Article

Optimal Fuzzy Controller Design for Autonomous Robot Path Tracking using Population-based Metaheuristics

Alejandra Mancilla [1], Mario García-Valdez [1,*], Oscar Castillo[1] and JJ Merelo-Guervós[2]

[1] Tijuana Institute of Technology;{alejanda.mancilla,mario,ocastillo}@tectijuana.edu.mx
[2] University of Granada; jmerelo@ugr.es
* Correspondence: mario@tectijuana.edu.mx; Tel.:+52-664-123-7806 (M.G.V.)
† This paper is an extended version of our paper published in INFUS-21.

Abstract: In this work we propose, through the use of population-based metaheuristics, an optimization method that solves the problem of autonomous path tracking using a rear-wheel fuzzy logic controller. This approach enables the design of controllers using rules that are linguistically familiar to human users. Moreover, a new technique that uses three different paths to validate the performance of each candidate configuration is presented. We extend on our previous work by adding two more membership functions to the previous fuzzy model, intending to have a finer-grained adjustment. We tuned the controller using several well-known metaheuristic methods, Genetic Algorithms (GA), Particle Swarm Optimization (PSO), Grey Wolf Optimizer (GWO), Harmony Search (HS), and the recent Aquila Optimizer (AO) and Arithmetic Optimization Algorithms. Experiments show that, compared to a published control law, the proposed fuzzy controllers have better RMSE-measured performance. Nevertheless, the experiments also highlight problems with the common practice of evaluating the performance of fuzzy controllers with a single problem case and performance metric, resulting in controllers that tend to be overtrained.

Keywords: Fuzzy Systems; Fuzzy Control; Bioinspired Algorithms.

Citation: Mancilla, A.; García-Valdez, M.; Castillo, O.; Merelo-Guervós, J.J. Optimal Fuzzy Controller Design for Autonomous Robot Path Tracking using Genetic Algorithms. *Symmetry* **2021**, *1*, 0. https://doi.org/

Received:
Accepted:
Published:

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

1. Introduction

Proposed by Lofti Zadeh [1], fuzzy logic introduces the concepts of fuzzy sets and fuzzy logic operators [2]. Contrary to boolean logic, in which an element is a member of a set or is not, in fuzzy logic, elements have a degree of membership to many sets. Fuzzy logic uses so-called membership functions (MFs) to assign a numerical value to each set member, indicating their degree of membership. MFs must be defined for each linguistic variable; these variables can be used in a rule-based system to express knowledge in a way similar to natural language, for instance, the rule "if distance is near" uses the linguistic variable distance, with a fuzzy MF assigning a degree of membership to the set near to each element of the distance domain; for instance, for 2mm and 50mm the function will assign the following degrees of membership: near(2)=.92 and near(50)=.40. These rule based systems can express complex relationships well suited for control applications.

That is why, since the earlier years of fuzzy logic theory, fuzzy inference systems [3] have been applied to control problems [3–6], in many research projects [7,8] and commercial systems.

Many papers address the problem of path tracking control with fuzzy logic, the earlier works used simple fuzzy rules to make adjustments to linear controllers [9], or integrated fuzzy logic to a complex adaptive controller [10]. Meng [11] proposes a Fuzzy PID (proportional–integral–derivative) Controller algorithm to a two-degree-of-freedom arm robot. Antonin et al. [12] propose a set of rules to emulate the way humans drive, using as inputs of curve and distance to the system. We have also reviewed works where authors use controllers for the follow-up and planning of routes. For example, using a simplified kinematic model of the bicycle type using the control law for the navigation and follow-up of a path [13], in work presented by Beleño et al. [14] they propose the planning and tracking of the trajectories of a land vehicle based on the steering control in a natural environment. Guerrero-Castelanos et al. [15] addresses the path following problem for the robot (3, 0) based on its kinematic model and proposes a solution by designing a control strategy that mainly considers the maximum permitted levels of the control signal.

