IGPL

AGV Fuzzy Control Optimized by Genetic Algorithms

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

[1] Electromechanical Engineering Department, University of Burgos, 09006-Burgos, Spain.

[2] Institute of Knowledge Technology, Complutense University of Madrid, 28040-Madrid, Spain.

*Corresponding author: J. Enrique Sierra-García; jesierra@ubu.es

Abstract

Automated Guided Vehicles (AGV) are an essential element of transport in industry 4.0. Although they may seem simple systems in terms of their kinematics, their dynamics are very complex and it requires robust and efficient controllers for their routes in the workspaces. In this paper, we present the design and implementation of an intelligent controller of a hybrid AGV for trajectory tracking based on fuzzy logic. The control scheme includes proportional-integral (PI) speed regulators for each wheel, and the intelligent control for the path following. In addition, genetic algorithms (GA) have been used to optimize the speed control strategy, aiming at improving efficiency and saving energy, and also the fuzzy controller has been enhanced with genetic algorithms. The latter uses a cost function that first maximizes the time in the circuit and then minimizes the guiding error. It has been validated on the mathematical model of a commercial hybrid AGV that merges tricycle and differential robot components. This model not only considers the kinematics and dynamics equations of the vehicle but also the impact of friction. The performance of the intelligent control strategy is compared with an optimized PID controller for four different paths. Simulation results in terms of the mean absolute error following those trajectories show that the intelligent controller reduces the error in a percentage that goes from 65,7% for the most complicated paths to 88,9 % for the simplest trajectories.

Keywords: intelligent control, fuzzy control, genetic algorithms, automated guided vehicle (AGV), path following, industry 4.0

1. Introduction

The use of Automated Guided Vehicles (AGVs) in industry is on the rise. These vehicles are an essential element of transport in industry 4.0 and help improve production. AGVs are industrial unmanned transport vehicles that are mainly used to automate logistics and production lines in factories and industries. They allow operations in a much more flexible way than traditional conveyor belts and help reduce quality errors in industrial processes [1].

Although these vehicles may seem simple systems in terms of their kinematics, their dynamics are very complex and it requires robust and efficient controllers for their routes in the workspaces [2-3]. AGVs are equipped with a guiding sensor that measures the distance between the AGV and a reference trajectory marked on the floor. Depending on the sensor that is used the reference can be implemented by a magnetic tape, if the sensor is a magnetic antenna, a painted line if the sensor is a camera, or a buried wire if the sensor is inductive. In the new

navigation systems like SLAM-based ones, the trajectory reference is not marked on the floor, but it is virtual, and it is stored in the system [4]. Regardless the sensor that is used, a controller must be used to ensure the AGV follows the desired trajectory at a target speed to fit production times [5].

Despite AGVs are widely accepted in the industry and have a relevant impact on productivity of enterprises, scarce research has been carried out on optimizing the control of these systems. Intelligent control techniques have been successfully applied to model and control complex engineering systems [6-8]. The combination of intelligent control techniques such as fuzzy logic and optimization techniques such as genetic algorithms can contribute to improve their performance. There are some previous works that use fuzzy logic to design AGV controllers. In [9] a fuzzy controller was designed using parallel distributed compensation with three state kinematic error model to minimize the error tracking. The AGV was designed using differential drive for small material transportation handling in manufacture area productions. Li et al, developed a tracking control algorithm based on fuzzy logic for batch-feeding AGV. The tracking control algorithm combined the traditional PID control algorithm with fuzzy logic and adopted fuzzy adaptive control for the proportional gain [10]. In [11] a design of omnidirectional automatic guided vehicle based on a hub motor is presented, and a joint controller for path tracking is proposed. The control includes a fuzzy system and a multi-step predictive optimal controller. Silvirianti et al design and implement an AGV speed control system to cope with load and contour variations using a fuzzy inference system integrated with a PID [12]. In a recent work, Abajo et al., optimize the control gains of a differential AGV by genetic algorithms [13]. However, none of these previous works on AGV control use genetic algorithms to optimize the design of the fuzzy controller neither apply it to a hybrid AGV.

Genetic algorithms and fuzzy logic, independently and in combination, have been used in a closely related application, the control of fleets of AGVs. To mention some examples, in [14] a multi-objective mathematical model was developed to optimize the task scheduling of AGVs. Two adaptive genetic algorithms and a multi-adaptive genetic algorithm were applied, taking into account the charging task and the AGV changeable speed to minimize makespan, the number of AGVs used, and the electricity consumed. Goli et al., developed a fuzzy mixed integer linear programming model for cell formation problems including the scheduling of parts within cells in a cellular manufacturing system where several automated guided vehicles were in charge of transferring the exceptional parts [15]. In [16] a computer-controlled material handling model using fuzzy logic and genetic algorithms is developed. The set of stations requesting transportation service was determined by means of fuzzy logic, while the sequence of stations in a loop was optimized with genetic algorithms. Finally, Zacharia et al., proposed an AGV routing and motion planning in a flexible manufacturing system using a fuzzy-based genetic algorithm [14].

Therefore, in view of the necessity of improving the tracking control of these unmanned vehicles, in this work a control strategy that combines fuzzy logic and genetic algorithms is proposed. The mathematical model of a hybrid AGV that merges the behavior of a tricycle, and a differential robot is used. This model represents a real commercial AGV that has a body that moves as a tricycle and a traction unit that is a differential robot. These elements are connected by a rotation joint. The model considers not only the kinematics and the dynamics of the AGV but also the effect of the friction. This non-linearity makes the representation closer to the reality. The control scheme consists, firstly, in an optimized speed PI control where genetic algorithms have been used to obtain the tuning parameters of the regulator. Once the vehicle velocity is optimized, a trajectory tracking fuzzy control -optimized by genetic algorithms- is proposed. In this case, the cost function is designed to address two objectives: preventing the AGV from leaving the path and minimizing the guiding error. Thus, firstly the objective function maximizes the time that the AGV travels without leaving the path, this way the AGV completes the route successfully. Once the route is finished, the function focuses on minimizing the guiding error. The optimized intelligent control approach is compared with an optimized PID with satisfactory results. Simulation experiments with four different trajectories confirm the validity of the control proposal.

The rest of the paper is structured as follows. The mathematical model of the AGV is briefly described in section 2. In section 3, the intelligent control architecture is designed and explained. The outcomes of the simulations are discussed in section 4. Conclusions and future work are presented at the end of the manuscript.

2. Mathematical model of the AGV

The suggested control approach was implemented using a mathematical model of a hybrid AGV that combines tricycle and differential movements. In the industry, this form of tow AGV is commonly employed. The traction unit acts as a mobile differential robot. This part is connected to the AGV body via a pivoting axle. The other part of the vehicle is the AGV body, which has the kinematics of a tricycle except that instead of a wheel, the traction unit controls the steering. The diagram of the hybrid AGV is shown in Fig. 1.

