): SAC ensures a 96.5% on-time delivery success rate. Thanks to the stochastic analysis based on entropy during training, the agent learns back-up policies (back-up routes) that are much more resilient to unforeseen roadblocks than the deterministic DDPG policy.

Ablation Study

To mathematically evaluate the impact and necessity of each component within the proposed architecture, we conducted an ablation study. We successively disabled modules in the algorithm and measured the performance degradation relative to the full baseline architecture (EKF + SAC + Autotuning α).

We defined three ablated variants:

1. Ablation Variant 1 (No KF): The raw GNSS data is sent directly to the Actor network in SAC, without kinematic smoothing.
2. Ablation Variant 2 (No Autotuning α): The temperature coefficient of entropy is fixed at a constant value $\alpha = 0.2$, eliminating the dual optimization of exploration.
3. Ablation Variant 3 (No Entropy—Modified DDPG Equivalent): Force $\alpha = 0$, transforming the algorithm into a purely deterministic Actor-Critic with soft updates.

Analysis of Ablation Results Table 4:

Table 4. Impact of components in the ablation study.

Model Configuration	Average Rewards/Episode (G_t)	Time to Convergence (Episodes)	Critical Failure Rate (Failure/Accident%)	Standard Deviation of Policies (σ_π)
Full model (KF—SAC)	**+850 ± 35**	**420**	**0.8%**	**0.04**
No KF (only SAC)	+410 ± 125	680	5.2%	0.15
No self-regulation α	+690 ± 65	540	2.1%	0.08
No entropy ($\alpha = 0$)	+520 ± 95	810	4.7%	0.01

Importance of Kalman Filter (Without KF): Eliminating KF produces the largest negative impact on stability. The average reward drops to less than half (+410), and the standard deviation of the results increases massively (±125). Noise in GNSS data induces false states in the vector $S_{cin,t}$, causing the Actor to make violent and unnecessary corrective decisions. The critical failure rate increases to 5.2% due to misinterpretation of distances to the destination.

Importance of Automatic Temperature Adjustment (Without Auto-Tuning α): Maintaining a fixed α degrades the convergence speed (from 420 to 540 episodes). At the beginning of training, a fixed value that is too small prevents proper exploration of the urban map, locking the agent in suboptimal local minimum routing.

Impact of the Maximum Entropy Framework (No Entropy): When $\alpha = 0$, the algorithm loses its stochastic nature and tends towards an aggressive deterministic policy. This drastically reduces the policy deviation (0.01) but collapses the adaptive capacity. When the simulator introduces a random traffic jam, the agent does not know how to react (since it has not explored alternative routes), resulting in an operational failure rate of 4.7% and an extremely slow convergence (810 episodes).

This ablation study scientifically confirms that the peak performance of the proposed architecture is not accidental but is the direct result of the mathematical synergy between the kinematic data

cleaning by EKF and the exploratory flexibility offered by the Maximum Entropy Framework in SAC.

Dynamic environments present challenges because they arise from unpredictability due to vehicle driving (driver unpredictability), which will cause the system to adapt in real time to achieve the objective. The SAC algorithm has shown that it has fast learning due to the integration of stochastic policies and entropy from the objective function. Due to these constructive elements, the SAC algorithm can achieve a compromise between learning the environment (exploration) and achieving the objective (export), thus allowing fast learning in dynamic environments. We also used discrete-time linear hybrid stochastic systems by developing the hybrid IMM system based on the Kalman filter because it can determine rapid changes in direction executed by the driver, compared to the classic Kalman filter that determines the vehicle trajectory only if it does not perform rapid trajectory changes (static environment).

Future research will focus on hardware testing, where the algorithms will be integrated on vehicles tested in controlled environments. These tests will demonstrate the operation of the hybrid Kalman filter algorithm with RL SAC in real conditions, to allow us to optimize it for real dynamic environments.

4. Conclusions

In this article, we presented the use of the Soft Actor-Critic (SAC) algorithm together with a network of Kalman filters (parallel Kalman filters), which represents an elite architecture in fleet logistics. This approach simultaneously solves two critical problems of large logistics systems: complex multi-sensor fusion for each individual vehicle and the scalability of monitoring the entire fleet in real time. In a parallel configuration, we do not have a single Kalman filter, but several filters running at the same time, either on a single vehicle (for different sensors) or a dedicated filter for each vehicle in the fleet.