An essential caveat of applying fuzzy control strategies to real-world problems is that we require the use of optimization or adaptive techniques to tune some aspects of the fuzzy inference system. From the membership functions (MFs) that define the linguistic variables, to the definition of a rule-based system, including a defuzzification method [16,17].

Tuning is needed because the fuzzy controllers' performance is highly dependent on the parameters of the fuzzy system used. Moreover, the option of making a manual selection of these parameters is difficult because the search space is of considerable size and requires the validation of establishing the controller's performance by running time-consuming simulations. Evolutionary Algorithms (EAs) and other population-based metaheuristics are often employed in tuning Fuzzy Inference Systems (FISs) [18,19].

Population-based metaheuristics, are stochastic techniques that search for optimal solutions by using two strategies: exploration and exploitation. Exploration searches the space globaly, avoiding to be traped in a local optima, while exploitation, searches locally for nearby promising solutions. Genetic Algorithms (GAs) are the canonical representatives of population-based metaheuristics [20]. GAs work on a set of potential solutions called a population that is evolved for a certain number of generations until a suitable or optimal solution is found [21]. In each generation, potential solutions (individuals) are evaluated, and then surviving individuals reproduce through genetic crossover (exploration) and mutation operators (exploitation) to generate new offspring. Finally, there is a replacement mechanism to select the most adapted offspring and used them to generate a new generation of the population. Most population-based metaheuristics, create an initial set of random candidate solutions, then these are evaluated against a quality function, then taking the quality and the position or structure of each solution, a nature-inspired metaheuristic is applied to generate a new set of candidates.

Population-based metaheuristics have been extensively used for structural optimization tasks [22–25], together with more traditional gradient-based algorithms. These nature-inspired inpired metaheuristic can be broadly grouped into evolutionary algorithms (EAs) [26] and swarm intelligence (SI) [27], as well as other categories; popular EAs are Genetic Algorithms (GAs) [28,29], Genetic Programming (GP) [26],and Differential Evolution (DE) [30], while examples of (SI) [27] are particle swarm optimization (PSO), [31], Grey Wolf Optimization (GWO) [32] and Aquila Optimizer (AO) [33]. Moreover, examples of other categories are the Harmony Search (HS) [34] algorithm, that is inpired by how jazz musicians improvise when playing with others, and finally, the arithmetic optimization algorithm (AOA) [35] inspired on the distribution behavior of the main arithmetic operators in mathematics.

We have found in the literature various studies that optimize the parameters of a fuzzy controller applied to mobile autonomous robots using different bio-inspired metaheuristics [36,37]. Wagner and Hagras [38] propose a GA to evolve the architecture of a type-2 fuzzy controller in robot navigation for real environments; they optimize the standard deviation of Gaussian type-2 MFs. Again a GA is presented by Wu & Wan Tan [39] for evolving the parameters of all the MFs of a coupled-tank liquid-level control system. Astudillo et al. [40] propose a new metaheuristic based on chemical reactions to tune the parameters of a fuzzy controller for a uni-cycle robot. There is also work focused on the metaheuristic optimization of fuzzy type-2 controllers, the main works are reviewed by Castillo [41], and the main reason for using this type of controller is to model the uncertainty of the sensor data or the fuzzy model itself.

Following the preliminary work in [42,43], we now discuss the novelty and main concerns of this paper. We have observed that most of these studies optimize the parameters of MFs directly. For instance, if we have a triangular function for a fuzzy set A, defined by a lower limit a, and upper limit of b and a value m where $a < m < b$ as

$$\mu_{trian}(x) = \begin{cases} 0, & x \leq a \\ \frac{x-a}{m-a}, & a < x \leq m \\ \frac{b-x}{b-m}, & m < x \leq b \\ 0, & x \geq b \end{cases}, \tag{1}$$