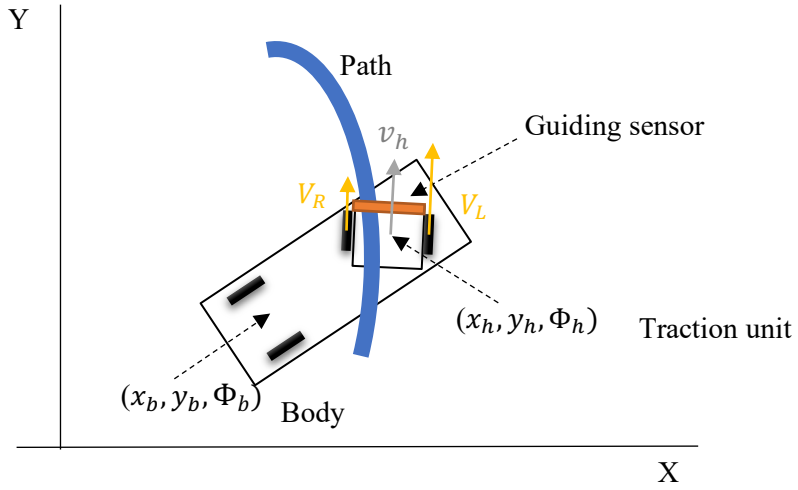

Fig. 1. Traction unit and body of the AGV

Different navigation systems or guiding sensors can be installed on this sort of AGV. A magnetic sensor situated in front of the traction unit was used in this study. A magnetic tape on the floor is used to trace the path to be followed. This magnetic tape is durable, requiring far less maintenance than optical systems such as QR codes or painted lines. The magnetic sensor measures the distance between the magnetic tape center and the driving unit center, also known as the guiding error.

The mechanical model of the AGV is described by equations (1-12). These equations are further explained in [2, 17]. The dynamic is described by (1-4).

The torques produced by the motors of each wheel are M_r and M_l, in Nm. These torques are reduced by the friction torques in the axles of the wheels, F_{sw_L} and F_{sw_R}, obtaining the effective torques, M_{er} and M_{el} (1).

For the sake of clarity, the effect of the friction with the floor and the aerodynamic friction, f_{rT}, have been included in the equation of the translational dynamic.

$$m_T \cdot \frac{R_h}{2}(\ddot{\theta}_R + \ddot{\theta}_L) = \frac{(M_{er} + M_{el})}{2R_h} - f_{rT}$$

In the same line, the rotational friction of the pivoting axle of the traction unit, f_{rR},. has been included in the equation of the rotational dynamic.

$$\frac{I_h \cdot R_h}{L_h}(\ddot{\theta}_R - \ddot{\theta}_L) = \frac{(M_{er} - M_{el})L_h}{2R_h} - f_{rR}$$

These statements can be expressed as,

$$M_{er} = M_r - F_{sw_R} \cdot \text{sign}(\dot{\theta}_R), M_{el} = M_l - F_{sw_L} \cdot \text{sign}(\dot{\theta}_L), \quad (1)$$

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

Where 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; L_h (m) is the distance between the wheels in 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); $\ddot{\theta}_r$ (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. Finally, $sign()$ is the sign function.

The translational and rotational friction forces are given by (3-4).

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

$$f_{rR} = F_{vh} \cdot \dot{\Phi}_h + F_{sh} \cdot \text{sign}(\dot{\Phi}_h), \quad (4)$$

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. δ_{air} (kg/m^3) is the density of the air, and S_{AGV} (m^2) is the front surface of the AGV.

That is, as it is possible to observe from the previous equations, the model considers the effect of the friction this vehicle experiments: the static and viscous friction in the mechanical joint between the traction unit and the body, f_{rR}; the friction in the axle of the wheels, F_{sw_R} and F_{sw_L}; and the ground friction and the aerodynamic effect, f_{rT}. On the other hand, the kinematic equations and the guiding sensor are given by (5-12).

$$V_L = R_h \cdot \dot{\theta}_L, V_R = R_h \cdot \dot{\theta}_R, \quad (5)$$

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

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

$$v_h = \sqrt{{\dot{x}_h}^2 + {\dot{y}_h}^2} = \frac{V_L+V_R}{2}, \quad (8)$$

$$\gamma = \Phi_h - \Phi_b, \gamma_{min} \le \gamma \le \gamma_{max}, \quad (9)$$

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

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

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

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_b (m) is the distance between the rear wheels and the center of the traction unit. 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, that otherwise could have been considered a disturbance, does not affect the movement.

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

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
$[Kp_v, Ki_v]$	PI gains for velocity control of wheels	$[2, 0.1]$
$[Kp_\Phi, Kd_\Phi, Ki_\Phi]$	PID gains for angular velocity control of the traction unit	$[9.8, 1, 0.1]$

3. Control architecture

As said, the control architecture consists of two controllers, both of them optimized using genetic algorithms. The first one is the speed PI control for each wheel. The second one is the tracking controller, based on fuzzy logic.

3.1 Optimized cruise velocity control

Fig. 2 shows the architecture of the speed control and how it is optimized by genetic algorithms. On the left of the figure there is a module (orange square) that, considering the inverse kinematics of the AGV, transforms the angular velocity and the longitudinal velocity reference into a speed reference for each wheel, $v_{L_{REF}}$ and $v_{R_{REF}}$. To optimize the cruise velocity control, the reference for the angular velocity is set to zero, and the reference for the longitudinal velocity $v_{t_{REF}}$ follows a profile designed ad-hoc which covers all working ranges. These speed references feed the reference inputs of the PI controller of each wheel. These PI controllers generate the torque control signals, M_L and M_R. The current speed of the wheels, v_L and v_R, are the feedback inputs of the PI controllers and are also used to evaluate the cost function, c_f.

At each iteration of the optimization process, the genetic algorithm receives the value of the cost function obtained in the previous iteration and generates a set of values of the tuning parameters of the PI controllers, $[Kp_{x1}, Kp_{y1}, Kp_{x2}, Kp_{y2}, Ki_v]$. The parameter Ki_v is the integral gain of the PI controllers, and the other four values are used to obtain the proportional gain of the PI controllers. When the speed reference is low the error tends to be small; thus, it is necessary a larger Kp to generate the minimum torque necessary to compensate the friction. However, this larger Kp is counterproductive if the speed reference is also large, as in this case the errors tend to be bigger and a large Kp produces oscillations. To solve it, the controller adjusts the Kp in function of the target speed. If the reference is small, the gain is large and vice versa. The cost function minimizes the mean absolute error between the longitudinal velocity, v_t, and its reference, $v_{t_{REF}}$.

