Received: 22 February 2022 | Revised: 21 May 2022 | Accepted: 31 May 2022

DOI: 10.1111/exsy.13076

ORIGINAL ARTICLE

Expert Systems WILEY

Combining reinforcement learning and conventional control to improve automatic guided vehicles tracking of complex trajectories

J. Enrique Sierra-Garcia[1] | Matilde Santos[2]

[1]Electromechanical Engineering Department, University of Burgos, Burgos

[2]Institute of Knowledge Technology, Complutense University of Madrid, Madrid, Spain

Correspondence
J. Enrique Sierra-Garcia, Electromechanical Engineering Department, University of Burgos, 09006, Burgos, Spain.
Email: jesierra@ubu.es

Abstract

With the rapid growth of logistics transportation in the framework of Industry 4.0, automated guided vehicle (AGV) technologies have developed speedily. These systems present two coupled control problems: the control of the longitudinal velocity, essential to ensure the application requirements such as throughput and tag time, and the trajectory tracking control, necessary to ensure the proper accuracy in loading and unloading manoeuvres. When the paths are very short or have abrupt changes, the kinematic constraints play a restrictive role, and the tracking control becomes more challenging. In this case, advanced control strategies such as those based on intelligent techniques, including machine learning (ML) can be useful. Hence, in this work, we present an intelligent hybrid control scheme that combines reinforcement learning-based control (RLC) with conventional PI regulators to face both control problems simultaneously. On the one hand, PIs are used to control the speed of each wheel. On the other hand, the input reference of these regulators is calculated by the RLC in order to reduce the guiding error of the path tracking and to maintain the longitudinal speed. The latter is compared with a PID path following controller. The PID regulators have been tuned by genetic algorithms. The RLC allows the vehicle to learn how to improve the trajectory tracking in an adaptive way and thus, the AGV can face disturbances or unknown physical system parameters that may change due to friction and degradation of AGV mechanical components. Extensive simulation experiments of the proposed intelligent control strategy on a hybrid tricycle and differential AGV model, that considers the kinematics and the dynamics of the vehicle, prove the efficiency of the approach when following different demanding trajectories. The performance of the RL tracking controller in comparison with the optimized PID gives errors around 70% smaller, and the average maximum error is also 48% lower.

KEYWORDS

automated guided vehicle (AGV), intelligent control, machine learning (ML), path following, PID, reinforcement learning (RL)

1 | INTRODUCTION

Unmanned transport vehicles, also known as automatic guided vehicles (AGV), are commonly used in the industry to replace manned industrial trucks and conveyors. They have proved to be an effective element in production workplaces, reducing logistical errors and operating expenses (Espinosa et al., 2021). With the rapid growth of logistic transportation in the Industry 4.0 framework, AGV technologies have developed speedily (Nakimuli et al., 2021).

There are different types of AGVs depending on the degree of freedom they have to navigate in the workspace, namely free navigation AGVs and path-follower AGVs. Free navigation AGVs receive the destination point and they move freely in the workspace to get there, avoiding obstacles if any. On the other hand, predefined-path AGVs must follow a reference path to reach the destination and cannot leave it. This path can be defined with physical marks such as a magnetic tape, a painted line, a buried wire,… or by virtual paths within the navigation system (Zamora-Cadenas et al., 2021). Free navigation AGVs normally describe a polyline path, turn on the spot and move straight ahead to reach the destination. This freedom of movement is very useful for service robotic applications such as in offices, museum guides, and so on. However, it can be counterproductive in industrial applications where it is often necessary to control a fleet of several AGVs, there may be people around, and where it is also necessary to meet a tag time due to the production line constraints. In these cases, when a more deterministic performance is required, it is more convenient to use pre-defined paths for AGVs.

In many industrial sectors, where these predefined-path AGVs are used such as in the automotive field, AGVs need to perform accurate manoeuvres, to pick up trolleys, to get under racks, to pick up pallets, to stop at charging stations, …. In these cases, a small error of just 10 cm may not be tolerable, and it may cause the production to stop. Moreover, it is noteworthy to point that, when we work with large AGVs or mobile large containers, a small lateral error, especially in curves, may become a much larger error at the edge of the AGV or the container.

In this industrial framework, two different coupled control problems can be identified for AGVs. On the one hand, it is necessary to control the longitudinal velocity, so the vehicle follows the speed profile required by the application. This allows the AGV to meet some requirements, such as a specific tag time, that is set to optimize the logistic flow, and also to get the vehicle arriving at the stations within the defined accuracy range. On the other hand, the guiding error must be minimized to prevent the vehicle from running through prohibited areas. To do so, these AGVs typically use a proportional-integral-derivative (PID) controller to adjust the angular velocity of the whole AGV or of the traction unit, depending on the type of AGV. It is well known that PID controllers work well with linear systems, but they may not give such good performance for more complex problems. Indeed, in these industrial scenarios, these conventional regulators work well when the trajectories are large and smooth. However, when the path is short or has abrupt changes, the kinematic constraints have a strong influence on the performance of the system, hence the control becomes more challenging. In addition, these systems may be subjected to disturbances or unknown physical parameters that may vary due to friction and degradation of AGV mechanical components. In these cases, the use of more advanced techniques such as those based on artificial intelligence (AI) and machine learning (ML) can be useful, as proved in many control engineering complex systems (Garcia-Auñón et al., 2017; Martín et al., 2016; Sierra-García & Santos, 2021a).

The hybridization of control techniques is very often the only solution to coupled control problems (Menoyo Larrazabal & Santos Peñas, 2016; Sierra-García & Santos, 2021b). This approach allows to split the control goals and to use different techniques to address each part. Following this perspective, in this work, we develop a hybrid control scheme that exploits reinforcement learning (RL) to reduce the trajectory tracking guiding error of a hybrid AGV, and that combines it with conventional PI regulators that control the speed of each wheel.

The main contributions of this work can be summarized as follows:

- Proposal of a hybrid control architecture that combines RL and PI regulators for tracking and speed control of AGVs
- Validation of the proposal in simulation using the mathematical model of a hybrid AGV (Sánchez et al., 2022). This vehicle, quite common in industry, has a complex dynamics because it is made up of a differential mobile robot and a tricycle one. The whole system is simulated considering the kinematic and dynamic physical equations.
- Definition of different demanding trajectories to test the proposal: an ellipse, a lemniscate, and two different closed polylines with abrupt changes.
- Testing the proposal considering changes in the friction coefficients.

Considering the inverse kinematics of the traction unit, the output of the RL controller (RLC) is used to generate the appropriate reference speed for each PI while ensuring the desired longitudinal velocity. Thus, the PI regulators make the AGV follow the longitudinal speed profile meanwhile the RL minimizes the tracking error. The speed control shows a linear behaviour, hence the use of PID regulators to control it. However, the strong non-linearity of some trajectories affects mostly the tracking control and is clearly present in the guiding error, thus an RL controller is used for this control task. Several key performance indicators (KPI) related to the guiding error show that this hybrid controller provides a better performance than the PID. In addition, it is shown that the RLC is much more robust against changes in the system friction coefficients.

The rest of this article is organized as follows. In Section 2, the mathematical model of the AGV is described. The RL-based controller is explained in Section 3. Simulations results of the AGV moving in a working scenario are discussed in Section 4. The manuscript ends with the conclusions and future works.

2 | RELATED WORKS

AGV control can be studied at different levels. From lower to higher, we can identify path following, path planning and fleet management. Path following is focused on the control of the AGV to minimize the guiding error once the path has been set. This control problem is mainly addressed in predefined-path AGVs. Path planning consists in finding the best trajectories to go from the current position to the destination, avoiding obstacles. Finally, fleet management assigns routes to multiple AGVs, schedule the resources, and control the traffic at the intersections.

Path planning is a key research topic in autonomous vehicles field. It is difficult to plan an optimal path from a starting position to a target point due to the complex environment that usually surrounds these types of unmanned vehicles. Different techniques have been applied to deal with this problem, including those from the artificial intelligence and machine learning fields. In Liao et al. (2020), the SARSA algorithm based on simulated annealing strategy has proved efficient to guide an AGV to find the optimal path in the grid large-scale state space, combining the potential field method and deep q-network algorithm (Liao et al., 2020). The paper by (Guo et al., 2020) addressed the path planning using the duelling double deep q-network with prioritized experience reply (Duelling DDQN-PER). The system learns how to control the AGV in a simulation environment of an intelligent logistics industry. The AGV approaches the target location and avoids obstacles using multi-modal sensory environment information. Due to the lack of a prior map, Sun and Li propose an end-to-end path planning method to make the AGV find the optimal action from the original visual image and Light Detection and Ranging (LIDAR) information (Sun & Li, 2020). In addition, a deep reinforcement learning (DRL) strategy is used to train the AGV, combining priority experience reply mechanism and double deep q-network with the duelling architecture. This way the AGV has certain adaptability to face the unknown and dynamic environment. The recent paper by Yu et al. (2021) obtains a path for an AGV in a virtual environment with unknown obstacles by an improved ant colony (ACO) algorithm. The ACO is used as training data of the reinforcement learning process to obtain the Q-table (Yu et al., 2021). During the AGV movement, the action is selected by the Q-table until the target point is reached. More sophisticated but related techniques have been also applied to AGVs path planning and obstacle avoiding. In (Hu, Yang, et al., 2021), the Multi-Agent Deep Deterministic Policy Gradient (MADDPG) method is proposed to solve the anti-conflict path planning, and the Gumbel-Softmax strategy is then applied to discretize the scenario created by the node network of the magnetic nail guided AGV. In (Van et al., 2021), an intelligent navigation system in unknown 2D environments based on DRL for an autonomous mobile robot is presented.