each value is optimized independently, sometimes validating only the restriction $a < m < b$. This has the advantage of not limiting the search, because candidate solutions can represent all possible MFs. This also means that the search space is not reduced. In this work, we propose a novel method that adds further restrictions, including a symmetric definition and limiting the range of values of each parameter. To achieved that, we first designed the structure of a parametrizable fuzzy controller, using five membership functions for each fuzzy input variable. In our previous work [42,43], we compared symmetrical and asymmetrical definitions and found that symmetrical restrictions give better results for rear-wheel-based controllers. We propose a new parametrization technique that enables the definition of symmetric MFs, by using an aperture factor instead of the previous method that used a delta from a fixed point. Moreover, in this work, we also propose a change on how candidate controllers are normally evaluated by other works in the literature, by using a single simulation and path as fitness function. As experiments show, in the case of rear-wheel path tracking, evaluating candidate solutions with three simulations using distinct paths, gives better results. As experiments show, these additions greatly improve the controller's optimal design by significantly decreasing the tracking error (RMSE) compared to our previous work. Experimental results also show that this method reduces the risk of generating an over-trained controller with low error for a specific path but cannot function in other paths.

To demonstrate the application of the method, we report an experimental case study using a bicycle-like mobile robot with nonholonomic constraints. In this work, we choose the problem of trajectory tracking since it has the particularity of being naturally symmetric since the error is measured by moving away either to the left or right of the desired path. Furthermore, in the literature, we have not found applications of fuzzy systems to follow the trajectory of a path so that this research could solve similar problems.

The main contribution of this work is to propose, through the use of population-based metaheuristics tied with a parametrizable fuzzy controllers, an optimization method to solve the problem of autonomous path tracking using a rear-wheel controller in the framework of fuzzy logic. This approach enables the design of controllers using rules that are linguistically familiar to human users.

We structure this document as follows: in Section 2, we present the proposed method, configurations, and we describe the experimental setup. In Section 3 we present the results achieved, and finally, in Section 4 we discuss the results and highlight future research directions.

2. Materials and Methods

2.1. Rear-Wheel Feedback and Kinematic Model

In this work, we use a simplified model of a bicycle-type kinematic robot consisting of two wheels connected by a rigid link of size l with nonholonomic restrictions [44,45]. The front-wheel can steer in the axis normal to the plane of motion, the steering angle is δ (see Figure 1), The position of the midpoint of the rear-wheel is given by the coordinates

x_r and y_r. The heading θ is the angle of the link between the two wheels and the x axis. We follow the model describe in [46], with the differential constraint:

$$\begin{aligned}\dot{x}_r &= v_r \cos(\theta),\\ \dot{y}_r &= v_r \sin(\theta),\\ \dot{\theta} &= \tfrac{v_r}{l}\tan(\delta).\end{aligned} \tag{2}$$

The controller selects the steering angle δ with a value between the limits of the vehicle $\delta \in [\delta_{min}, \delta_{max}]$ and a desired velocity v_r again limited by $v \in [v_{min}, vmax]$. The heading rate ω is related to the steering angle by

$$\delta = \arctan\left(\frac{l\omega}{v_r}\right), \tag{3}$$

and we can simplify the heading dynamics to

$$\dot{\theta} = \omega, \quad \omega \in \left[\frac{v_r}{l}\tan(\delta_{min}), \frac{v_r}{l}\tan(\delta_{max})\right]. \tag{4}$$

.

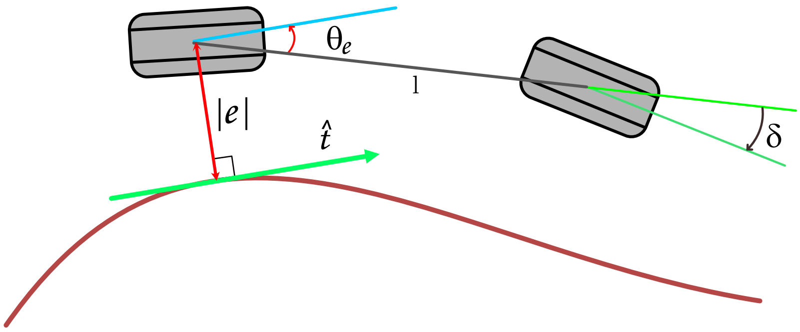


Figure 1. Feedback and actuator variables for the rear-wheel-based control. The magnitude of e illustrated in red, is the error measured from the rear wheel to the nearest point on the path. When e > 0, the wheel is at the right of the path, and when e < 0 is at the left. θ_e is the difference between the tangent at the nearest point in the path and the the heading θ. The output of the controller is the heading rate ω, from which we calculate the steering angle δ of the front wheel.