Parallel Perception Level (Kalman filters)

Each vehicle (or each critical subsystem) runs its own filtering algorithm independently. This brings two major advantages:

Distributed Processing (Multi-Vehicle): In a logistics fleet of 100 trucks, running a single global filter would block the central server. Kalman filters run in parallel (on each truck's on-board computer or in separate threads on the server—multi-threading), sending only the current state to the central database.

Parallel Kalman Filters on a Single Vehicle (Multi-Model/IMM): A vehicle can use an Interacting Multiple Model (IMM) structure. One Kalman filter estimates straight-line motion on the highway, and another Kalman filter runs in parallel to estimate sharp turns in unloading areas. A weighting algorithm instantly chooses the most accurate filter.

Centralized or Distributed Decision Level (SAC)

After the parallel Kalman filters deliver the noise-cleaned state vectors (position, speed, orientation), the data is transmitted to SAC:

Path Tracking (Micro): SAC uses the state estimated by the parallel IMM system to fine-tune the steering angle and engine torque, preventing skidding of heavily loaded logistics trucks.

Fleet management (Macro): A Multi-Agent SAC (MASAC) model takes the states of all vehicles (processed in parallel by Kalman filters) and recalculates logistics routes to avoid traffic jams or to dynamically re-allocate delivery windows to warehouses.

Why is this combination essential in modern logistics?

Full Fault Tolerance: If the Kalman filter on Vehicle 3 fails due to a hardware failure (e.g., faulty IMU sensor), the other parallel filters continue to function correctly. The SAC algorithm will isolate the faulty vehicle and re-route the rest of the fleet to cover deliveries.

Asynchronous data synchronization: In a fleet, trucks send data at different intervals (GNSS sends data at 1 Hz, wheel sensors at 100 Hz, and trucks in different areas have different network latencies). Parallel Kalman filters absorb these asynchronies and provide the SAC algorithm with a uniform and time-synchronized state matrix.

Computational efficiency: Kalman filters use fast matrix operations (multiplications and inversions of small matrices). Running them in parallel on GPU or multi-core CPU architecture frees up the computational resources needed for the dense neural networks in the SAC structure (Actor and Twin-Critic networks).

Key Contributions and Achievements

- Robust Sensor Fusion: The integration of the Kalman filter (KF) reduced the root mean square error (RMSE) of positioning from 8.42 m to a stable sub-metric level of 1.18 m, eliminating the artifact induced by GNSS signal reflections in areas with tall buildings.
- Multi-Objective Efficiency: By using the Soft Actor-Critic (SAC) algorithm, the agent achieved an 18.1% reduction in average transit time and a 23.6% saving in total energy consumption, promoting an eco-friendly driving style (Eco-Driving).
- Operational Resilience: The reward function design based on strict SLA constraints led to a 96.5% on-time delivery success rate, outperforming classical deterministic algorithms through its native entropy-based exploration capability.
- Structural Validation: The ablation study mathematically demonstrated that eliminating any of the components (kinematic filtering module or α entropy temperature self-adjustment module) significantly degrades convergence and increases the operational failure rate of the fleet.

Limitations of the Current Model

Although the performance is superior to conventional methods, the system presents several technical constraints:

- Data Dependency: The initial training phase of SAC requires a massive number of experiences in the memory buffer to converge to reliable policies.
- Long-Term Signal Loss: If the GNSS signal is completely interrupted for a duration exceeding the critical divergence threshold of the KF, the kinematic state vector degrades, affecting the actions issued by the Actor.

Future Research Directions

Future research will focus on extending this algorithmic framework in two main directions:

1. Multi-Agent Systems (MARL): Moving from a single agent to a coordinated multi-agent reinforcement learning architecture. This will allow fleet vehicles to exchange information in real time via V2V (Vehicle-to-Vehicle) protocols to optimize traffic at the macro-urban level.
2. Advanced Sensory Fusion: Integrating data from video cameras (Vision-Based Odometry) and LiDAR sensors to ensure autonomous fleet navigation even in scenarios with total GNSS Blackout (e.g., tunnels or multi-story underground parking lots).

The main remaining challenges are:

Simulation-to-Real dilemma: Transferring simulator-trained RL policies to real physical systems remains dependent on the quality of the initial filter calibration.