RL-based path following has been applied to autonomous vehicles, although mainly marine or aerial ones. In (Zhang et al., 2020) the authors proposed a deep interactive reinforcement learning method for path following of autonomous underwater vehicles by combining DRL and interactive RL. The learning method learns from both, human rewards, and environmental rewards at the same time. In (Martinsen & Lekkas, 2018), the straight-path following problem is solved for underactuated marine vessels under the influence of unknown ocean current by RL. To do it, a dynamic model of the Marine Systems Simulator was used to simulate the motion of a mariner-class vessel. In (Rubí et al., 2021), the authors deal with the path following problem of a quadrotor vehicle based on DRL theory. Three different approaches of the Deep Deterministic Policy Gradient algorithm are implemented. In (Zhu et al., 2019), a path-integral-based RL algorithm is proposed for the path following strategy of an auto-assembly mobile robot, where three kernel techniques are introduced. In (Zhao et al., 2022), authors present an adaptive optimal control to minimize a cost function without discount factor. It reformulates the infinite-horizon optimal setpoint tracking problem for linear discrete-time systems with external disturbances. Then, a Q-learning algorithm is designed to learn the suboptimal control policy by using measured data. Lala et al. validate a model-free Value Iteration Reinforcement Learning (MFVI-RL) control on a visual servo tracking system. In this case, the learning is based on a virtual state representation reconstructed from input–output (I/O) system samples (Lala et al., 2021). In (Yeh & Yang, 2021) reinforcement learning methods are used to perform automatic tuning of the gains of a PID. Finally, Radac et al. describe a three-level model-free hierarchical learning approach to solve the reference trajectory tracking problem for control systems. It consists of a low-level neural controller, an intermediate level Model-free Iterative Learning Control, and a high level control with a library of memorized primitives (Radac & Precup, 2016).

RL has been also explored for AGV fleet management. To mention some works, a self-adaptive traffic control model combining behaviour trees (BTs) and reinforcement learning (RL) is proposed in (Hu, Jia, et al., 2021). The BTs are used to enumerate all the possible states in AGVs traffic control and then, the RL model is further developed based on these. A cyber-physical system that uses multiagent system technology is designed, where components such as AGVs and traffic commander are defined as specific agents that cooperate autonomously with each other. In (Hu et al., 2020), an adaptive DRL-based strategy for an AGVs real-time scheduling with mixed rule is proposed. The goal is to minimize the makespan and delay ratio. In (Xue et al., 2018), a multi-AGV flow-shop scheduling problem with a reinforcement learning method is presented. The idea of this proposal is to obtain a AGV schedule that minimizes the average job delay and the total makespan.

As it is possible to see, there are some papers on RL-based path-planning and fleet management, but the works on AGV path-following based on RL are scarce, although most authors agree that AGVs need to have self-learning and adaptive capabilities to cope with changes in the partially unknown environment which in they move.

3 | MATHEMATICAL MODEL OF THE AGV

A model of a hybrid tricycle and differential AGV has been used to implement the proposed control strategy. This type of tow AGV is widely used in the industry. The traction unit works as a differential mobile robot. However, this part is joined to the body of the AGV by an axle on which it pivots. On the other hand, the kinematics of the body of the AGV is like a tricycle vehicle but, in this case, instead of having a wheel to control the steering, this is done by the traction unit. Figure 1 shows a schematic representation of the hybrid AGV.

The electromechanical model of the AGV is described by Equations (1–12). These equations are explained in more detail in (Sierra-García & Santos, 2020a).

Dynamic equations:

$$M_{eR} = M_R - F_{sw_R} \cdot \mathrm{sign}\left(\dot{\theta}_R\right), M_{eL} = M_L - F_{sw_L} \cdot \mathrm{sign}\left(\dot{\theta}_L\right), \tag{1}$$

$$\begin{bmatrix} m_T \cdot R_h/2 & m_T \cdot R_h/2 \\ \frac{I_h \cdot R_h}{L_h} & -\frac{I_h \cdot R_h}{L_h} \end{bmatrix} \begin{bmatrix} \ddot{\theta}_R \\ \ddot{\theta}_L \end{bmatrix} = \begin{bmatrix} \frac{(M_{eR}+M_{eL})}{2R_h} - f_{rT} \\ \frac{(M_{eR}-M_{eL})L_h}{2R_h} - f_{rR} \end{bmatrix}, \tag{2}$$

$$f_{rT} = 0.5 \cdot \delta_{air} \cdot S_{AGV} \cdot C_{aero} \cdot \left(v_h{}^2\right) \cdot \mathrm{sign}(v_h) + 9.8 \cdot m_T \cdot C_{roll} \cdot \mathrm{sign}(v_h), \tag{3}$$

$$f_{rR} = F_{vh} \cdot \dot{\Phi}_h + F_{sh} \cdot \mathrm{sign}\left(\dot{\Phi}_h\right), \tag{4}$$

Kinematic equations:

$$v_L = R_h \cdot \dot{\theta}_L, v_R = R_h \cdot \dot{\theta}_R, \tag{5}$$

$$\dot{x}_h = \frac{V_L + V_R}{2} \cos(\Phi_h), \dot{y}_h = \frac{V_L + V_R}{2} \sin(\Phi_h), \dot{\Phi}_h = \frac{v_R - v_L}{L_h}, \tag{6}$$

$$\dot{x}_b = v_h \cos(\gamma) \cos(\Phi_b), \dot{y}_b = v_h \cos(\gamma) \sin(\Phi_b), \dot{\Phi}_b = \frac{v_h}{L_b} \sin(\gamma), \tag{7}$$

$$v_h = \sqrt{\dot{x}_h^2 + \dot{y}_h^2} = \frac{v_L + v_R}{2}, \tag{8}$$

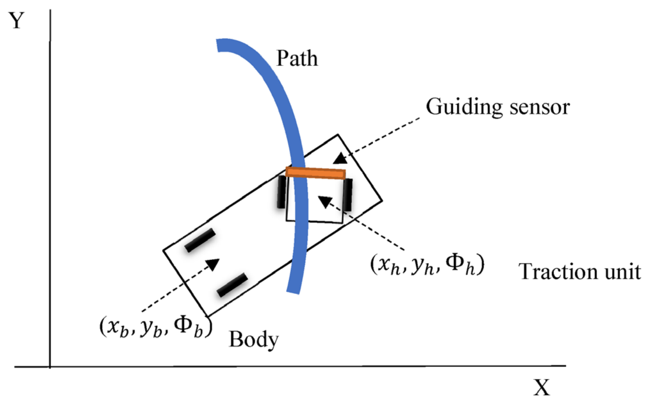

FIGURE 1 Traction unit and body of the AGV.

$$\gamma = \Phi_h - \Phi_b, \gamma_{min} \le \gamma \le \gamma_{max}, \tag{9}$$

$$\dot{x}_b \sin(\Phi_b) - \dot{x}_b \cos(\Phi_b) = 0, \tag{10}$$

$$\dot{x}_b \sin(\Phi_b + \gamma) - \dot{x}_b \cos(\Phi_b + \gamma) - \dot{\Phi}_b L_b \cos(\Phi_b) = 0, \tag{11}$$

Guiding sensor:

$$err_{gui} = f_{sen}(x_h, y_h, \Phi_h, \text{path}) \tag{12}$$

where (x_h, y_h, Φ_h) and (x_b, y_b, Φ_b) denote the position (m) and orientation (rad) of the body and the traction unit, respectively. The variable v_h (m/s) is the longitudinal velocity of the traction unit, L_h (m) is the distance between the wheels in the traction unit, L_b (m) is the distance between the rear wheels and the center of the traction unit, m_T (kg) is the total mass of the system, that is, the mass of the AGV, m_{AGV} (kg), plus the mass of the load, m_L (kg). The radius of the wheels in the traction unit is R_h (m); I_h (kg m^2) is the rotational inertia of the traction unit; $\ddot{\theta}_r$ (rad/s^2) and $\ddot{\theta}_l$ (rad/s^2) are the angular velocities of the right and left wheel of the traction unit, respectively; M_{eR} and M_{eL} are the effective torques in the right and left wheel of the traction unit, in Nm; f_{rT} (N) and f_{rR} (N) are the translational and rotational friction forces, δ_{air} (kg/m^3) is the density of the air, and S_{AGV} (m^2) is the front surface of the AGV. The coefficients involved in the system are: C_{aero}, the aerodynamic coefficient; C_{roll}, the rolling coefficient; F_{vh} and F_{sh} are the viscous and static friction coefficients, respectively, in the traction unit; F_{vw} and F_{sw} are the viscous and static friction coefficients, respectively, in the wheels. The torques produced by the motors of each wheel are M_R and M_L, in Nm. Finally, sign() is the sign function. The effect of the inertia of the wheels has been neglected. It is supposed that the AGV is working in an indoor environment and hence the wind does not affect the movement.

The parameters that have been used in the simulation are collected in Table 1.

This type of AGV can be equipped with different navigation systems or guiding sensors. In this work, a magnetic sensor placed in the front of the traction unit has been considered. The path to be followed is drawn on the floor by a magnetic tape. This magnetic tape is very robust, it hardly degrades with time and requires less maintenance than optical systems such as QR codes or painted lines. As it is shown in Figure 2, the magnetic sensor provides the distance between the center of the magnetic tape and the center of the guiding sensor, what is usually called the guiding error.

In Figure 2, P_1 and P_2 denote an edge and the center of the guiding sensor and are obtained considering the rotation and the translation of the drive unit in the inertial frame (13–14). The set of points that belong to the projection of the guiding sensor are R_S, giving by Equation (15). The point where the projection of the guiding sensor intersects with the predefined path is P_s (16). As the projection of the guiding sensor

TABLE 1 Parameters of the AGV model.

Parameter	Description	Value/units
L_h	Distance between wheels of the AGV	30 cm
L_b	Distance between rear wheels and traction unit	100 cm
R_h	Radius of front wheels	6 cm
R_b	Radius of rear wheels	9 cm
m_{AGV}	Mass of the AGV	100 kg
I_h	Inertia of traction unit	0,11 kg m^2
δ_{air}	Density of the air	1.225 kg/m^3
S_{AGV}	Front Surface of the AGV	0.5 m^2
C_{roll}	Rolling coefficient	0.01
C_{aero}	Aerodynamic coefficient	0.35
F_{sh}	Static friction coefficient of traction unit	0.1 N
F_{vh}	Viscous friction coefficient of traction unit	0.01 Ns/rad
F_{sw}	Static friction coefficient of traction wheels	2.94e-2
F_{vw}	Viscous friction coefficient of traction wheels	5e-4 Ns/rad
$[K_{vp}, K_{vI}]$	PID constants for velocity control of wheels	[2, 0.1]
$[K_{\Phi p}, K_{\Phi D}, K_{\Phi I}]$	PID constants for angular velocity control of the traction unit	[9.8, 1, 0.1]

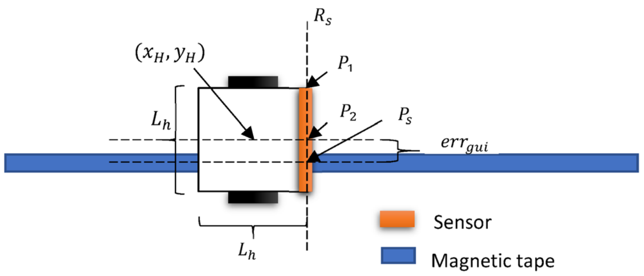

FIGURE 2 Calculation of the guiding error.