Now we explain the path tracking method described by Paden et al. [46]. This controller takes the feedback from the rear-wheel position as Figure 1 illustrates. The path (shown in red in the figure) is a continuous function with properties described in [47], and the feedback is a function of the nearest point on the reference path given by

$$s(t) = \arg\min_{\gamma} \|(x_r(t), y_r(t)) - (x_{ref}(\gamma), y_{ref}(\gamma))\|. \tag{5}$$

and the tracking error vector is

$$d(t) = (x_r(t), y_r(t)) - (x_{ref}(s(t)), y_{ref}(s(t))) \tag{6}$$

The heading error is based on a unit vector $\hat{t}$ (shown in green on Figure 1) tangent to the path at $s(t)$ given by

$$\hat{t} = \frac{\left(\left.\frac{\partial x_{ref}}{\partial s}\right|_{s(t)}, \left.\frac{\partial y_{ref}}{\partial s}\right|_{s(t)}\right)}{\left\|\left(\frac{\partial x_{ref}(s(t))}{\partial s}, \frac{\partial y_{ref}(s(t))}{\partial s}\right)\right\|}, \tag{7}$$

The error e is the cross product of the two vectors

$$e = d_x\hat{t}_y - d_y\hat{t}_x \tag{8}$$

The heading error uses the angle θ_e between the robot's heading vector and $\dot{(}t)$

$$\theta_e(t) = \theta - arctan_2\left(\frac{\partial x_{ref}(s(t))}{\partial s}, \frac{\partial y_{ref}(s(t))}{\partial s}\right) \tag{9}$$

2.2. Fuzzy Controller

To design a fuzzy controller for the model we just described, we must decide which variables we are going to treat as fuzzy variables. First, we must consider the error (e), defined as the distance from the rear wheel to the path's closest point. This error is positive if it is on the right and negative if on the left of the path. Another input variable is the angle θ_e defined between the heading vector and the tangent vector of the path. This angle can also be negative or positive depending on the vehicle's position concerning the path; the controller output is the heading rate ω, which allows us to calculate the steering angle δ. The target velocity of the robot will be constant, so we will not control (or fuzzyfy) the velocity v using the fuzzy controller. We will use the following simple proportional controller instead

$$a = K_p(v_{ref} - v_r). \tag{10}$$

2.2.1. Parametrizable Fuzzy Controller

In this section, we explain the proposed a parametrizable fuzzy controller, suitable to optimization using a bio-inspired metaheuristic. We must define the structure of the fuzzy controller consisting of fuzzy rules and parametrizable membership functions. Extending our previous work [43], in which we used three MFs for each variable we now add two more membership functions to the previous fuzzy model, with the intention of having a finer grain of control. Adding more MFs also adds more complexity to the rules, and now we have even more parameters to tune. Instead of having just two values for each type of error `hi` and `low` we add a middle value, represented by the `medium` membership function. The knowledge in the fuzzy rule base is now more complex than before, with 25 rules. In this case, defining the rules was not done by simply specifying a driver's knowledge. We needed to adjust the rules by running several simulations. The rules are presented in Table 1.

Table 1. Proposed fuzzy rules for the basic controller with three membership functions.