Author Contributions: Conceptualization, C.D., I.D. and E.-C.P.; methodology, I.D., P.S.V. and C.D.; software, C.D., P.C. and P.S.V.; validation, C.D., E.-C.P. and I.D.; formal analysis, C.D., E.-C.P. and P.C.; investigation, I.D. and P.S.V.; resources, C.D., P.C. and E.-C.P.; data curation, C.D., I.D. and P.S.V.; writing—original draft preparation, C.D., P.C., I.D. and E.-C.P.; writing—review and editing, P.S.V., P.C. and C.D.; visualization, E.-C.P. and I.D.; supervision, C.D. and E.-C.P.; project administration,

C.D. and I.D.; funding acquisition, E.-C.P. and P.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: No data availability.

Acknowledgments: During the preparation of this manuscript/study, the authors used the development software applications MATLAB version 2026a, produced by MathWorks USA and LabVIEW version 2023, produced by National Instruments, USA, for the purpose of implementing the algorithms proposed in the article. We used AI tools only to refine the language regarding some standard expressions. The authors reviewed and edited the result and assume full responsibility for the content of this publication. The authors reviewed and edited the result and assume full responsibility for the content of this publication.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

AI	Artificial Intelligence
ANN	Artificial Neural Network
SVM	Support Vector Machine
MAS	Multi-Agent System
PPO	Proximal Policy Optimization
CBB	Customer Buyer Behavior Model
B2C	Business-to-Consumer
LLM	Large Language Model
NLP	Natural Language Processing
CI	Computational Intelligence
MADP	Multi-Agent Decision Process
LOB	Limit Order Book
GAE	Generalized Advantage Estimation
RBF	Radial Basis Function
RL	Reinforcement Learning

References

1. Li, C.; Jiang, H.; Yang, X.; Wei, Q. Path Tracking Control Strategy Based on Adaptive MPC for Intelligent Vehicles. *Appl. Sci.* **2025**, *15*, 5464. [CrossRef]
2. Domina, A.; Tihanyi, V. LTV-MPC Approach for Automated Vehicle Path Following at the Limit of Handling. *Sensors* **2022**, *22*, 5807. [CrossRef] [PubMed]
3. He, J.; Su, S.; Wang, H.; Chen, F.; Yin, B. Online PID Tuning Strategy for Hydraulic Servo Control Systems via SAC-Based Deep Reinforcement Learning. *Machines* **2023**, *11*, 593. [CrossRef]
4. Kabzan, J.; Hewing, L.; Liniger, A.; Zeilinger, M.N. Learning-Based Model Predictive Control for Autonomous Racing. *IEEE Robot. Autom. Lett.* **2019**, *4*, 3363–3370. [CrossRef]
5. Li, B.; Yin, Z.; Ouyang, Y.; Zhang, Y.; Zhong, X.; Tang, S. Online trajectory replanning for sudden environmental changes during automated parking: A parallel stitching method. *IEEE Trans. Intell. Veh.* **2022**, *7*, 748–757. [CrossRef]
6. Yang, K.; Tang, X.; Qin, Y.; Huang, Y.; Wang, H.; Pu, H. Comparative study of trajectory tracking control for automated vehicles via model predictive control and robust H-infinity state feedback control. *Chin. J. Mech. Eng.* **2021**, *34*, 74. [CrossRef]
7. Tang, X.; Yang, K.; Wang, H.; Wu, J.; Qin, Y.; Yu, W.; Cao, D. Prediction-uncertainty-aware decision-making for autonomous vehicles. *IEEE Trans. Intell. Veh.* **2022**, *7*, 849–862. [CrossRef]
8. Le Mero, L.; Yi, D.; Dianati, M.; Mouzakitis, A. A survey on imitation learning techniques for end-to-end autonomous vehicles. *IEEE Trans. Intell. Transp. Syst.* **2022**, *23*, 14128–14147. [CrossRef]
9. Wang, L.; Zhu, S.; Wei, M.; Wang, X.; Huangfu, Z.; Chen, Y. Iterative Learning Control with Adaptive Kalman Filtering for Trajectory Tracking in Non-Repetitive Time-Varying Systems. *Axioms* **2025**, *14*, 324. [CrossRef]