with the path may intersect in more than one point, the guiding error, err_{gui}, is defined as the minimum distance, that is, (17). The process to obtain these values is formalized by Equations (13–17).

$$P_1 = \begin{bmatrix} \cos(\Phi_h) & -\sin(\Phi_h) \\ \sin(\Phi_h) & \cos(\Phi_h) \end{bmatrix} \begin{bmatrix} L_h/2 \\ L_h/2 \end{bmatrix} + \begin{bmatrix} x_h \\ y_h \end{bmatrix}, \tag{13}$$

$$P_2 = \begin{bmatrix} \cos(\Phi_h) & -\sin(\Phi_h) \\ \sin(\Phi_h) & \cos(\Phi_h) \end{bmatrix} \begin{bmatrix} L_h/2 \\ 0 \end{bmatrix} + \begin{bmatrix} x_h \\ y_h \end{bmatrix}, \tag{14}$$

$$R_s = \left\{ (x,y) \in \mathbb{R}^2 \middle| y = \left(\frac{P_2.y - P_1.y}{P_2.x - P_1.x} \right) (x - P_1.x) + P_1.y \right\}, \tag{15}$$

$$P_S = \left\{ (x,y) \in \mathbb{R}^2 \,\middle|\, (x,y) \in R_s \wedge (x,y) \in path \right\}, \tag{16}$$

$$err_{gui} = \underset{p \in P_S}{\mathrm{MIN}} [dist(p, P_2)], \tag{17}$$

4 | CONTROL ARCHITECTURE

As said before, there are two coupled control problems in the AGV. The speed control is designed to follow the speed profile, $v_{t_{REF}}$, which depends on the application, and the tracking control minimizes the guiding error. Both controls are important and must be simultaneously implemented. The control of the velocity is required to meet some specific requirements of the production chain and/or the logistic process, such as the tag time or the pace of the flow. Moreover, being able to follow the correct speed profile is key to ensure that the AGV stops at the transport stations accurately. On the other hand, the tracking control must be also precise for safety reasons and to guarantee a correct logistic operation. Indeed, it is necessary to prevent the AGV from operating in restricted areas; it must be able to pass through narrow corridors and to be correctly positioned at the stations; in addition, it is important to avoid unexpected safety stops that can take place if the AGV travels very close to obstacles detected by the lidar.

Equations (1–4) showed that the relationship between the torque applied by the motors, M_R and M_L, to the wheels and the velocity of the wheels, $\dot{\theta}_R$ and $\dot{\theta}_L$, are mainly linear. The non-linearities are introduced by the kinematic equations and the paths that must be followed. Thus, in this approach we exploit this fact by using a conventional linear PI controller to control the wheel speed; and an intelligent technique based on RL to address the non-linearities of the tracking control. Both techniques are interconnected because the output of the RLC feeds the inputs of the PI regulators. This way, iteration by iteration, the RLC learns to follow the path by exploring the action space in order to find the actions that most reduce the guiding error at each state.

The intelligent hybrid control scheme is shown in Figure 3. It consists of three different controllers: the RLC and two PIs. For each wheel, a specific PI adjusts the input torque of the motors (M_R, M_L) to follow the wheel speed reference $(v_{R_{REF}}, v_{L_{REF}})$, (16–19). These wheel speed references are obtained considering the inverse kinematics of the traction unit (14–15). The inverse kinematics module receives the angular reference velocity w_{REF} obtained by the RL controller and the longitudinal reference speed, v_{tREF}. This way the speed profile required by the application, v_{tREF}, is considered in the control scheme, and the angular velocity is adjusted to follow the path correctly, solving both control problems simultaneously.

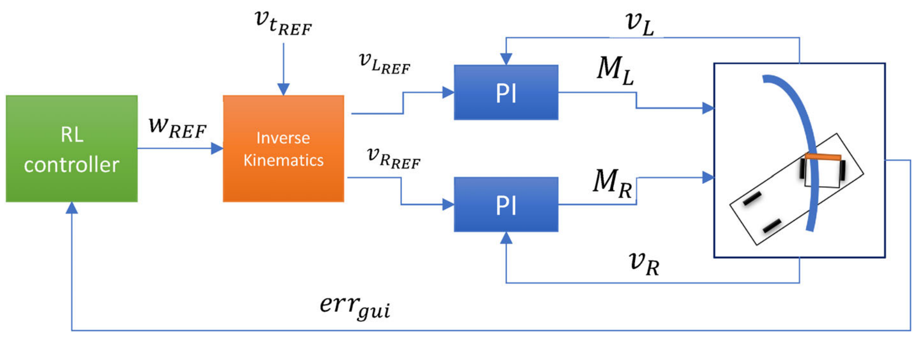

FIGURE 3 Proposed intelligent hybrid control architecture.

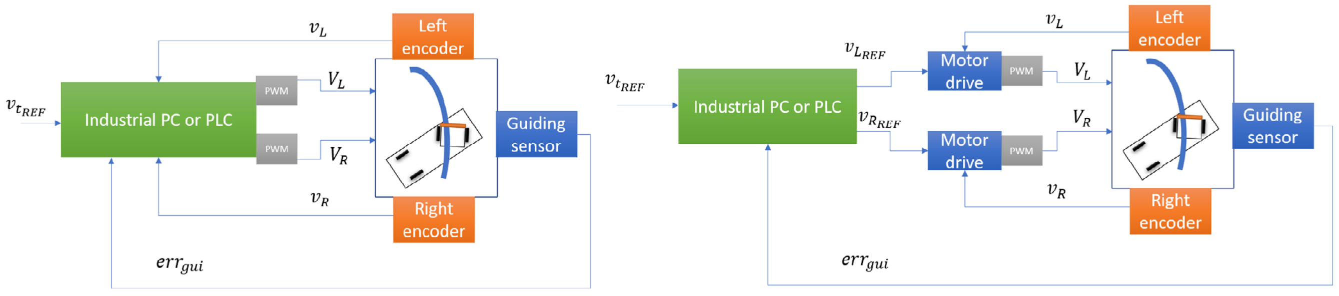

FIGURE 4 Implementation of the control on an industrial PC only (left) and with motor drives (right).

In turn, the reference for the angular speed of the traction unit, w_{REF}, is obtained by the RLC to minimize the guiding error and to follow the path. It receives the guiding error, err_{gui}, from the guiding sensor as input. From this value, it estimates the state of the system, computes the reward, updates the policy, and proposes the new action to be executed.

There are different options to implement this control scheme. The first alternative is that all the modules can be run in an industrial computer or program logic controller (PLC) (Figure 4, left). This way the output of the guiding sensor and the encoders are connected to the controller. This controller generates the pulse width modulation (PWM) signals for the motors. In the second approach, the PI controllers run in motor drives as they usually provide this speed control feature or have the capacity to implement this function (Figure 4, right). In this topology, the main controller still receives the guiding error, but it generates the speed reference to the motor drives. Normally, the motor drives also provide some interface to notify the current speed to the main controller in order to let it know the state. The selection of the implementation mode will depend on the interfaces available to control the motors of the vehicle.

The performance of this control strategy can be formalized by Equations (18–24).

$$w_{REF}(t_i) = f_{RL}(err_{gui}(t_{i-1}), err_{gui}(t_{i-2})), \tag{18}$$

$$v_{L_{REF}}(t_i) = v_{t_{REF}}(t_i) - \frac{w_{REF}(t_i)\cdot L_h}{2}, \tag{19}$$

$$v_{R_{REF}}(t_i) = v_{t_{REF}}(t_i) + \frac{w_{REF}(t_i)\cdot L_h}{2}, \tag{20}$$

$$err_{vL}(t_i) = v_{L_{REF}}(t_i) - v_L(t_{i-1}), \tag{21}$$

$$err_{vR}(t_i) = v_{R_{REF}}(t_i) - v_R(t_{i-1}), \tag{22}$$

$$M_L(t_i) = K_{vp}\cdot err_{vL}(t_i) + K_{vI}\cdot \int err_{vL}(t_i)dt, \tag{23}$$

$$M_R(t_i) = K_{vp}\cdot err_{vR}(t_i) + K_{vI}\cdot \int err_{vR}(t_i)dt \tag{24}$$

The gains of the PI speed controller have been tuned by trial and error and have been set to $[K_{vp}, K_{vI}] = [2, 0.1]$. In addition, the desired value of the longitudinal speed, $v_{t_{REF}}$, is usually given by the real application according to the expected logistic flow.

4.1 | Reinforcement learning controller

Iterative learning has been widely used to model learning processes (Trojaola et al., 2020). Reinforcement learning is one of these computational learning approaches. It consists of an environment, an agent, and an interpreter. The agent, taking into consideration the current state of the environment s_t and the previous rewards $[r_t, r_{t-1}, \ldots r_{t-N}]$, selects the best action a_t to be carried out. This action produces an effect on the environment. This fact is observed by the interpreter who gives information about the new state, s_{t+1}, and the reward, r_{t+1}, of the previous action a_t, to the agent, closing the loop. Some authors consider that the interpreter is embedded in either the environment or the agent; anyway, the function of the interpret is always present (Sierra-García & Santos, 2020b).

The discrete reinforcement learning can be expressed as follows (Sutton & Barto, 2018):

- S is a finite set of states
- A is a finite set of actions
- $\mathrm{s_t}$ is the state at t
- $\mathrm{a_t}$ is the action executed when the agent knows the environment at state $\mathrm{s_t}$
- r_{t+1} is the reward received after action a_t is executed in the state s_t.
- s_{t+1} is the state after action a_t is executed at state s_t.
- The environment or world is a Markov process:

$$\mathrm{MDP} = < \mathrm{s_0, a_0, r_1, s_1, a_1, r_2, s_2, a_2 \ldots} >$$

- $\pi : S \times A \to [0,1]$ is the policy; this function calculates the probability of selecting an action a for every pair (s,a).
- $p^a_{ss'} = \Pr\{s_{t+1} = s' | s_t = s \wedge a_t = a\}$ is the probability that state s changes to $s^{'}$ with action a
- $p^\pi(s', a')$ is the probability of selecting action a' at state s' under policy π
- $r^a_s = E\{r_{t+1} | s_t = s \wedge a_t = a\}$ is the expected one-step reward
- $Q^\pi_{(s,a)}$ is the expected long-term reward.