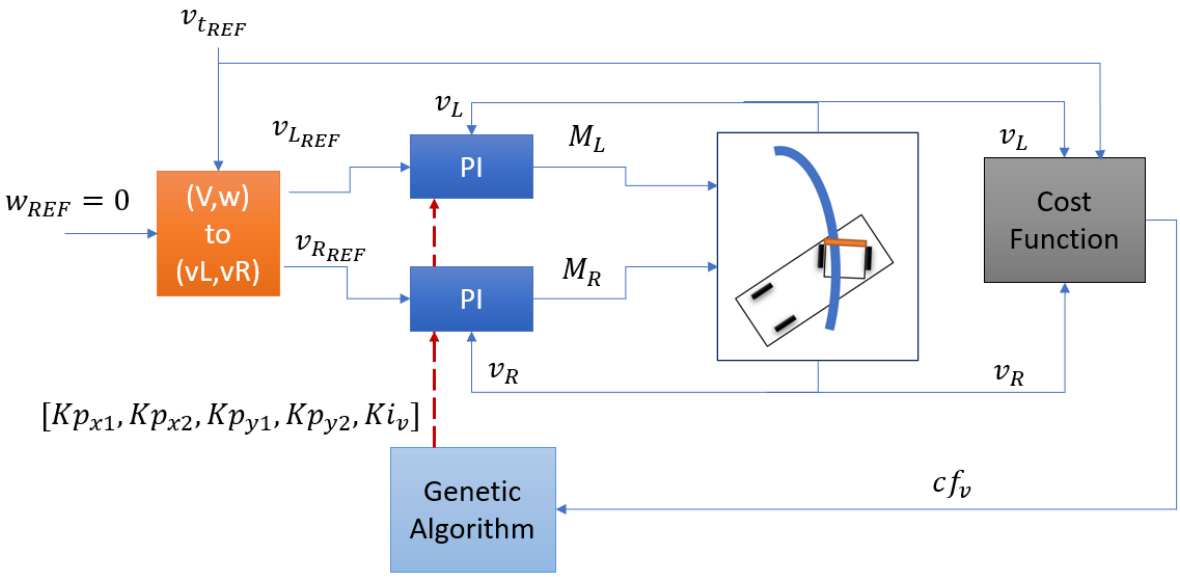

Fig. 2. Architecture to optimize the velocity control

This control strategy can be formalized by equations (13-24):

$$v_{L_{REF}}(t_i) = v_{t_{REF}}(t_i) - \frac{w_{REF}(t_i)\cdot L_h}{2}, \tag{13}$$

$$v_{R_{REF}}(t_i) = v_{t_{REF}}(t_i) + \frac{w_{REF}(t_i)\cdot L_h}{2}, \tag{14}$$

$$err_{vL}(t_i) = v_{L_{REF}}(t_i) - v_L(t_{i-1}), \tag{15}$$

$$err_{vR}(t_i) = v_{R_{REF}}(t_i) - v_R(t_{i-1}), \tag{16}$$

$$v_t(t_i) = \frac{1}{2}(v_L(t_{i-1}) + v_R(t_{i-1})), \tag{17}$$

$$Kp_v(t_i) = \begin{cases} Kp_{y1}(n) & v_{t_{REF}}(t_i) < Kp_{x1}(n) \\ \frac{Kp_{y2}(n)-Kp_{y1}(n)}{Kp_{x2}(n)-Kp_{x1}(n)}(v_{t_{REF}}(t_i) - Kp_{x1}(n)) + Kp_{y1}(n), & Kp_{x1}(n) < v_{t_{REF}}(t_i) < Kp_{x2}(n) \\ Kp_{y2}(n) & Kp_{x2}(n) < v_{t_{REF}}(t_i) \end{cases}, \tag{18}$$

$$M_L(t_i) = Kp_v(t_i) \cdot err_{vL}(t_i) + Ki_v(n) \cdot \int err_{vL}(t_i)dt, \tag{19}$$

$$M_R(t_i) = Kp_v(t_i) \cdot err_{vR}(t_i) + Ki_v(n) \cdot \int err_{vR}(t_i)dt \tag{20}$$

$$n = \begin{cases} n+1 & t_i = T_{sim} \\ n & othercase \end{cases} \tag{21}$$

$$cf_v(n) = \begin{cases} \frac{1}{T_{sim}}\sum_i |v_{t_{REF}}(t_i) - v_t(t_i)| & t_i = T_{sim} \\ c_f(n-1) & othercase \end{cases} \tag{22}$$

$$\begin{bmatrix} Kp_{x1}(n) \\ Kp_{x2}(n) \\ Kp_{y1}(n) \\ Kp_{y2}(n) \\ Ki_v(n) \end{bmatrix} = \begin{cases} ga_v(cf_v) & t_i = T_{sim} \\ \begin{bmatrix} Kp_{x1}(n-1) \\ Kp_{x2}(n-1) \\ Kp_{y1}(n-1) \\ Kp_{y2}(n-1) \\ Ki_v(n-1) \end{bmatrix} & othercase \end{cases} \tag{23}$$

$$t_i = \begin{cases} 0 & t_i = T_{sim} \\ t_{i-1} + Tc & othercase \end{cases} \tag{24}$$

Where n is the iteration of the optimization process, T_{sim} is the duration of the simulation, and $ga_v: \mathbb{R} \rightarrow \mathbb{R}^5$ represents the function implemented by the genetic algorithm.

The individual in the genetic algorithm is the set of parameters $[Kp_{x1}, Kp_{y1}, Kp_{x2}, Kp_{y2}, Ki_v]$. These tuning gains are subjected to the following constraints:

- $Ki_v \geq 0$
- $K_{px1} \geq 0, K_{px2} \geq 0, K_{py1} \geq 0, K_{py2} \geq 0$
- $K_{px1} \leq K_{px2}, K_{py1} \geq K_{py2}$

The configuration of the genetic algorithm is as follows. The size of the population is 50 individuals, the crossover fraction is set to 0.8, the mutation rate is 0.2, and the elite size is 5%. The genetic algorithm runs during 1000 iterations. The fitness function is given by the equation (22). The individuals are randomly initialized. This configuration was selected by trial and error, as recommended for optimization problems with 5 parameters.

3.2 Optimization of the trajectory tracking control

Once the cruise velocity control has been optimized, the trajectory tracking control is designed based on fuzzy logic and it is optimized using genetic algorithms. Fig. 3 shows the architecture of this controller. This tracking control generates the reference for the angular velocity, w_{REF}, used as input in the velocity control shown in Fig. 2. The core component in the control architecture is the fuzzy logic controller (FLC), a 0-type Sugeno fuzzy inference system with three inputs: the guiding error measured by the sensor, err_{gui}, its derivative and its integral. Three gaussian fuzzy sets that correspond to labels Negative, Zero and Positive have been assigned to each of the inputs. The output of the FLC is the reference of the angular velocity, w_{REF}. The fuzzy rules are associated with three possible values [w_{MIN}, 0, w_{MAX}]. This controller is formalized by (25-27).

$$F_{j,k} = e^{-\frac{(x_j - \mu_{jk})^2}{2\sigma_{jk}^2}}, j = 1, \dots N_I, k = 1, \dots N_F \tag{25}$$

$$w_i = \prod_l F_l, \quad l \in R_i, \ i = 1, \dots N_R \tag{26}$$

$$w_{REF}(t_i) = \frac{\sum w_i \cdot z_i}{\sum w_i}, i = 1, \dots N_R \tag{27}$$

Where x_j is the value of the input j, $F_{j,k}$ represents the value (degree) of the membership function k of the input j, $N_I = 3$ is the number of fuzzy inputs of the controller, $N_F = 3$ is the number of fuzzy sets per input, and $N_R = N_I^{N_F} = 27$ is the total number of fuzzy rules. Each *i*-rule R_i represents the set of pairs (input-fuzzy set); to better explain this notation, the set R_1 is [(err_{gui}, NEG),($\dot{err}_{gui}, NEG$), ($\int err_{gui}, NEG$)]. Finally, $z_i \in [w_{MIN}, 0, w_{MAX}]$ represents the output value of the *i*-rule. This way the FLC obtains an output value between w_{MIN} and w_{MAX}, considering the guiding error, its derivative, and its integral value. The values [w_{MIN}, w_{MAX}] have been set to [−1.6, 1.6] rad/s. These values are selected as follows. In this AGV, the typical steering range is from $-\frac{\pi}{4}$ to $\frac{\pi}{4}$. To cover this movement range in 1 second, we need a speed of $\frac{\pi}{2}$ rad/s. This is approximately 1.6 rad/s.