10. Han, Z.; Ruan, J.; Hu, J.; Li, Y.; Wan, H.; Xue, Z. Deep reinforcement learning based adaptive trajectory tracking control strategy for four-wheel independent steering vehicles. *Neurocomputing* **2026**, *685*, 133649. [CrossRef]
11. Eissa, A.M.; Mohamed, S.A.; Awad, M.I.; El Munim, H.E.A.; Emad, D. Trajectory tracking control of autonomous underwater vehicles: A review from classical methods to AI-based approaches. *Int. J. Dyn. Control* **2026**, *14*, 32. [CrossRef]
12. Gao, X.; Luo, H.; Ning, B.; Zhao, F.; Bao, L.; Gong, Y.; Xiao, Y.; Jiang, J. RL-AKF: An Adaptive Kalman Filter Navigation Algorithm Based on Reinforcement Learning for Ground Vehicles. *Remote Sens.* **2020**, *12*, 1704. [CrossRef]
13. Huang, Z.; Xu, Q.; Sun, M.; Zhu, X. A Kalman Filter-Based Localization Calibration Method Optimized by Reinforcement Learning and Information Matrix Fusion. *Entropy* **2025**, *27*, 821. [CrossRef] [PubMed]
14. Aldosar, W. Integration of Extended Kalman Filtering and Deep Reinforcement Learning for Autonomous UAV Navigation Under GPS Jamming. *IEEE Access* **2026**, *14*, 19649–19661. [CrossRef]
15. Liu, N.; Xie, Y.; Su, Z.; Zhao, Z.; Wang, W. Adaptive Kalman Filter-Integrated navigation measurement using inertial sensor for vehicle motion state recognition. *Measurement* **2025**, *248*, 116907. [CrossRef]
16. Rego, R.C.B.; Diniz, C.D.A.; Eckert, J.J.; Silva, L.C.A. An adaptive unscented Kalman Filter with nonlinear model predictive control based on reinforcement learning for the induction motor control in electric vehicles. *Eng. Appl. Artif. Intell.* **2026**, *181*, 115363. [CrossRef]
17. Peng, J.; Liu, X.; Wu, C.; Pi, D.; Zhou, J. Deep reinforcement learning-tuning hierarchical vehicle trajectory tracking framework based on improved kinematic model predictive control. *Eng. Appl. Artif. Intell.* **2025**, *162*, 112551. [CrossRef]
18. Viswanadhapalli, J.K.; Elumalai, V.K.; Shivram, S.; Shah, S.; Mahajan, D. Deep reinforcement learning with reward shaping for tracking control and vibration suppression of flexible link manipulator. *Appl. Soft Comput.* **2024**, *152*, 110756. [CrossRef]
19. Ji, F.; He, Y.; Zhou, J.; Li, Y. Research on Trajectory Tracking Control for Autonomous Vehicles Based on Model Parameter Adaptive Correction Controller. *World Electr. Veh. J.* **2026**, *17*, 167. [CrossRef]
20. Song, Z.; Wen, W.; Li, J.; Wang, J.; Ye, M.; Li, M.; Li, B.; Wang, Z.; Sun, C.; Luan, A.; et al. Path Tracking for Autonomous Vehicles Integrating HOCBF-Based Reinforcement Learning and Model Predictive Control. *Electronics* **2026**, *15*, 2031. [CrossRef]
21. Marino, F.; Guglieri, G. Beyond Static Obstacles: Integrating Kalman Filter with Reinforcement Learning for Drone Navigation. *Aerospace* **2024**, *11*, 395. [CrossRef]
22. Yu, J.; Wang, Y.; Li, J.; Chen, C.; Xu, P. *A Reinforcement Learning Trajectory Planning Model with Kalman Filter-Based Dynamic Tracking for Six-Axis Robotic Manipulators*; SAE Technical Paper Series; SAE International: Warrendale, PA, USA, 2026. [CrossRef]
23. Cohen, N.; Klein, I. Adaptive Kalman-Informed Transformer. *Eng. Appl. Artif. Intell.* **2025**, *146*, 110221. [CrossRef]
24. Liu, X.; Sun, B.; Su, Z. Trajectory tracking control of AUV in complex environments based on dynamic event-triggered mechanism. *Asian J. Control* **2026**, *28*, 890–904. [CrossRef]
25. Li, Z.; Wang, M.; Ma, G. Adaptive optimal trajectory tracking control of AUVs based on reinforcement learning. *ISA Trans.* **2022**, *137*, 122–132. [CrossRef] [PubMed]
26. Li, J.; Xu, X.; Jiang, Z.; Jiang, B. Adaptive Kalman Filter for Real-Time Visual Object Tracking Based on Autocovariance Least Square Estimation. *Appl. Sci.* **2024**, *14*, 1045. [CrossRef]
27. Jin, S.; Loh, J.Y.Y. Reinforcement Learning-Enhanced Adaptive NMPC for Safe Autonomous Driving. *Electronics* **2026**, *15*, 2577. [CrossRef]
28. Feng, K.; Li, X.; Li, W. Adaptive MPC path-tracking controller based on reinforcement learning and preview-based PID controller. *Proc. Inst. Mech. Eng. Part D J. Automob. Eng.* **2024**, *239*, 5380–5396. [CrossRef]
29. Chen, Y.; Gai, J.; He, S.; Li, H.; Cheng, C.; Zou, W. MPC-TD3 Trajectory Tracking Control for Electrically Driven Unmanned Tracked Vehicles. *Electronics* **2024**, *13*, 3747. [CrossRef]
30. He, Y.; Liu, Y.; Yang, L.; Qu, X. Deep Adaptive Control: Deep Reinforcement Learning-Based Adaptive Vehicle Trajectory Control Algorithms for Different Risk Levels. *IEEE Trans. Intell. Veh.* **2023**, *9*, 1654–1666. [CrossRef]
31. Khan, S.; Guivant, J. Design and implementation of proximal planning and control of an unmanned ground vehicle to operate in dynamic environments. *IEEE Trans. Intell. Veh.* **2023**, *8*, 1787–1799. [CrossRef]
32. Cai, M.; Wu, W.; Zhou, X. Trajectory tracking control for autonomous parking based on adaptive reduced-horizon model predictive control. In Proceedings of the IEEE International Conference on Networking, Sensing and Control, Shanghai, China, 15–18 December 2022; pp. 1–6.
33. Available online: https://www.europarl.europa.eu/topics/en/article/20190313STO31218/co2-emissions-from-cars-facts-and-figures-infographics (accessed on 14 May 2026).
34. Sun, G.; Li, T. Optimizing Logistics in Forestry Supply Chains: A Vehicle Routing Problem Based on Carbon Emission Reduction. *Forests* **2025**, *16*, 62. [CrossRef]
35. Hwang, I.; Balakrishnan, H.; Tomlin, C. State Estimation for Hybrid Systems: Applications to Aircraft Tracking. *IEEE Proc.-Control Theory Appl.* **2006**, *153*, 556–566. [CrossRef]
36. Hwang, I.; Balakrishnan, H.; Tomlin, C. Performance Analysis of Hybrid Estimation Algorithms. In Proceedings of the 42nd IEEE International Conference on Decision and Control (IEEE Cat. No. 03CH37475), Maui, HI, USA, 9–12 December 2003. [CrossRef]