Rule 1:	If θ_e is	`hi_neg`	and e is	`hi_neg`	then ω is	`hi_pos`
Rule 2:	If θ_e is	`hi_neg`	and e is	`med_neg`	then ω is	`hi_pos`
Rule 3:	If θ_e is	`hi_neg`	and e is	`low`	then ω is	`hi_pos`
Rule 4:	If θ_e is	`hi_neg`	and e is	`med_pos`	then ω is	`med_pos`
Rule 5:	If θ_e is	`hi_neg`	and e is	`hi_pos`	then ω is	`low`
Rule 6:	If θ_e is	`med_neg`	and e is	`hi_neg`	then ω is	`med_pos`
Rule 7:	If θ_e is	`med_neg`	and e is	`med_neg`	then ω is	`med_pos`
Rule 8:	If θ_e is	`med_neg`	and e is	`low`	then ω is	`med_pos`
Rule 9:	If θ_e is	`med_neg`	and e is	`med_pos`	then ω is	`med_pos`
Rule 10:	If θ_e is	`med_neg`	and e is	`hi_pos`	then ω is	`low`
Rule 11:	If θ_e is	`low`	and e is	`hi_neg`	then ω is	`hi_pos`
Rule 12:	If θ_e is	`low`	and e is	`med_neg`	then ω is	`low`
Rule 13:	If θ_e is	`low`	and e is	`low`	then ω is	`low`
Rule 14:	If θ_e is	`low`	and e is	`med_pos`	then ω is	`low`
Rule 15:	If θ_e is	`low`	and e is	`hi_pos`	then ω is	`hi_neg`
Rule 16:	If θ_e is	`med_pos`	and e is	`hi_neg`	then ω is	`low`
Rule 17:	If θ_e is	`med_pos`	and e is	`med_neg`	then ω is	`med_neg`
Rule 18:	If θ_e is	`med_pos`	and e is	`low`	then ω is	`med_neg`
Rule 19:	If θ_e is	`med_pos`	and e is	`med_pos`	then ω is	`med_neg`
Rule 20:	If θ_e is	`med_pos`	and e is	`hi_pos`	then ω is	`med_neg`
Rule 21:	If θ_e is	`hi_pos`	and e is	`hi_neg`	then ω is	`low`
Rule 22:	If θ_e is	`hi_pos`	and e is	`med_neg`	then ω is	`med_neg`
Rule 23:	If θ_e is	`hi_pos`	and e is	`low`	then ω is	`hi_neg`
Rule 24:	If θ_e is	`hi_pos`	and e is	`med_pos`	then ω is	`hi_neg`
Rule 25:	If θ_e is	`hi_pos`	and e is	`hi_pos`	then ω is	`hi_neg`

The other component of a fuzzy controller are MFs, these must be defined as a parametrizable structures, suitable to be optimized using a metaheuristic. We describe this structures in the following section.

2.3. Parametrizable Membership Functions

In this section, we describe our proposed method for MFs parameter optimization. First, we need to establish which parameters of the MFs we are going to keep fixed and which parameters we are going to optimize. As general rule, we keep the MFs symmetrical around zero, this means that the middle point of the triangular MF for `low` will be fixed at zero in all cases. We also kept the extreme values of `high` trapezoidal MFs, fixed at 50 and 5, positive or negative depending on the side. We kept the parameters of ω fixed, to limit the search space. The parameters are illustrated in Figure 2. In this case, we just needed ten variables to parameterize the controller.

Another essential aspect to consider when tuning the above parameters is the range of values that each parameter can have. Normally, we keep all the parameters in the same range when using a population-based metaheuristic. In our previous work [43], we compared two ranges $[0,1]$ and $[0,2]$ our experiments showed better results with the narrower range, so we selected the same configuration for the three MFs controllers for this work. In this work, we propose a simple technique to change the tuning ranges for the MFs while keeping the adjustable parameters in the same range of values $[0,1]$. We define different ranges for the input variables and normalize the values before they are passed to the membership functions. We can treat this as an aperture factor, now each parameter of a MF can have a distinct range, while keeping the parameters or be optimized fixed on $[0,1]$. We defined from our experience tha range for each parameter, these are shown in Table 3.

Table 2. Ten parameter configuration for five MFs fuzzy controller.