The 27 fuzzy rules have been grouped according to the different possible cases, and consequently the number of rules is reduced when the input integral error does not affect the outcome. The following complete fuzzy rule base is obtained:

- If $err_{gui} = NEG$ and $\dot{err}_{gui} = NEG$ then w_{MIN}
- If $err_{gui} = NEG$ and $\dot{err}_{gui} = ZERO$ then w_{MIN}
- If $err_{gui} = NEG$ and $\dot{err}_{gui} = POS$ then 0
- If $err_{gui} = ZERO$ and $\dot{err}_{gui} = NEG$ then w_{MIN}
- If $err_{gui} = ZERO$ and $\dot{err}_{gui} = ZERO$ and $\int err_{gui} = NEG$ then w_{MIN}
- If $err_{gui} = ZERO$ and $\dot{err}_{gui} = ZERO$ and $\int err_{gui} = ZERO$ then 0
- If $err_{gui} = ZERO$ and $\dot{err}_{gui} = ZERO$ and $\int err_{gui} = POS$ then w_{MAX}
- If $err_{gui} = ZERO$ and $\dot{err}_{gui} = POS$ then w_{MAX}

- If $err_{gui} = POS$ and $\dot{err}_{gui} = NEG$ then 0
- If $err_{gui} = POS$ and $\dot{err}_{gui} = ZERO$ then w_{MAX}
- If $err_{gui} = POS$ and $\dot{err}_{gui} = POS$ then w_{MAX}

At each iteration of the optimization process, the genetic algorithm receives the value of the cost function, cf_{gui}, which calculates the centers, $\mu \in \mathbb{R}^9$, and the widths, $\sigma \in \mathbb{R}^9$, of the 9 fuzzy membership functions of the FLC. There are 9 fuzzy membership functions to be optimized as there are 3 inputs and each input has 3 fuzzy sets assigned. That is, the individual of the genetic algorithm is formed by a set of 18 values $[\mu \in \mathbb{R}^9, \sigma \in \mathbb{R}^9]$. The cost function is calculated by (28).

$$cf_{gui} = \begin{cases} 1 + (T_{sim} - t_i)/T_{sim} & |err_{gui}(t_i)| > err_{gui_{MAX}} \\ \frac{1}{T_{sim} \cdot err_{gui_{MAX}}} \sum_i |err_{gui}(t_i)| & |err_{gui}(t_i)| < err_{gui_{MAX}} \wedge t_i = T_{sim} \end{cases} \quad (28)$$

The cost function is piecewise defined, and it has two parts. Starting for the second line of equation (28), this expression is normalized as it is divided by $err_{gui_{MAX}}$, so its maximum value is 1, and thus the range is [0, 1]. On the other hand, regarding the first line of equation (28), the term added to 1, i.e., $(T_{sim} - t_i)/T_{sim}$, is also normalized to [0, 1]. Adding the term 1 moves this range from [0, 1] to [1, 2]. Thus, the range of both lines are clearly separated, with no overlap between them.

This way, the optimizer first works on the completion of the simulation without leaving the path, that corresponds to the first expression of equation (8). Once the AGV completes the simulation, the value of the cost function is below 1 and the optimizer focuses on reducing the guiding error (line 2 of equation 28).

The reasoning under this cost function is the following. If the absolute value of the guiding error, $|err_{gui}(t_i)|$, surpasses the maximum guiding error, $err_{gui_{MAX}}$, the guiding sensor is no longer able to perceive the path and it means that the AGV has lost the trajectory. In this case, that corresponds to the first expression in (28), the fitness function minimizes the error between the expected duration of the simulation, T_{sim}, and the period until the AGV leaves the trajectory, t_i. This way, it maximizes the time that the AGV travels without leaving the path until the AGV completes the route successfully, during the time T_{sim}. If the AGV completes the simulation successfully, that is $|err_{gui}(t_i)| < err_{gui_{MAX}}$ and $t_i = T_{sim}$, the cost function uses then the MAE of the guiding error, so the smaller MAE, the smaller the fitness value and thus, the error. The maximum error $err_{gui_{MAX}}$ is 0.4 m.

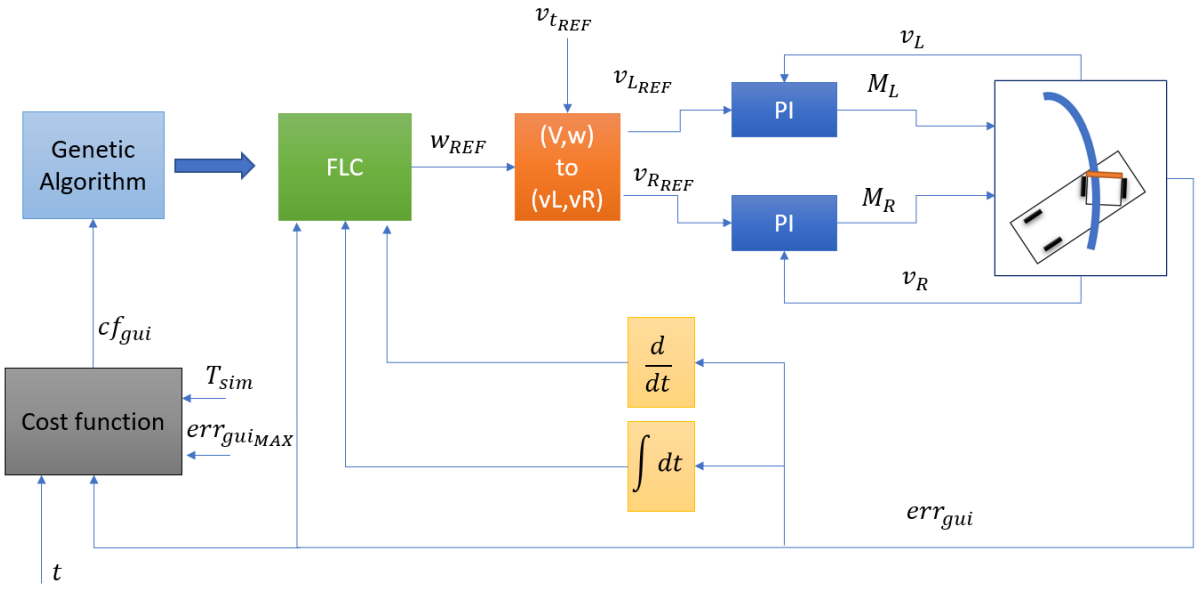

Fig. 3. Strategy to optimize the trajectory tracking control

The individual of the genetic algorithm is the set of values $[\mu \in \mathbb{R}^9, \sigma \in \mathbb{R}^9]$. These values are subjected to the following constraints:

- $|\mu_{jk}| \geq 0.5, j = 1, \dots N_I, k = 1, \dots N_F$
- $|\sigma_{jk}| > 0,\ j = 1, \dots N_I, k = 1, \dots N_F$
- $|\mu_{j,1}| < |\mu_{j,2}| < |\mu_{j,3}|, j = 1, \dots N_I$

The configuration of the genetic algorithm is as follows. The size of the population is 200 individuals, the crossover fraction is 0.8, the mutation rate is 0.2, and the elite size is 5%. The genetic algorithm runs during 1000 iterations. The fitness function is given by the equation (28). The individuals are randomly initialized.