The general objective of the reinforcement learning algorithm is to find the best policy π^* that maximizes $Q^\pi_{(s,a)}$ for every state; formally:

$$\pi^* = \mathrm{argMAX}_\pi \left[Q^\pi_{(s,a)} \right] \forall s \in S \tag{25}$$

To obtain the state of the system, the error, err_{gui}, and its derivate, $\dot{err}_{gui}$, are scaled and then discretized. After this, err_D is an integer between 0 and $(n_e - 1)$, and $derr_D$ is also an integer between 0 and $(n_{de} - 1)$. There is a bi-univocal correspondence between err_{gui} and err_D since each err_{gui} only matches one err_D, and conversely (26–27). In turn, there is a bi-univocal correspondence between $\dot{err}_{gui}$ and $derr_D$. These discretized states, err_D and $derr_D$, are linearly combined to obtain the discretized state of the system, s_t, in a way that ensures the bi-univocal correspondence (28).

$$err_D = INT\left(n_e \frac{MIN(e_{\max}, MAX(err_{gui}, e_{\min})) - e_{\min}}{e_{\max} - e_{\min}} \right) \tag{26}$$

$$derr_D = INT\left(n_{de} \frac{MIN(de_{\max}, MAX(\dot{err}_{gui}, de_{\min})) - de_{\min}}{de_{\max} - de_{\min}} \right) \tag{27}$$

$$s_t = err_D \cdot n_{de} + derr_D \tag{28}$$

Where INT(x) is the maximum integer that is smaller than x, MIN is the minimum, MAX is the maximum, and $[e_{\min}, e_{\max,} de_{\min}, de_{\max}, n_e, n_{de}]$ is a set of parameters to adjust the number of states and the size of each state.

A practical way to illustrate how this state assignment works is to think about a table. The rows of the table are pointed at by err_D and the columns by $derr_D$. If we go through the table from top to bottom, and from left to right, we can assign a unique integer to each cell in the table using Equation (28). Thus, there exists a bi-univocal correspondence $s_t \leftrightarrow (err_D, derr_D)$.

The reward strategy also considers the guiding error and its derivative. If the AGV performs an action and as a result the vehicle moves closer to the path, this action is rewarded; otherwise, the action is punished. This can be calculated considering the sign of the derivative of the guiding error (29).

$$r_t = \begin{cases} -\dot{err}_{gui} & err_{gui} > 0 \\ -\left|\dot{err}_{gui}\right| & err_{gui} = 0 \wedge \dot{err}_{gui} \neq 0 \\ de_{max} & err_{gui} = 0 \wedge \dot{err}_{gui} = 0 \\ \dot{err}_{gui} & err_{gui} < 0 \end{cases} \tag{29}$$

The desirable state from the point of view of guiding error is to obtain zero error and to remain stable there. Thus, if the guiding error and the derivative of this error are zero, this action is rewarded with the maximum value as this action tends to maintain the system in the desired state (29). However, if the guiding error is zero but its derivative is different from zero, this action is punished as it tends to leave the zero-error state.

There are different options to update the policy and to obtain the long-term expected reward. We have used the summatory of rewards. Using this update rule, the system is able to remember all previous rewards by accumulating them (30). This update rule has speed up the learning in other control problems (Sierra-García & Santos, 2020c). But this update rule is only recommended if the reward calculator assigns positive and negative rewards, as in this case. Positive values are used as rewards and negative ones as punishments. If only positive rewards are used, some actions may be selected repeatedly and many others could be explored at a lower frequency. The use of punishments (negative rewards) prevents some actions to be over selected in a natural way.

$$Q^{\pi}_{(s_t,a_t)} = Q^{\pi}_{(s_t,a_t)} + r_{t+1} \tag{30}$$

Once the long-expected reward is updated, the next action is obtained considering the long-expected rewards associated to the current state, s_t. This action is selected by the ϵ-greedy strategy (Sutton & Barto, 2018). Each control period, a random number is generated; if the number is below the threshold ϵ, the next action a_t is randomly generated. In other case, the controller selects the action with biggest expected reward for the current state, s_t (31). Then this discrete action a_t is transformed into a reference for the angular velocity by Equation (32).

$$a_t = \begin{cases} \underset{a \in A}{\operatorname{argMAX}}\left[Q^{\pi}_{(s_t,a)}\right] & rand() \geq \epsilon \\ n_{act} \cdot rand() & rand() < \epsilon \end{cases} \tag{31}$$

$$w_{REF} = w_{r_{min}} + a_t\left(w_{r_{max}} - w_{r_{min}}\right)/n_{act} \tag{32}$$

where ϵ is a real number between 0 and 1 that represents the probability to select a random action; n_{act} is the number of actions, and $[w_{r_{min}}, w_{r_{max}}]$ is a set of parameters to adjust the size of the discrete actions.

As shown, the equations of the RLC (18–32) have low computational complexity and can be executed in whatever PLC or industrial PC considering this control cycle. Normally, a control cycle between 10 and 20 ms is used with this type of AGV. In the simulation experiments we have considered a cycle time of 10 ms.

During the simulation, the sample time, T_s, is varied to reduce the discretization error, with a maximum value set to 5 ms. An action is executed within a control period, and the reward of this action is evaluated in the next control period. As the maximum sample time of the simulation is smaller than the control period, the simulation has enough time to react to the action of the controller and to calculate the next state before the next control period happens.

5 | RESULTS AND DISCUSSION

This intelligent control approach has been tested in simulation with the model of a hybrid AGV with the parameters listed in Table 1. The results have been obtained using Matlab/Simulink software. Each simulation runs for 100 s, unless the AGV leaves the path and then the simulation is stopped.

In these industrial vehicles the controller of the guiding error is normally a PID. Thus, the performance of the proposed controller is compared with a PID regulator that follows Equation (33).

$$w_{REF}(t_i) = K_p \cdot err_{gui}(t_i) + K_d \cdot \frac{d}{dt} err_{gui}(t_i) + K_i \cdot \int err_{gui} \tag{33}$$

where $[K_p, K_d, K_i]$ are the tuning parameters of the PID, that have been obtained by trial and error. Their values are $[9.8, 1, 0.1]$.

The following KPIs are defined to test the performance of the controller:

$$MAE = \frac{1}{T_{sim}} \sum_i \left| err_{gui}(t_i) \right| \cdot T_s(t_i) \tag{34}$$

$$RMSE = \sqrt{\frac{1}{T_{sim}} \sum_i err_{gui}(t_i)^2 \cdot T_s(t_i)} \tag{35}$$

$$ME = \frac{1}{T_{sim}} \sum_i err_{gui}(t_i) \cdot T_s(t_i) \tag{36}$$

$$STD = \sqrt{\frac{1}{T_{sim}} \sum_i \left[err_{gui}(t_i) - ME \right]^2 \cdot T_s(t_i)} \tag{37}$$

$$MAX = \underset{t_i \in T_{sim}}{\mathrm{MAX}} \left[err_{gui}(t_i) \right] \tag{38}$$

Four different paths have been used to validate the controller: an ellipse, a lemniscate, and two closed polylines. Ellipses are useful as they can be used to approximate closed oval loops that are used in industrial applications. The lemniscate is also interesting as the curving radius changes along the trajectory and this fact can be used for modelling paths with change of direction. It is not easy to draw curve paths with magnetic tapes or another guiding references; it requires some practice. For this reason, it is faster to draw polyline paths; indeed, sometimes they are used if the technician is not very experienced with curving paths. However, these paths are not smooth and discontinuities are introduced in the velocity, making easier for the AGV to get out of its guide. The expressions of these trajectories are as follows,

Ellipse:

$$x(t) = a_{eli} \cdot \cos(t) \tag{39}$$

$$y(t) = b_{eli} \cdot \sin(t) \tag{40}$$

Lemniscate:

$$x(t) = \sqrt{2} \cdot a_{lem} \cdot \frac{\cos(t)}{1 + \sin(t)^\wedge 2} \tag{41}$$

$$y(t) = \sqrt{2} \cdot a_{lem} \cdot \frac{\cos(t) \cdot \sin(t)}{1 + \sin(t)^\wedge 2} \tag{42}$$

Polyline:

$$x(t) = Px_{i-1} + t, i = 1 \ldots M, 0 > t > (Px_i - Px_{i-1}) \tag{43}$$

$$y(t) = \frac{Py_i - Py_{i-1}}{Px_i - Px_{i-1}} t + Py_{i-1}, i = 1 \ldots M, 0 > t > (Px_i - Px_{i-1}) \tag{44}$$

where a_{eli} and b_{eli} are the semi-axis of the ellipse, a_{lem} is the width of the lemniscate and $[Px, Py] \in \mathbb{R}^{2M}$ is the set of points of the polyline. Due to the kinematic constraints of the AGV, the path gets harder as a_{eli}, b_{eli}, or a_{lem} get smaller and smaller. If they are too small, the AGV leaves out the path. Several simulations have been carried out to find parameters small enough to make the trajectory tracking complex, but not too small to make it impossible. Finally, these values have been set to $[a_{eli}, b_{eli}, a_{lem}] = [1.4, 0.7, 2.34]$.

Two different polyline trajectories were selected, an 8-sided polygon (octagon) and a 24-sided polygon. The 8-sided polygon has the set of points

$$\{(\mathrm{Px},\mathrm{Py})\}=\{(0,0),(1.25,0),(2.5,1.25),(2.5,2.5),(1.25,3.75),(-1.25,3.75),(-2.5,2.5),(-2.5,1.25),(-1.25,0),(0,0)]$$

whereas the 24-sided polygon is defined by the set of points

$$\begin{aligned}\{(Px,Py)\}=\{&(0,0),(1.25,0),(2.5,1.25),(3.75,1.25),(5,0),(6.25,0),(7.5,1.25),(8.75,1.25),(10,0),(10,-1.25),(8.75,-2.5),(7.5,-2.5),\\&(6.25,-1.25),(5,-1.25),(3.75,-2.5),(2.5,-2.5),(1.25,-1.25),(0,-1.25),(-1.25,-2.5),(-2.5,-2.5),(-3.75,-1.25),\\&(-3.75,0),(-2.5,1.25),(-1.25,0),(0,0)\}.\end{aligned}$$

These polygons are shown in Figures 7 and 8.