Variable	Linguistic Value	MF	Parameters
θ_e	high negative	μ_{trap}	$[-50, -5, -b, -b+c]$
θ_e	medium negative	μ_{tria}	$[-d-e, -d, -d+e]$
θ_e	low	μ_{tria}	$[-a, 0, a]$
θ_e	medium positive	μ_{tria}	$[-d-e, d, d+e]$
θ_e	high positive	μ_{trap}	$[b-c, b, 5, 50]$
$error$	high negative	μ_{trap}	$[-50, -5, -g, -g+h]$
$error$	medium negative	μ_{tria}	$[-i-j, -i, -i+j]$
$error$	low	μ_{tria}	$[-f, 0, f]$
$error$	medium positive	μ_{tria}	$[-i-j, i, i+j]$
$error$	high positive	μ_{trap}	$[g-h, g, 5, 50]$
ω	high negative	μ_{trap}	$[-50, -5, -1, -0.5]$
ω	medium negative	μ_{tria}	$[-1, -0.5, 0]$
ω	low	μ_{tria}	$[-0.5, 0, 0.5]$
ω	medium positive	μ_{tria}	$[0, 0.5, 1]$
ω	high positive	μ_{trap}	$[0.5, 1, 5, 50]$

Table 3. Ranges defined for each parameter for the 5MF controller.

Parameter	Range	Parameter	Range
a	[0,1]	f	[0, 1]
b	[0.5,2]	g	[0.5, 2]
c	[0,2]	h	[0, 2]
d	[0.5,1.5]	i	[0.5,1.5]
e	[0,1]	j	[0, 1]

2.3.1. Optimization Problem Formulation

An optimization procedure is needed to tune the paremeters of the membership functions in order to generate a fuzzy controller that maintains a low positional error over a desired path. The optimization problem can be defined as searching for the parameter vector x^{mf} that defines the membership functions which minimize the error. This can be defined as:

$$\underset{x^{mf}}{\arg\min}\{FC_{error}\} \tag{11}$$

where the fitness function FC_error is the average RMSE of three simulations:

$$FC_{error} = \frac{1}{3}\sum_{k=1}^{3} rmse(FC(x^{mf}), s_k, t_{max}) \tag{12}$$

in which k is the number of paths s_k. A simulation consists of runing a control problem as defined in Section 2.1 in which the control is performed by the fuzzy controller $FC(x^{mf})$ with membership functions defined by x^{mf}. The constant t_{max} is the number of cycles executed in one simulation. As described before, the controller objective is to minimize the tracking error, to obtain the RMSE we can use the tracking error vector defined in Equation 13:

$$rmse = \sqrt{\frac{\sum_{t=1}^{t_{max}}(x_r(t), y_r(t)) - (x_{ref}(s_k(t)), y_{ref}(s_k(t)))^2}{t_{max}}} \tag{13}$$

The vector x^{mf} defines the paremeters of the membership functions defined in Table 2 and the knowledge base in Table 1:

$$x^{mf} = \{x_1^a, x_2^b, x_3^c, x_4^d, x_5^e, x_6^f, x_7^g, x_8^h, x_9^i, x_{10}^j\} \tag{14}$$

The range of all parameters is between 1 and 0:

$$0 \leq x_i \leq 1 \tag{15}$$

2.3.2. Metaheuristic Optimization Procedure

In general, a population-based metaheuristic, needs to evaluate the fitness of each candidate solution (also called individuals) in its population. This process is illustrated in Figure 2. Each candidate solution is represented by a vector x^{mf}, here implemented as a list of objects of type `float` . To evaluate each solution, we need to generate an instance of the fuzzy controller. Once created, the fuzzy controller is passed as a parameter, together with the paths the mobile robot will follow. The output of the simulations is the RMSE of the accumulated errors e obtained during the simulations. We consider this measure as the fitness of the candidate solution.