The performance of this optimization process can be described by the following algorithm:

$[\mu, \sigma] \leftarrow rand()$

For n=1 **to** 1000

While $t_i < T_{sim} \wedge |err_{gui}(t_i)| < err_{gui_{MAX}}$

$[err_{gui}(t_i), t_i] \leftarrow simNextStep(\mu, \sigma);$

end

$$cf_{gui} = \begin{cases} 1 + (T_{sim} - t_i)/T_{sim} & |err_{gui}(t_i)| > err_{gui_{MAX}} \\ \dfrac{1}{T_{sim} \cdot err_{gui_{MAX}}} \sum_i |err_{gui}(t_i)| & |err_{gui}(t_i)| < err_{gui_{MAX}} \bigwedge t_i = T_{sim} \end{cases}$$

$[\mu, \sigma] \leftarrow ga(cf_{gui})$

end

4. Results and Discussion

This intelligent control approach has been tested in simulation using the software Matlab/ Simulink on the model of a hybrid AGV described previously. The parameters of the model were listed in Table 1. Each simulation ends when the AGV leaves the circuit or when the maximum simulation time, set to 100 s, is reached. The sample time, T_s, was automatically adjusted during the simulation to reduce the discretization error, being its maximum value 25 ms. On the other hand, the control period was set to 25 ms.

The standard trajectory controller in these vehicles is a PID regulator, that will be used for comparison purposes. The PID controller follows equation (29).

$$w_{REF}(t_i) = K_p \cdot err_{gui}(t_i) + K_d \cdot \frac{d}{dt} err_{gui}(t_i) + K_i \cdot \int err_{gui} \qquad (29)$$

Where $[K_p, K_d, K_i]$ are the tuning parameters of the PID, and the control action is the angular speed reference, w_{REF}. We have evaluated the control strategy with four different trajectories, an ellipse, a lemniscate, and two closed polylines. For each trajectory we have used the genetic algorithm to find the optimum tuning parameters of the FLC and the PID controllers.

4.1 Optimization of the velocity control

In order to tune the speed controller so to achieve good performance for different speed values, a speed profile defined as a train of pulses of different amplitudes was designed. This way, a wide range of speeds is used. This signal is divided into two parts. In the first one, the first 100 s, there is an ascending stair of steps, from 0.1m/s

to 0.8m/s amplitude followed by a descending ramp of steps down to 0m/s (see Fig. 4, from 0 to 100 s). In the second part of this speed signal, there are several trains of pulses with different amplitudes, from 0.1 m/s to 0.8 m/s, to represent a variable speed. In this second part, the duration of each step is 5 s. Each step is followed by another step with the same duration but inverse amplitude. Once the steps reach 0.8 m/s, a new train of steps is carried out and the initial value is incremented (see Fig. 4). The reason to design this pattern is to cover all possible working points and all possible amplitudes of the input. Considering this speed reference, the genetic algorithm searches the best set of parameters $[Kp_{x1}, Kp_{y1}, Kp_{x2}, Kp_{y2}, Ki_v]$ to minimize the speed error.

Once the speed controller is optimized with this speed profile, the trajectory controller is tested with a different speed profile. The longitudinal reference speed of the AGV is a sinusoidal signal with amplitude from 0 to 0.7m/s and average value of 0.35 m/s. A speed of 0.7 m/s is a typical value for industrial applications with these AGVs.

In Fig. 4a), this speed profile is shown (blue line) with the AGV longitudinal speed (red line) once it has been trained. It is possible to see how the controller makes the AGV follow correctly all the steps. The variable vFoll (m/s) is the longitudinal velocity of the AGV. It is noteworthy to remind that the equations of speed controller are given by (13-20) and the control architecture of Fig. 2 has been used to obtain these results. In this case, the optimum values of the gains $[Kp_{x1}, Kp_{y1}, Kp_{x2}, Kp_{y2}, Ki_v]$ are [0.5569907, 10.41152, 1.452401, 4.729725, 0.003572956].

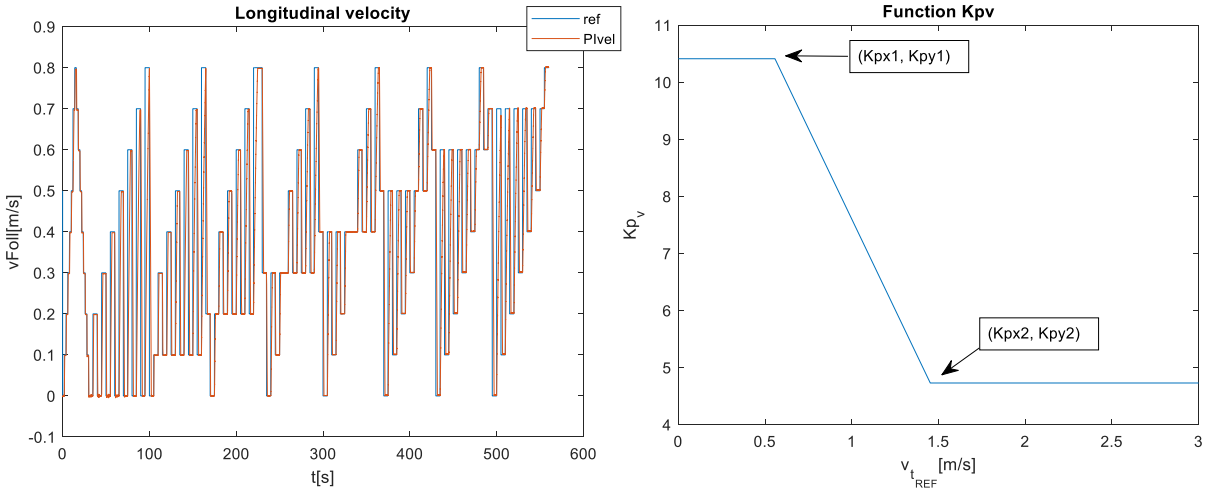

Fig. 4. a) Longitudinal velocity tracking for a train of pulses; b) Function Kpv

Figure 4b) shows the function $K_{pv}(v_{t_{REF}})$ obtained after the optimization process. As it is possible to see, it has two different thresholds, Kp_{x1} and Kp_{x2}. When the reference speed is below Kp_{x1}, the K_{pv} is constant and its value is set to Kp_{y1}. On the other hand, when it is above Kp_{x2}, the K_{pv} is set to Kp_{y2}. Finally when the reference speed is between these two values, the K_{pv} varies linearly.

4.2 Optimization of the trajectory control

In the following experiments we are going to compare the performance of the optimized intelligent control proposal and the optimized PID to track four different trajectories: ellipse, lemniscate, octagon, and 24-sided polygons. The longitudinal reference speed of the AGV, $v_{t_{REF}}$, is a sinusoidal signal with amplitude from 0 to 0.7m/s and average value of 0.35 m/s. A speed of 0.7 m/s is typical value for industrial applications with these AGVs. The figures below show the performance obtained by both controllers once they have been optimized by the genetic algorithms.

Figures 5-8 illustrate the trajectory followed by the AGV traction unit (left) and the guiding error (right), when the PID (blue line) and the fuzzy-based (red line) controllers are applied. The reference path is shown in black.

In the elliptical trajectory the AGV starts at point (0, 0.7), and moves clockwise. It is possible to observe in Fig. 5, left, how the FLC follows very well the path; indeed, the FLC and the reference lines are almost overlapped.