5.1 | Performance of the controller

In the following experiments, we are going to evaluate the solutions proposed for the two coupled control problems involved in the AGV movement: the path tracking and the longitudinal speed control. The performance of the controller is tested with the four described trajectories: ellipse, lemniscate, octagon, and 24-sided polygons. The longitudinal reference speed of the AGV, v_{tREF}, is a sinusoidal signal with amplitude 0.35 m/s and average value of 0.35 m/s. A maximum speed of 0.7 m/s, a common value for this variable in industrial applications with this type of AGV, is allowed. Indeed, the maximum speed is 1 m/s and it is only reached in long straight paths. The results show the performance obtained by the RL controller once it has learnt to follow the path. Each training episode lasts 100 s or until the AGV goes off the path.

Figures 5-8 show the trajectory described by the traction unit of the AGV (left) and the guiding error (right), when the PID and the RL controllers are used. The reference path is represented in blue, the path followed by the AGV when the PID is used in yellow, and the obtained when the RL is used in red. On the other hand, the guiding error with the PID controller is represented in red, and the guiding error with the RL-based control is represented in blue.

For the elliptical trajectory, the AGV starts at the top of the ellipse, point (0, 0.7), and moves clockwise. It is possible to observe in Figure 5, left how the RL line and the reference are almost overlapped and cannot be distinguished in some parts of the trajectory. However, the difference between the PID line and the reference is noticeable. Therefore, the performance of the RL controller is much better. This can be also seen in the representation of the error signals (Figure 5, right). The RL error is below 2 cm, but with the PID the error has big peaks up to 8–10 cm. As the AGV follows the path in clockwise direction, the negative error peaks are bigger than the positive ones.

In the case of a lemniscate trajectory, the path followed by the AGV with the PID and the RL control methods are more similar, and with both the reference is well followed (Figure 6, left). The error with the RL controller is noisier but with a lower amplitude (Figure 6, right). Even more,

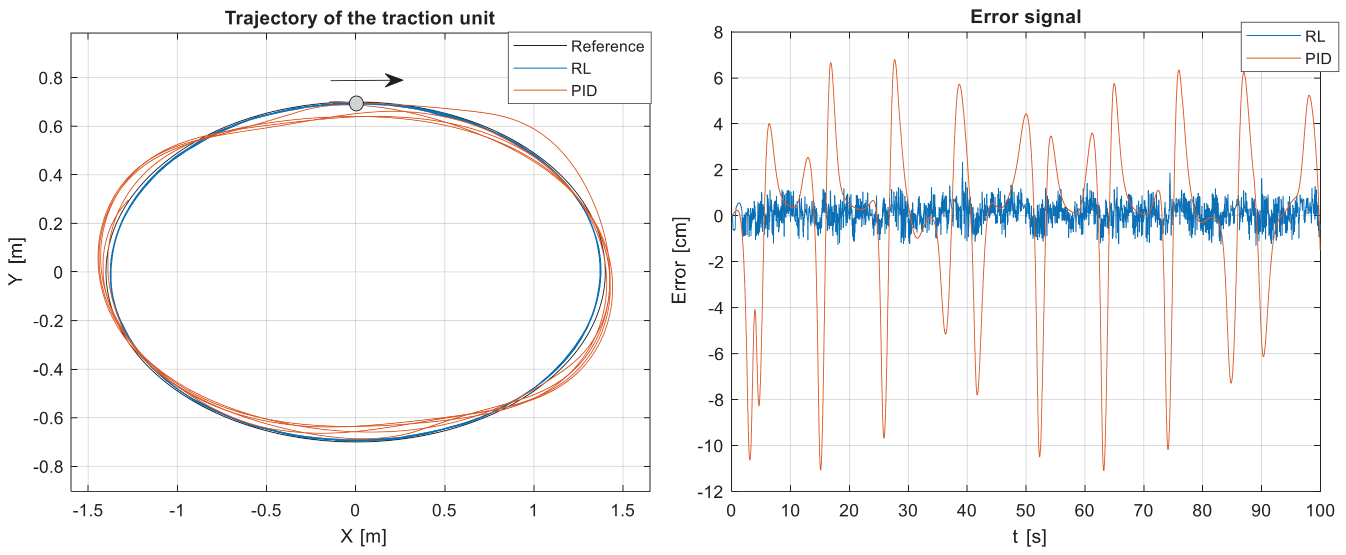

FIGURE 5 Elliptical trajectory (left) and corresponding guiding error (right).

the mean error with the RL control is around 1 cm and the PID response error is slightly higher. The biggest error with the PID is a large negative peak around t = 2 s, that happens when the AGV starts moving as the AGV is aligned with the x-axis and it must make an abrupt manoeuvre. This can be also observed in Figure 6, left, considering that the AGV starts at the top right of the lemniscate, point (2.02, 1.17), moving clockwise, and the biggest error occurs at that point, at the beginning.

In Figure 7, left, an 8-sided polygon is selected as AGV reference trajectory. The AGV starts at the point (0, 0) and the trajectory is counter clockwise. Again, the path obtained with the RL controller follows almost perfectly the reference. Regarding the error, the oscillations with the PID are very noticeable (Figure 7, right). Even more, although both control strategies present peaks in the error signal, the PID response has much larger error peaks, about 10 cm or even higher. These peaks appear with both controllers because they correspond to the moments when the AGV passes through the edges of the polygon.

As expected, when the polygon has more sides (Figure 8, left), the trajectory is more difficult to follow, and changes are more abrupt. Thus, they are more peaks in the error (Figure 8, right). But again, the RL controller makes the AGV to better follow the reference. Thus, the corresponding error signal is smaller. It is interesting to remark than the peaks seem to be related with the angle between the sides of the polygon, larger angles produce more abrupt changes in the trajectory and larger peaks in the error signal. In this case the AGV starts at point (0, 0) and the trajectory is clockwise.

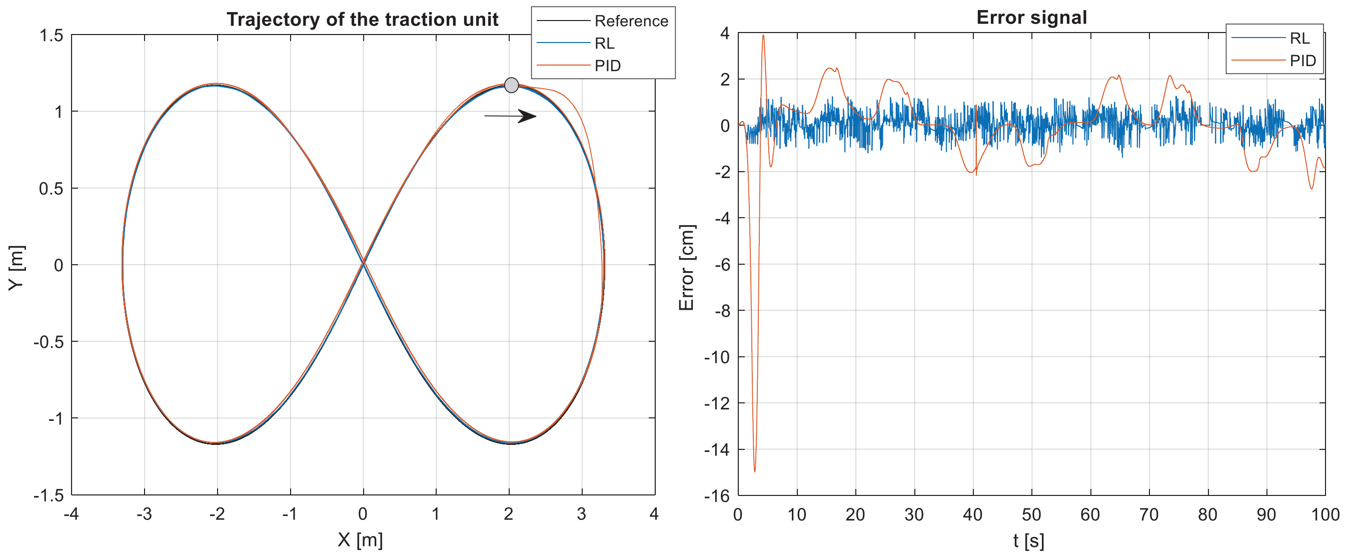

FIGURE 6 Lemniscate trajectory (left) and corresponding guiding error (right).

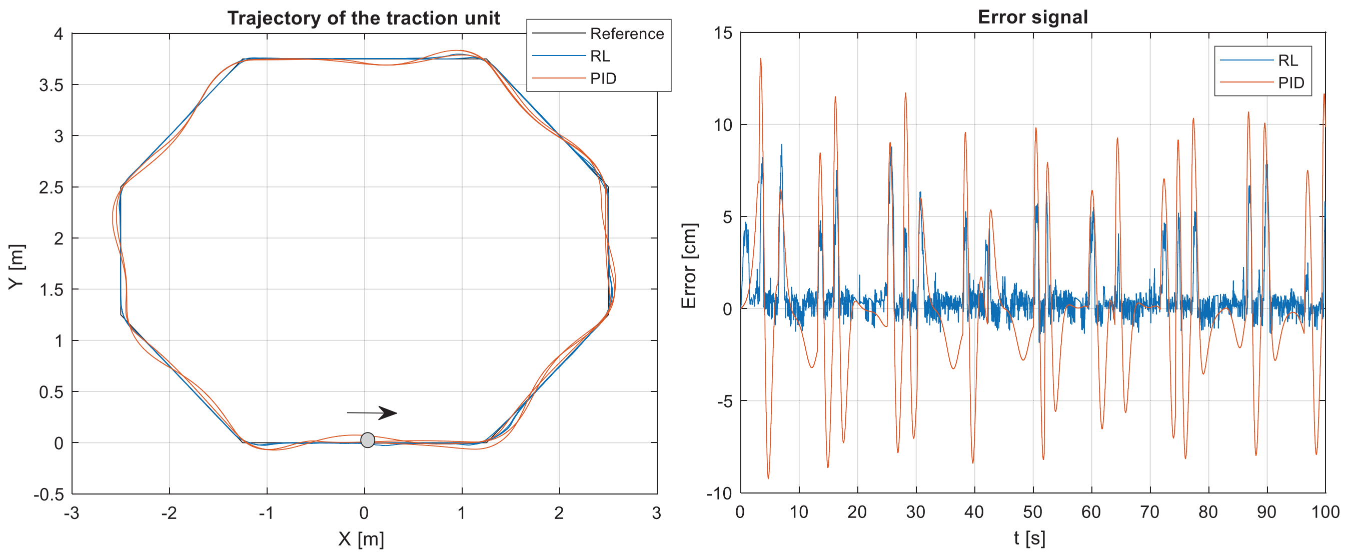

FIGURE 7 An 8-sided polygon trajectory (left) and corresponding guiding error (right).