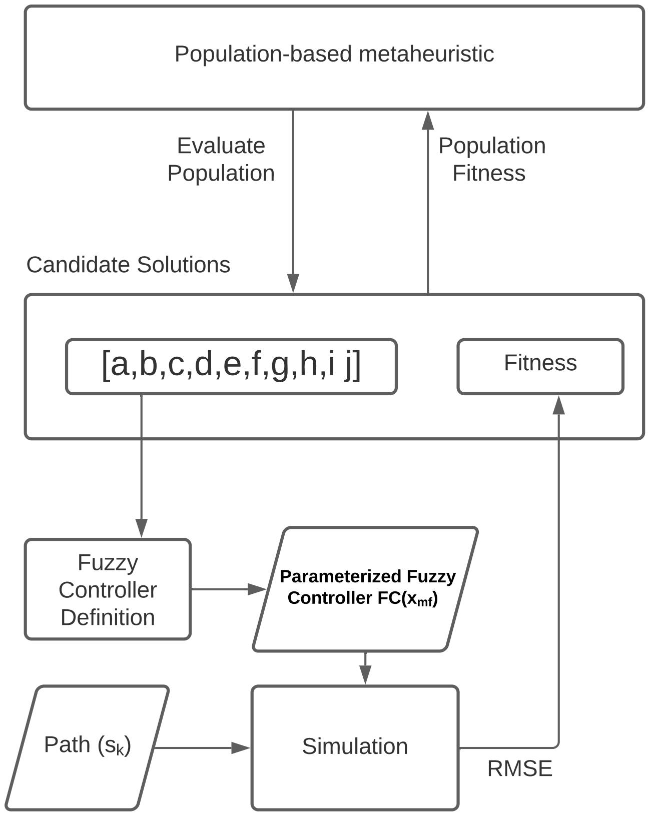
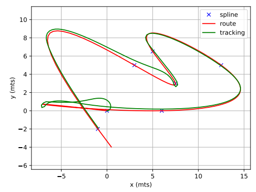
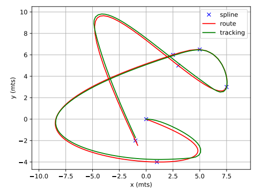
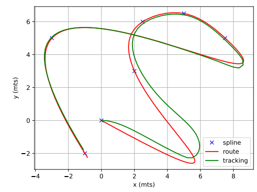


Figure 2. For each candidate solution in the population, a controller is created with the parameters, this controller is then tested by running one or more simulations. The RMSE of the tracking is considered as the fitness for that particular candidate solution. This process is repeated for each member of the population.

In our previous work [43] we used a GA to evolve the controllers using a single path; we noticed that this could lead to over-training. Similar to what happens in supervised learning algorithms, the controller found by the GA could be specialized only to the path used for its evolution. We found that the controller with only three MFs could follow new unseen paths, but this was not the case for the five MFs controller. It is well known that in rule-based learning, adding more rules can harm the capacity to generalize to unseen problems [48]. Noticing this, we added a list of three paths to evaluate the fitness for the experiments, which is the average RMSE of the three simulations. We ran experiments with one and three paths. We defined each path using cubic splines over a list of coordinates; this is a common approach in the literature [49]. The paths and the parameters to define them are shown in Table 4. The first path was used on previous work and was taken from the library of [50], this path starts with a very narrow curve to the left that is difficult for controllers to follow, but it remains differentiable throughout the path, we call this path "M". The other paths are called "A" and "S" and have smoother curves, with long straight segments and different curvatures. All paths have seven anchor points for the cubic spline.

Table 4. Tracks used for fitness evaluation, they are defined by cubic splines with the parameters shown below each plot.

Track M	**Track A**	**Track S**
		
$ax = [0, 6, 12, 5, 7.5, 3, -1]$ $ay = [0, 0, 5, 6.5, 3, 5, -2]$	$ax = [0, 1, 2.5, 5, 7.5, 3, -1]$ $ay = [0, -4, 6, 6.5, 3, 5, -2]$	$ax = [0, 2, 2.5, 5, 7.5, -3, -1]$ $ay = [0, 3, 6, 6.5, 5, 5, -2]$