In the case of the PID there is a remarkable difference between the trajectory and the reference. This performance is also shown in the error signal, where the error with the FLC is much smaller than with the PID. The latter even presents large peaks, over 12 cm, at the beginning of the trajectory, around t=5s.

In the lemniscate trajectory, shown in Fig. 6, the AGV starts at point (2.02, 1.17), and moves clockwise. In this case both, the FLC and the PID, are quite close the reference and they are practically overlap with it. However, the error signal allows us to identify some differences. Again, the error with the FLC is smaller than with the PID. At the beginning of the trajectory, as the AGV starts with an angle of 0º and it needs a large correction to follow the curve, both present an error peak. After this peak the error decreases and remains small with the FLC, but with the PID new peaks, though smaller, appear. In fact, there is a relatively large error in the trajectory around the point (3, 0.5), shown by the blue line at around t = 5 s, meaning that the AGV goes away from the reference.

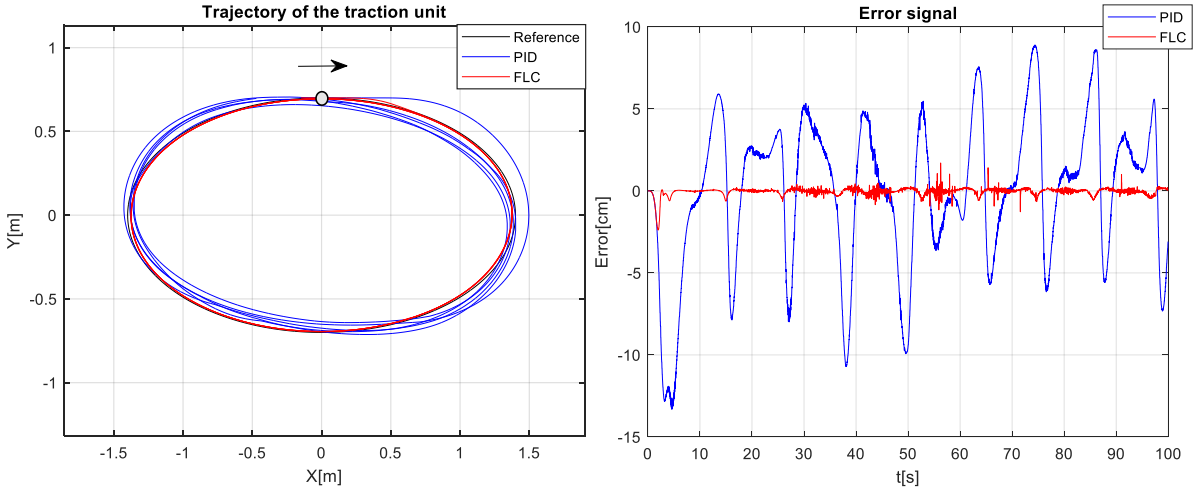

Fig. 5. Elliptical trajectory (left) and corresponding guiding error (right)

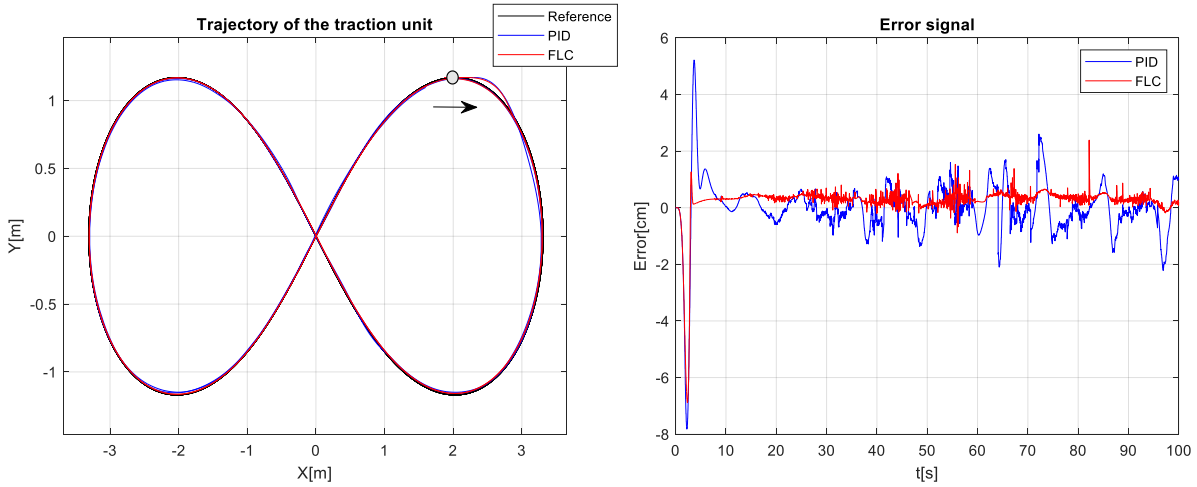

Fig. 6. Lemniscate trajectory (left) and corresponding guiding error (right)

The trajectory in Fig. 7 is an 8-sided polygon, where the starting point is (0,0) and the AGV travels counterclockwise. In this case both controllers track quite well the reference, but there are many more abrupt peaks with the PID, though sometimes are larger with the fuzzy-based controller. These oscillations can also be observed in the sides of the polygon described by the AGV when the PID is used, such as in the side between (-2.5, 1.25) and (-1.25, 0). Therefore, the FLC again gives a better performance. As expected, in both cases the largest peaks in the error appear when the AGV travels through the edges of the polygon.

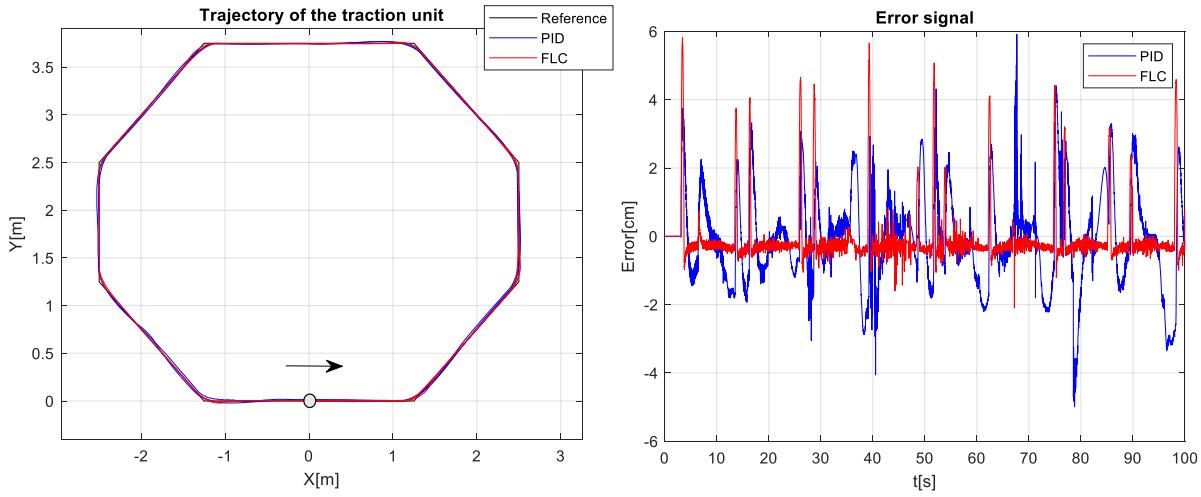

Fig. 7. An 8-sided polygon trajectory (left) and corresponding guiding error (right)

When the reference has more sides, meaning more abrupt changes, it is harder to follow it, as in the case of the 24-sided polygon shown in Fig. 8, left. The error oscillations with the PID are more noticeable and, on the contrary, the FLC better follows the reference, only small errors appear close to the corners. In the error signal is also possible to see that the PID presents more and larger peaks than the FLC. As in the case of the octagon, the peaks seem to be related to the edges of the polygon. In this trajectory the starting point is (0,0) and the AGV goes clockwise.