In addition to the graphical results, quantitative results have been obtained. Table 2 shows the RMSE, the MAE, and the standard deviation of the guiding error for each trajectory and controller. The PID columns represent the KPIs obtained when the PID control is applied and RL columns when the RL controller is used. The last row indicates the average value of the previous rows. Best results have been boldfaced and the results of the trajectory with smaller KPIs have been italicized.

From Table 2, we can confirm what we have observed in the previous figures: every KPI is smaller, that is, better, when the RL-based controller is applied. The largest improvement is obtained for the ellipse, where an improvement of an 85% in MAE, an 87% in RMSE, an 87% in STD, and 82% in MAX is obtained with the intelligent control. The average improvement is again very high, 81% in MAE, 79% in RMSE, 84% in STD, and 63% in MAX. On the whole, the smallest KPIs are obtained for the lemniscate regardless the controller and the worst error metrics for the polygonal trajectories. This was expected as these paths are more abrupt and thus more difficult to follow.

The RL tracking controller has also been compared with a PID optimized by genetic algorithms. RMSE has been used as function cost for the optimization of the PID parameters. The gains of the PID have been tuned for each trajectory. Both control schemes use the same PI speed controller optimized with a genetic algorithm, with gains $[K_{vp}, K_{vI}] = [2.34, 0.1239]$. The function cost used to optimize the speed controller is the RMSE of the speed error. As it is possible to see in Table 3, the results obtained with the optimized PID (OPID) have improved regarding the results of the PID of Table 2; however, RL tracking control still gives a better performance (Tables 2 and 3). Indeed (Table 3), the average MAE, RMSE and STD is 71%, 70%, and 71% smaller than with the trajectory tracking PID control. Moreover, the RLC average MAX is 48% lower than the PID's. It is true that the optimized PI speed controller improves slightly the results of the RL in Table 2, but not for all trajectories, thus the global effect is not very remarkable.

These results allow us to conclude that the RL controller gives a better performance than the PID for these trajectories.

As explained before, the longitudinal target speed follows a sinusoidal profile. To test if this profile is well followed or it is affected by the type of path, Figure 9 shows the longitudinal speed of the AGV when the RL controller is applied and Figure 10 with the PID. In both cases, it is noticeable how the sinusoidal profile is accurately followed regardless the type of trajectory. To appreciate the small variations between the trajectories a zoom is necessary. These variations are larger in the case of the RL controller and during the acceleration phases.

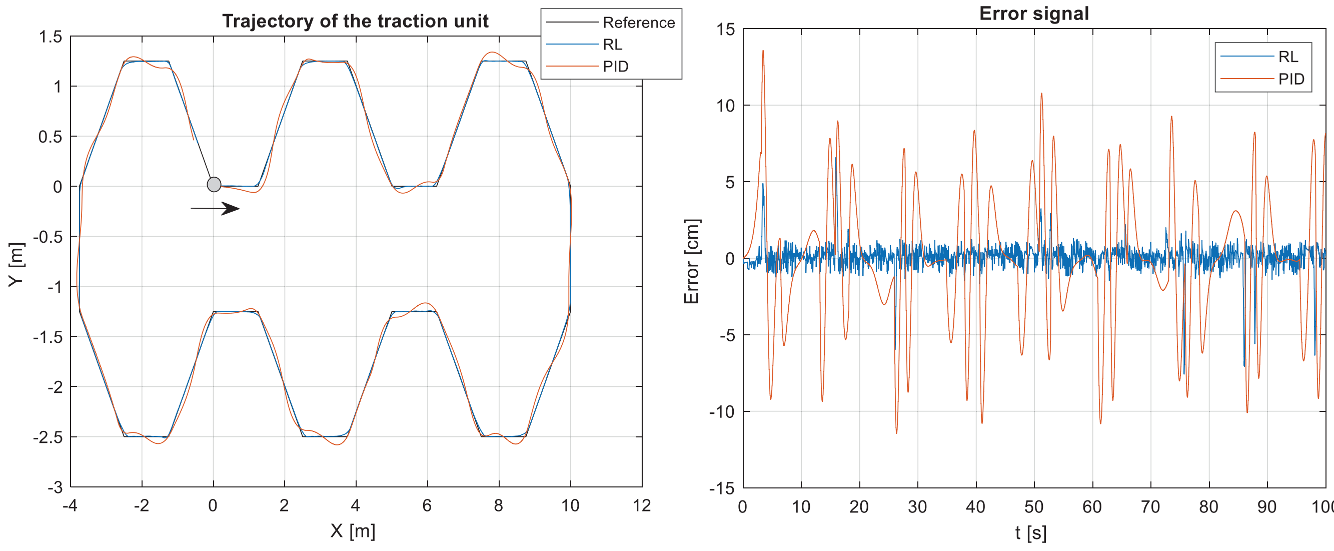

FIGURE 8 A 24-sided polygon trajectory (left) and corresponding guiding error (right).

TABLE 2 Comparison of KPIs for different trajectories with both controllers.

	MAE		RMSE		STD		MAX	
Trajectory	PID	RL	PID	RL	PID	RL	PID	RL
Ellipse	2.51	**0.37**	3.59	**0.46**	4.58	**0.58**	11.07	***2.02***
Le3mniscate	*1.06*	***0.32***	*1.88*	***0.43***	*2.21*	***0.44***	*14.98*	**1.38**
8-S Polygon	3.00	**0.64**	4.22	**1.11**	5.38	**0.89**	13.58	**8.39**
24-S Polygon	3.18	**0.48**	4.3	**0.84**	5.19	**0.79**	13.24	**7.57**
Average	2.44	**0.45**	3.50	**0.71**	4.34	**0.68**	13.21	**4.84**

TABLE 3 Comparison of KPIs for different trajectories. Tracking control PID and PI speed control are optimized by genetic algorithms.

	MAE		RMSE		STD		MAX	
Trajectory	**OPID**	**RL**	**OPID**	**RL**	**OPID**	**RL**	**OPID**	**RL**
Ellipse	1.53	**0.38**	2.15	**0.47**	2.36	**0.47**	7.30	***1.51***
Lemniscate	*0.63*	***0.29***	*1.14*	**0.41**	*1.14*	**0.41**	*9.45*	**2.72**
8-S Polygon	1.67	**0.51**	2.50	**0.87**	2.50	**0.86**	7.71	**5.43**
24-S Polygon	1.85	**0.49**	2.65	**0.83**	2.65	**0.79**	7.71	**6.92**
Average	1.42	**0.41**	2.11	**0.64**	2.16	**0.63**	8.04	**4.14**

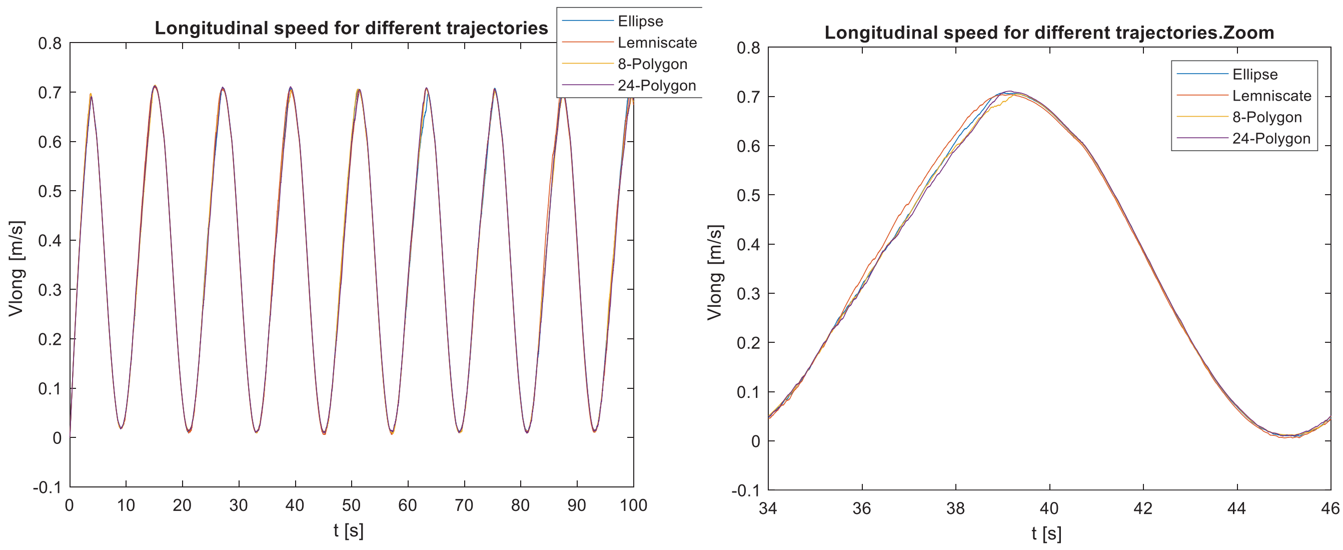

FIGURE 9 Longitudinal speed when the RL controller is used (left) and a zoom of this figure (right).

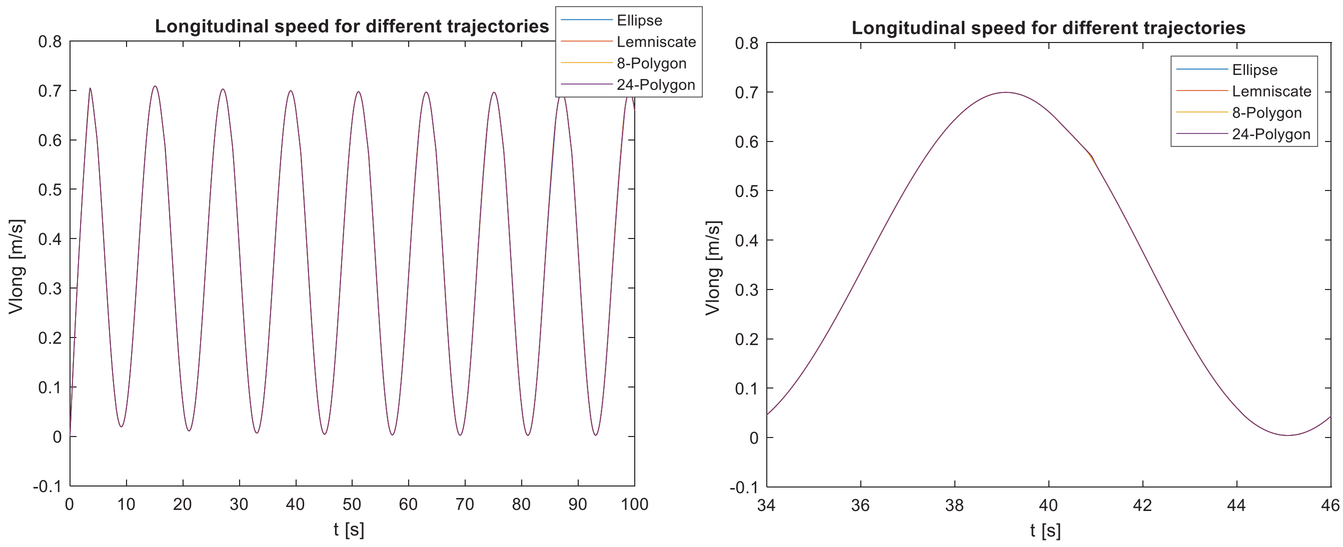

FIGURE 10 Longitudinal speed when the PID controller is used (left) and a zoom of this figure (right).