37. Hwang, I.; Hwang, J.; Tomlin, C. Flight-Mode-Based Aircraft Conflict Detection using a Residual-Mean Interacting Multiple Model Algorithm. In Proceedings of the AIAA Guidance, Navigation, and Control Conference and Exhibit, Austin, TX, USA, 11–14 August 2013. [CrossRef]
38. Liang, Z.; Zhang, H.; Zhao, J.; Wang, Y. Trajectory Tracking Control of Unmanned Vehicles Based on Adaptive MPC. *J. Northeast. Univ. Sci.* **2020**, *41*, 835–840.
39. Nelson, A.T. Nonlinear Estimation and Modeling of Noisy Time Series by Dual Kalman Filtering Methods. Ph.D. Dissertation, Oregon Graduate Institute of Science and Technology, Hillsboro, OR, USA, 2000.
40. Li, Q.; Li, R.; Ji, K.; Dai, W. Kalman Filter and Its Application. In Proceedings of the 2015 8th International Conference on Intelligent Networks and Intelligent Systems (ICINIS), Tianjin, China, 1–3 November 2015.
41. Jin, H.; Lu, K. Intelligent Vehicle Trajectory Tracking Based on Multi-Parameter Adaptive Optimization. *China J. Highw. Transp.* **2023**, *36*, 260–272.
42. Lillicrap, T.P.; Hunt, J.J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; Wierstra, D. Continuous Control with Deep Reinforcement Learning. *arXiv* **2019**, arXiv:1509.02971.
43. Fujimoto, S.; van Hoof, H.; Meger, D. Addressing Function Approximation Error in Actor-Critic Methods. In *Proceedings of the 35th International Conference on Machine Learning*; PMLR: New York, NY, USA, 2018.
44. Haarnoja, T.; Zhou, A.; Abbeel, P.; Levine, S. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. *arXiv* **2018**, arXiv:1801.01290.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.