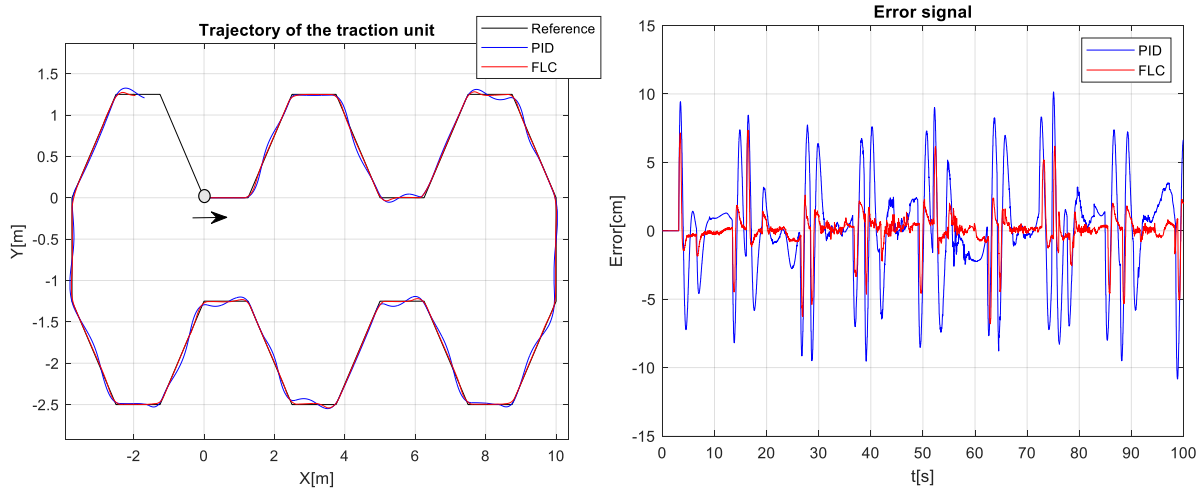

Fig. 8. A 24-sided polygon trajectory (left) and corresponding guiding error (right)

To check if a controller that has been optimized for a specific trajectory can be used for different ones, several experiments were carried out. The PID and FLC controllers have been optimized to track a reference and then have been applied and evaluated with the rest of the trajectories. Table 2 summarizes all these combinations, showing the mean absolute error (MAE) for each trajectory and controller. Each row shows the results obtained when the controller is optimized with the trajectory indicated by the first column of the table. In turn, each column indicates the trajectory used to test the PID and the FLC controllers. The last row presents the average value of the previous rows with the percentage of error reduction achieved with the intelligent controller. This table allows us to confirm the results shown in the previous figures in a quantitative way. The best results per column have been boldfaced.

For all the trajectories the MAE obtained with the FLC is smaller than the one obtained with the PID. In the case of the PID, the best result is achieved with the combination lemniscate (training)-lemniscate (testing). In the case of the FLC, the best result is obtained with the combination ellipse-ellipse. On the whole, trajectories with less abrupt changes allows better results. It is interesting to note how the best results per column do not always appear in the main diagonal, that is, contrary to what one may think, training and evaluating the controller with the same trajectory is not always the best option. To mention some examples: in the case of the

PID, to follow the ellipse is better to train the controller with the 8-S polygon; and for the FLC, to follow the 24-S polygon the best training trajectory is the ellipse. A possible explanation may be that if a trajectory is difficult to follow, many more iterations will be necessary to find a solution if we start from scratch. However, if we train the controller with simpler trajectories, we may find viable solutions beforehand that may also be useful for the trajectory to be tested. In our case, the best trajectory to train the PID is the 8-S polygon and the best one to train the FLC is the ellipse. In addition, Table 3 compares the error KPIs provided by both controllers in terms of MAE, root mean squared error (RMSE) and standard deviation (STD). For all trajectories the KPIs obtained with the FLC are lower than with the PID.

Table 2: Comparison of MAE (cm) for different trajectories with both controllers

Tested with	Ellipse		Lemniscate		8-S Polygon		24-S Polygon	
Trained with	PID	FLC	PID	FLC	PID	FLC	PID	FLC
Ellipse	3,54	**0,14**	4,03	---	1,93	**0,28**	2,86	**0,35**
Lemniscate	1,76	0,42	**0,6**	**0,4**	1,75	0,47	1,78	0,62
8-S Polygon	**1,44**	0,31	0,9	0,34	**1,05**	0,51	**1,29**	0,54
24-S Polygon	1,97	0,53	0,83	0,39	2,42	0,75	2,68	0,81
Average	2,1	0,39 (81,4%)	3,43	0,38 (88,9%)	1,37	0,47 (65,7%)	1,57	0,46 (70,7%)

Table 3: Comparison of KPIs for different trajectories with both controllers

Trajectory	MAE (cm)		RMSE (cm)		STD (cm)	
	PID	RL	PID	RL	PID	RL
Ellipse	1.44	**0.14**	1.88	**0.44**	1.88	**0.26**
Lemniscate	0.6	**0.4**	1.09	**0.72**	1.09	**0.67**
8-S Polygon	1.05	**0.47**	1.41	**0.77**	1.41	**0.77**
24-S Polygon	1.29	**0.35**	1.67	**0.98**	1.67	**0.98**
Average	1.1	0.34(69.1%)	1.51	0.73(51.7%)	1.51	0.67(55.6%)

Figures 9-11 show the optimized fuzzy membership functions for the for trajectories. All inputs have three fuzzy sets: negative (blue), zero (red) and positive (yellow). For the error input (ERR), the three first trajectories are optimized by remarkably similar functions, being the negative and zero fuzzy sets very close and partially overlap, and the positive fuzzy set far away from them. However, the membership functions optimized for the 24 S-polygon are symmetric and very close together. For the derivative of the guiding error (DERR) input, the membership functions obtained by the AG with the ellipse and the lemniscate trajectories are similar. However, for the two polygonal trajectories are similar each other but wider for the octagon. Regarding the integral error input (IERR), the fuzzy sets obtained for the ellipse have very wide negative membership function and the same with the positive one for the 24 s-polygon. It may be explained due to the fact that the ellipse drawn by AGV is smaller than the reference and the error tends to be negative, so this function contributes more to reduce the accumulation of this negative error.