5.2 | Evolution of the guiding error during the learning

As it has been shown in the previous section, the performance of the RL controller is better than the PID control. Nevertheless, this is not true from the first episode since the RL controller must learn how to control the AGV. Indeed, it learns by interacting with the system episode by episode, and the guiding error is reduced and tends to converge to a value. Each episode finishes when the simulation time is completed or when the AGV loses the reference.

Figure 11 shows the evolution of the MAE (left) and RMSE (right) while the system is learning. The blue lines represent the value of the KPI for the ellipse, the red lines for the lemniscate, the pink for the 8-sided polygon, and the black one for the 24-sided polygon trajectories. The straight lines represent the response with the RL controller and the dashed lines with the PID. As the PID does not learn, the value of the KPI for this controller is the same for all episodes.

As expected, during the first episodes the KPIs are worse with the RL-based control. However, the error decreases fast and few episodes later it is already smaller. For the first episodes all the trajectories give a similar error as the AGV loses soon the reference, then the error starts to decrease until it converges to a value. Interestingly, although the values of the KPI for the lemniscate and the ellipse are quite different when the PID is applied, the final values and the evolution of the KPIs when using the RL controller are similar. The ellipse and the lemniscate lines converge, more or less, to the same value and the polygons trajectories errors tend also to converge to a similar value.

The ellipse and lemniscate lines experiment a faster learning and converge quicker than for the other trajectories; according to the figures, it takes around five episodes and about ten episodes for the ellipse and lemniscate trajectories to converge, respectively. It seems that the control law is easier to learn for smoother paths such as ellipses or lemniscates than for polygons with abrupt changes. Another interesting result is the fact that the MAE converges faster than the RMSE. This may be explained due to the squared term in the RMSE that tends to amplify the small errors more than the MAE.

Hence, these figures confirm the results presented in Table 2. The RL controller gives smaller KPIs than the PID for all the trajectories considered. The episode when the value of the PID is overpassed depends on the type of trajectory. The AGV RL control learns quicker for the ellipse trajectory and the slowest one is for the 24-sided polygon.

5.3 | Robustness with changes in friction

In the previous experiments, the friction coefficients have been set to the values of Table 1. In this section, we study the effect of varying these values. We can distinguish two different frictions, the static friction experimented by each wheel when it rotates, that is called F_{sw}, and the static friction suffered by the traction head when it rotates, called F_{sh} (Figure 1). These values mainly depend on the floor characteristics and may change. Among other factors, the friction can change due to the humidity, dust, and degradation. For instance, dust in suspension that can be accumulated on the wheel axle or on the axle of the tractor head makes the coefficient of friction increase and thus, more energy is needed to rotate the wheels.

Several simulations have been carried out, increasing the value of the friction coefficients until the AGV leaves the path without finishing one loop. The coefficient values when this happens have been collected in Table 4. For each simulation, the friction coefficient of both wheels has been set to the same value, F_{sw}. The PID columns refer to the values obtained with the PID and RL means that the RL controller is used. The RL controller is trained during 25 episodes.

In Table 4, it is possible to observe how the values of the RL columns are much bigger than the PID ones, up to 20 times higher. This means that the RL controller is much more robust against changes in the friction coefficients. Another interesting result is that, in general, the PID is more sensitive to changes in the friction in the traction unit, but the RL is more sensitive to changes in the friction of the wheels.

Sometimes it can happen that one wheel gets almost stuck and the other one does not, due to dust or small parts that are on the floor. In these cases, the coefficient of friction of one wheel is larger than the other one. To study this effect, for each trajectory we set F_{sw} to the 80% of

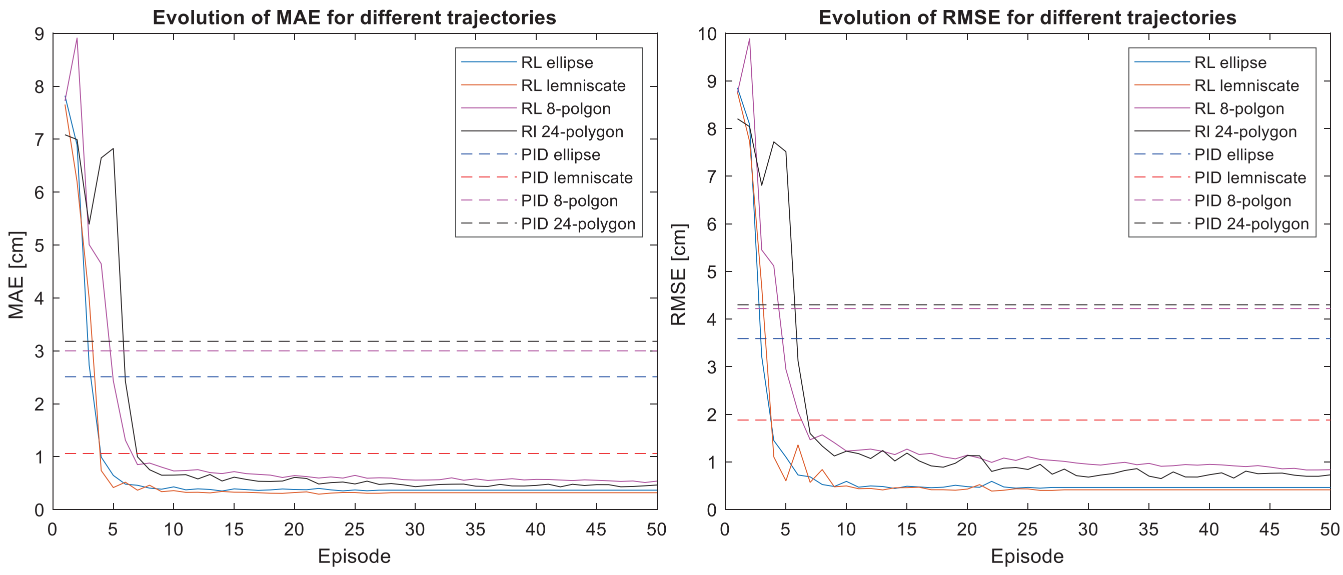

FIGURE 11 Evolution of MAE (left) and RMSE (right) for different trajectories

TABLE 4 Comparison of friction values that make the AGV leave the path.

	F_{sh}		F_{sw}	
Trajectory	**PID**	**RL**	**PID**	**RL**
Ellipse	0.77	**17**	1.16	**4.27**
Lemniscate	0.98	**19**	0.09	**4.75**
8-S Polygon	0.49	**12**	0.6	**4.15**
24-S Polygon	0.49	**17**	0.6	**2.35**

TABLE 5 Comparison of unbalance in the wheel due to friction.

	D_R		D_L	
Trajectory	**PID**	**RL**	**PID**	**RL**
Ellipse	0	**0.25**	0	**0.5**
Lemniscate	**0.5**	0.3	0	**0.35**
8-S polygon	0.05	**0.15**	0.05	**0.5**
24-S polygon	0.05	**0.5**	0.05	**0.15**

the value given in Table 4, and we then define two new coefficients that will help us adjust which part of this friction is applied to the left wheel, K_{sw_L}, and to the right wheel, K_{sw_R}. Both coefficients are related, and its addition is below 1. The distribution of friction coefficient in each wheel can be determined by (45).

$$K_{sw_R} = 1 - K_{sw_L}, F_{sw_L} = K_{sw_L} \cdot 2F_{sw}, F_{sw_R} = K_{sw_R} \cdot 2F_{sw} \tag{45}$$

For each trajectory, we performed several simulations for different values of K_{sw_L}, from 0.5 to 1, with a step of 0.05, until the AGV leaves the path. The distance between 0.5 and the maximum K_{sw_L} that still keeps the AGV on the path is obtained (Table 5, column D_L). The same experiments were performed for different values of K_{sw_R}, from 0.5 to 1 with a step of 0.05, until the AGV leaves the path. Results of the distance between 0.5 and the maximum K_{sw_R} that keeps the AGV on the path is shown in Table 5, column D_R. A larger value in Table 5 indicates a better robustness against unbalance in the wheels due to friction. Therefore, all values in the table must be less or equal to 0.5. This metric is represented by (46).

$$D_L = K_{sw_L} - 0.5, D_R = K_{sw_R} - 0.5 \tag{46}$$

Overall, it can be drawn that the robustness here defined with respect to friction is better when the RL controller is applied, with the only exception of the lemniscate trajectory. Moreover, it is important to note that the total F_{sw} is much larger in the case of RLC as we are applying the 80% of the value of Table 5. For the lemniscate path, the PID copes much better with the left wheel unbalance, and the RL controller only gets a 60%. This variation between left and right wheel unbalance with the PID may be explained since initially the AGV has to make a sharp turn to the right. Thus, it is necessary a much bigger torque in the left wheel, and the corresponding excess in friction of this wheel reduces the effective torque and complicates the manoeuvre. This is practically the only case where the PID gets a better response, in the rest it is zero or very low, only 0.05.

6 | CONCLUSIONS AND FUTURE WORKS

Pre-defined path industrial AGVs, also known as path-follower industrial AGVs, present two coupled control problems that must be simultaneously solved the control of the longitudinal speed and the path tracking. AGVs commonly use a PID controller to adjust the angular velocity to follow the defined trajectory. This normally works well when paths are smooth and large. However, when the dimensions of the circuit are small or more sophisticated trajectories are designed, with abrupt changes of direction, the performance of this conventional controller is not so good and more advanced approaches are needed.

To face these coupled control problems, in this article, we propose a hybrid control scheme that combines RL and traditional linear controllers. Conventional PI controllers are used to maintain the desired speed of each wheel. These speeds are calculated to guarantee the longitudinal reference speed demanded by the application and the angular speed requested by the RL controller. Even more, the RLC is able to learn how to adjust the angular speed to follow the desired path. The RLC feeds the inputs of the PI controllers, this way both control techniques are combined. This hybrid control strategy is applied on a hybrid AGV that is a combination of a tricycle and a differential robot. Based on its kinematics and dynamics, different challenging trajectories have been tested. Guiding error is used in different metrics to quantify the performance of this control strategy and to compare it with the PID control. Extensive simulation results show how the performance of the proposed control approach is much better than the traditional one. Considering all the trajectories, the average improvement provided by the intelligent approach is 81% better regarding the MAE, 79% better with respect to the RMSE, and 84% better in terms of STD.

In addition, quantitative results show that the intelligent RL-based control strategy gives better robustness against changes in the friction coefficient, even when there is a high unbalance between the wheels. Indeed, the friction value that makes the AGV leave the path is around 20 times higher when the intelligent hybrid control approach is applied. This is an interesting and practical result. Another conclusion is that, as expected, paths with polygon shapes are more difficult to follow because of the manoeuvrability of the AGV.

Among other possible future works, we may highlight the application of this control architecture to different AGVs such as forklifts and the definition of new reward strategies. It would be also desirable to implement the controller in the PLC of an AGV prototype.

ACKNOWLEDGMENT

Open access funding enabled and organized by Projekt DEAL.

CONFLICT OF INTEREST

The authors declare that there is no conflict of interest regarding the publication of this paper.

DATA AVAILABILITY STATEMENT

Data sharing not applicable to this article as no datasets were generated or analysed during the current study.

ORCID

J. Enrique Sierra-Garcia https://orcid.org/0000-0001-6088-9954
Matilde Santos https://orcid.org/0000-0003-1993-8368

REFERENCES

Espinosa, F., Santos, C., & Sierra-García, J. E. (2021). Multi-AGV transport of a load: State of art and centralized proposal. *Revista Iberoamericana de Automática e Informática Industrial*, *18*(1), 82–91. https://doi.org/10.4995/riai.2020.12846

Garcia-Auñón, P., Santos-Peñas, M., & de la Cruz Garcia, J. M. (2017). Parameter selection based on fuzzy logic to improve UAV path-following algorithms. *Journal of Applied Logic*, *24*, 62–75.

Guo, X., Ren, Z., Wu, Z., Lai, J., Zeng, D., & Xie, S. (2020, November). A deep reinforcement learning based approach for AGVs path planning. In *2020 Chinese automation congress (CAC)* (pp. 6833–6838). IEEE.

Hu, H., Jia, X., He, Q., Fu, S., & Liu, K. (2020). Deep reinforcement learning based AGVs real-time scheduling with mixed rule for flexible shop floor in industry 4.0. *Computers & Industrial Engineering*, *149*, 106749.

Hu, H., Jia, X., Liu, K., & Sun, B. (2021). Self-adaptive traffic control model with behavior trees and reinforcement learning for AGV in industry 4.0. *IEEE Transactions on Industrial Informatics*, *17*(12), 7968–7979.

Hu, H., Yang, X., Xiao, S., & Wang, F. (2021). Anti-conflict AGV path planning in automated container terminals based on multi-agent reinforcement learning. *International Journal of Production Research*, 1–16. https://doi.org/10.1080/00207543.2021.1998695

Lala, T., Chirla, D. P., & Radac, M. B. (2021). Model reference tracking control solutions for a visual servo system based on a virtual state from unknown dynamics. *Energies*, *15*(1), 267.

Liao, X., Wang, Y., Xuan, Y., & Wu, D. (2020, November). AGV path planning model based on reinforcement learning. In *2020 Chinese automation congress (CAC)* (pp. 6722–6726). IEEE.

Martín, S., Romana, M. G., & Santos, M. (2016). Fuzzy model of vehicle delay to determine the level of service of two-lane roads. *Expert Systems with Applications*, *54*, 48–60.

Martinsen, A. B., & Lekkas, A. M. (2018). Straight-path following for underactuated marine vessels using deep reinforcement learning. *IFAC-PapersOnLine*, *51*(29), 329–334.

Menoyo Larrazabal, J., & Santos Peñas, M. (2016). Intelligent rudder control of an unmanned surface vessel. *Expert Systems with Applications*, *55*, 106–117.

Nakimuli, W., Garcia-Reinoso, J., Sierra-Garcia, J. E., Serrano, P., & Fernández, I. Q. (2021). Deployment and evaluation of an industry 4.0 use case over 5G. *IEEE Communications Magazine*, *59*(7), 14–20.

Radac, M. B., & Precup, R. E. (2016). Three-level hierarchical model-free learning approach to trajectory tracking control. *Engineering Applications of Artificial Intelligence*, *55*, 103–118.

Rubí, B., Morcego, B., & Pérez, R. (2021). Deep reinforcement learning for quadrotor path following with adaptive velocity. *Autonomous Robots*, *45*(1), 119–134.

Sánchez, R., Sierra-García, J. E., & Santos, M. (2022). Modelado de un AGV híbrido triciclo-diferencial. *Revista Iberoamericana de Automática e Informática industrial*, *19*(1), 84–95.

Sierra-García, J. E., & Santos, M. (2020a). Mechatronic modelling of industrial AGVs: A complex system architecture. *Complexity*, *2020*, 21. https://doi.org/10.1155/2020/6687816

Sierra-García, J. E., & Santos, M. (2020b, September). Control of industrial AGV based on reinforcement learning. In *International workshop on soft computing models in industrial and environmental applications* (pp. 647–656). Springer.

Sierra-García, J. E., & Santos, M. (2020c). Exploring reward strategies for wind turbine pitch control by reinforcement learning. *Applied Sciences*, *10*(21), 7462.

Sierra-García, J. E., & Santos, M. (2021a). Neural networks and reinforcement learning in wind turbine control. *Revista Iberoamericana de Automática e Informática Industrial*, *18*(4), 327–335. https://doi.org/10.4995/riai.2021.16111

Sierra-García, J. E., & Santos, M. (2021b). Lookup table and neural network hybrid strategy for wind turbine pitch control. *Sustainability*, *13*(6), 3235.

Sun, Y., & Li, H. (2020, October). An end-to-end reinforcement learning method for automated guided vehicle path planning. In *International symposium on artificial intelligence and robotics 2020* (Vol. 11574, p. 115740X). International Society for Optics and Photonics.

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT Press.

Trojaola, I., Elorza, I., Irigoyen, E., Pujana-Arrese, A., & Calleja, C. (2020). The effect of iterative learning control on the force control of a hydraulic cushion. *Logic Journal of the IGPL, 30(2), 214–226*

Van, N. T. T., Tien, N. M., Cuong, N. M., Duyen, H. T. K., & Duy, N. D. (2021). Constructing an intelligent navigation system for autonomous mobile robot based on deep reinforcement learning. In *Soft computing: Biomedical and related applications* (pp. 251–261). Springer.

Xue, T., Zeng, P., & Yu, H. (2018, February). A reinforcement learning method for multi-AGV scheduling in manufacturing. In *2018 IEEE international conference on industrial technology (ICIT)* (pp. 1557–1561). IEEE.

Yeh, Y. L., & Yang, P. K. (2021). Design and comparison of reinforcement-learning-based time-varying PID controllers with gain-scheduled actions. *Machines*, *9*(12), 319.

Yu, W., Liu, J., & Zhou, J. (2021). A novel automated guided vehicle (AGV) remote path planning based on RLACA algorithm in 5G environment. *Journal of Web Engineering*, 20(8), 2491–2520.

Zamora-Cadenas, L., Velez, I., & Sierra-Garcia, J. E. (2021). UWB-based safety system for autonomous guided vehicles without hardware on the infrastructure. *IEEE Access*, *9*, 96430–96443.

Zhang, Q., Lin, J., Sha, Q., He, B., & Li, G. (2020). Deep interactive reinforcement learning for path following of autonomous underwater vehicle. *IEEE Access*, *8*, 24258–24268.

Zhao, J., Yang, C., Gao, W., & Zhou, L. (2022). Reinforcement learning and optimal setpoint tracking control of linear systems with external disturbances. In *IEEE Transactions on Industrial Informatics.*, In press. https://doi.org/10.1109/TII.2022.3151797.

Zhu, W., Guo, X., Fang, Y., & Zhang, X. (2019). A path-integral-based reinforcement learning algorithm for path following of an autoassembly mobile robot. In *IEEE Transactions on Neural Networks and Learning Systems*, *31*(11), 4487–4499.

AUTHOR BIOGRAPHIES

J. Enrique Sierra-Garcia was born in Burgos, Spain. He received his M.Sc. degrees in Electronics and Telecommunications from the University of Valladolid (UVA) in 2007 and 2015 respectively, his M.Sc degree in Control Engineering from the National University for Distance Education (UNED) in 2014, and his Ph.D in Computer Science in 2019. Since 2012 he has been with the University of Burgos, where he is currently a Lecturer in System Engineering and Automatic Control in the Department of Electromechanical Engineering. He is the founder of the Joint Research Unit ASTI-UBU on Autonomous Vehicles, Mobile Robotics and AGVs. He has been principal researcher in more than 10 research projects related with mobile robotics. His major research interests are: Intelligent Control, Robotics, Autonomous Guided Vehicles, Modeling, Simulation, Wind energy.

Matilde Santos was born in Madrid, Spain. She received her B.Sc. and M.Sc. degrees in Physics (Computer Engineering) and her Ph.D in Physics from the University Complutense of Madrid (UCM). She is currently Full Professor in System Engineering and Automatic Control. She is member of the European Academy of Sciences and Arts. She has published many papers in international scientific journals and several books and book chapters. She has supervised more than 12 Ph.Ds. She has worked on several national, European and international research projects, leading some of them. She currently serves as member of the editorial board of prestigious journals and she is editor-in-chief assistant of one of them. Her major research interests are: Intelligent Control (fuzzy and neuro-fuzzy), Pattern Recognition, Modelling and Simulation, Wind energy.

How to cite this article: Sierra-Garcia, J. E., & Santos, M. (2022). Combining reinforcement learning and conventional control to improve automatic guided vehicles tracking of complex trajectories. *Expert Systems*, e13076. https://doi.org/10.1111/exsy.13076