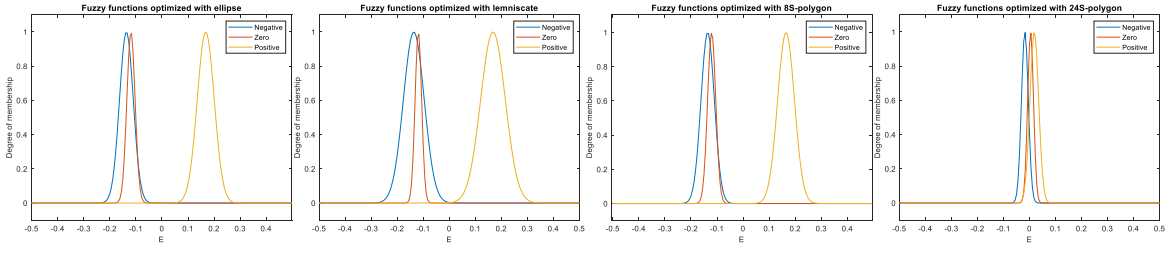

Fig. 9. Optimized fuzzy membership functions of the input ERR

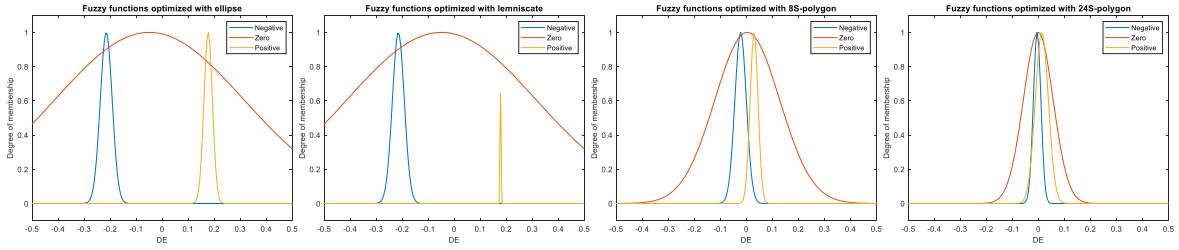

Fig. 10. Optimized fuzzy membership functions of the input DERR

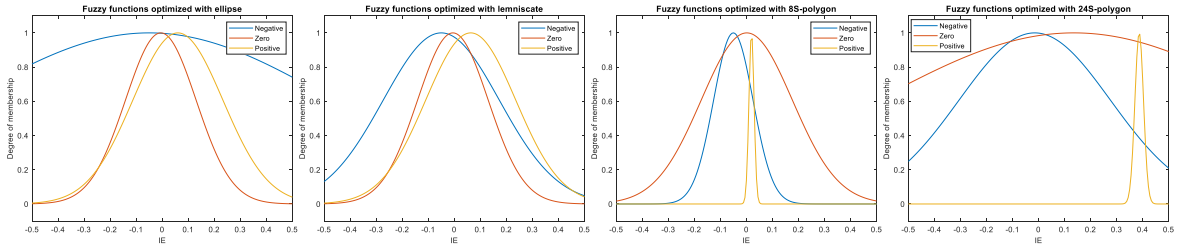

Fig. 11. Optimized fuzzy membership functions of the input IERR

5. Conclusions and future works

AGVs are generally well recognized in the industry and have a considerable influence on the productivity of the company. However, little research has been done to improve its control. The control of industrial AGVs is still mainly based on traditional control techniques such as PID regulators. This paper shows how the use of intelligent control techniques like fuzzy logic and optimization approaches as genetic algorithms can help improve their performance.

The AGV trajectory controller developed in this article applies fuzzy logic and genetic algorithms. Using the mathematical model of a hybrid AGV, which combines a tricycle and a differential robot, its speed control is optimized using genetic algorithms. Once the speed is adjusted, a fuzzy control architecture for trajectory tracking is proposed. This control is also optimized by GA, maximizing the time the AGV follows the circuit and minimizing guiding error. The performance of the fuzzy controller is compared to that of an optimized PID controller. Finally, simulations are used to test the validity of the strategy using four different paths.

As future works, we may highlight the application of this control architecture to other types of AGVs, such as forklifts, and the use of other multi-objective optimization approaches, such as swarm algorithms, to minimize the guiding error and the control effort. We also plan to integrate the controller into the PLC of an AGV prototype.

Data Availability

The findings of this study have been generated by the equations and parameters cited within the article.

Conflicts of Interest

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

References

1. 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
2. Sierra-García, J. E., & Santos, M. (2020). Mechatronic modelling of industrial AGVs: a complex system architecture. *Complexity*, *2020*.

3. Mahulea, C., González, R., Montijano, E., Silva, M. 2021. Path planning of multirobot systems using Petri net models. Results and open problems. Revista Iberoamericana de Automática e Informática Industrial, 18(1):19-31.
4. Sharma, O., Sahoo, N. C., & Puhan, N. B. (2021). Recent advances in motion and behavior planning techniques for software architecture of autonomous vehicles: A state-of-the-art survey. Engineering applications of artificial intelligence, 101, 104211.
5. Suárez Feijóo, R., Palomo Avellaneda, L., Martínez Miralles, J. R., Clos Costa, D., & García, N. (2020). Dual-arm dexterous mobile manipulator with new omnidirectional wheels. Revista RIAI, 17(1), 10-21.
6. Sierra-García, J. E., & Santos, M. (2021). Intelligent control of an UAV with a cable-suspended load using a neural network estimator. Expert Systems with Applications, 183, 115380.
7. Sierra-García, J. E., & Santos, M. (2021). Switched learning adaptive neuro-control strategy. Neurocomputing, 452, 450-464.
8. Ren, Z., Lai, J., Wu, Z., & Xie, S. (2021). Deep neural networks-based real-time optimal navigation for an automatic guided vehicle with static and dynamic obstacles. Neurocomputing, 443, 329-344.
9. Septyan, M., & Agustinah, T. (2019, March). Trajectory tracking automated guided vehicle using fuzzy controller. In 2019 International Conference of Artificial Intelligence and Information Technology (ICAIIT) (pp. 169-174). IEEE.
10. Li, Y., Huang, D., Feng, D., Zhang, L., Wu, X., Huang, S., & Huang, S. (2020). Tracking control algorithm based on fuzzy logic for batch-feeding AGV. In Proceedings of the Seventh Asia International Symposium on Mechatronics (pp. 564-573). Springer, Singapore.
11. Wu, X., & Yang, Y. (2020). Path tracking controller design of automatic guided vehicle based on four-wheeled omnidirectional motion model. International Journal of Automotive and Mechanical Engineering, 17(2), 7996-8010.
12. Krisna, A. R., Rusdinar, A., Yuwono, S., & Nugraha, R. (2017, April). Speed control system design using fuzzy-PID for load variation of automated guided vehicle (AGV). In 2017 2nd International Conference on Frontiers of Sensors Technologies (ICFST) (pp. 426-430). IEEE.
13. Abajo, M. R., Sierra-García, J. E., & Santos, M. (2021, September). Evolutive Tuning Optimization of a PID Controller for Autonomous Path-Following Robot. In International Workshop on Soft Computing Models in Industrial and Environmental Applications (pp. 451-460). Springer, Cham.
14. Zacharia, P. T., & Xidias, E. K. (2020). AGV routing and motion planning in a flexible manufacturing system using a fuzzy-based genetic algorithm. The International Journal of Advanced Manufacturing Technology, 109(7), 1801-1813.
15. Goli, A., Tirkolaee, E. B., & Aydın, N. S. (2021). Fuzzy integrated cell formation and production scheduling considering automated guided vehicles and human factors. IEEE Transactions on Fuzzy Systems, 29(12), 3686-3695.
16. Gola, A., & Kłosowski, G. (2019). Development of computer-controlled material handling model by means of fuzzy logic and genetic algorithms. Neurocomputing, 338, 381-392.
17. 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.