2.3.3. Complexity of the Optimization Procedure

The computational complexity of the optimization procedure described earlier, depends mostly on the evaluation of the fitness function. In this case, is difficult to estimate the cost of each simulation step because it depends on an ordinary differential equation solver (ODEInt), in particular this is a multi-step solver (lsode) and the number of iterations changes depending on the initial conditions. Moreover, in order to calculate the current nearest point to the path as described in Equation 5, there is a local optimization procedure to find γ to establish this nearest point $(x_ref(\gamma), y_ref(\gamma))$.j Nevertheless, we can consider the cost of the function evaluation as domain dependent and with a high order complexity of at least $\backslash^{\ni}$. Because of this, normally the number of function evaluations needed to find a solution considered when establishing the performance of a metaheuristic algorithm.

On the side of population-based algorithm, and in particular all the algorithms that we tested on this paper, follow the same procedure: randomly initialize the population, this has a $\mathcal{O}(n)$ complexity, with n the population size (the number of candidate solutions). After the evaluation, there is a step for updating the solution the complexity for this is normally $\mathcal{O}(m * n) + \mathcal{O}(m * n * l)$ m is the number of iterations and l the number or parameters in the fitness function. Some algorithm add a step for keeping only the generated solutions if the new fitness is better, this has an additional cost of $\mathcal{O}(n)$, finally there are algorithms that order the population by fitness after every iteration, this adds a complexity of $\mathcal{O}(nlogn)$.

2.3.4. Experimental Setup

In this section, we compare the parameter optimization procedure proposed in the previous section, using the following algorithms:

1. Genetic Algorithm (GA) [20]
2. Particle Swarm Optimization (PSO) [27]
3. Aquila Optimization (AO) [33]
4. Grey Wolf Optimizer (GWO) [32]
5. Arithmetic Optimization Algorithm (AOA) [35]
6. Harmony Search (HS) [34]

All algorithms have the same population size of 50, and the number of iterations was set at 30, from this values, the total number of function evaluations is 1000. The parameters of the comparative algorithms is shown in Table 5.

Table 5. Parameter values for the algorithms compared

Algorithm	Parameter	Value
GA	Selection	Tournament Selection (k =3)
	Mutation	Gaussian ($\mu = 0.0$ and $\sigma = 0.2$)
	Mutation probability	0.3
	Crossover	One point crossover (probability = 0.7)
PSO	Topology	Fully connected
	Speed limit	Min=-0.25,Max=0.25
	Cognitive and Social constants	$C_1 = 2, C_2 = 2$
AO	α	0.1
	δ	0.1
GWO	Convergence parameter (a)	Linearly decreased from 2 to 0
AOA	α	5
	μ	0.5
HS	HMConsidering Rate (HMCR)	0.95
	Pitch Adjusting Rate	0.05

The fitness function returns the RMSE of the simulation as mentioned earlier, but to economize the computational resources, while the simulations were running, we interrupted those simulations where the robot was clearly out of the path or did not finish near the final point of the path. In these cases, we assigned a very low fitness (we want to minimize the error) of 5000 to the first case and 2000 to the second.

As the basis for comparison, we compare or results against the controller in [46], with the following control law defined as

$$\omega = \frac{v_r \mathcal{K}(s)\cos(\theta_e)}{1 - \mathcal{K}(s)e} - (k_\theta |v_r|)\theta_e - \left(k_e v_r \frac{\sin(\theta_e)}{\theta_e}\right)e, \tag{16}$$

the parameters of the simulations and the canonical controller we are comparing against are summarized in Table 6.

Table 6. Simulation and controller parameters.

Parameter	Value
Wheel-base	$l = 2.5$
Steering limit	$\lvert\delta\rvert \leq \frac{\pi}{4}$
Initial configuration	$x_r(0), y_r(0), \theta(0) = (0, 0, 0)$
Velocity controller configuration	$K_p = 1, v(0) = 0, a(0) = 0$
Target velocity	$v_r = \frac{10}{3}$
Maximum time	50
Control law parameters	$k_e = 0.3, k_\theta = 1.0$

We ran the experiments on a Desktop PC with AMD Ryzen 9 3900x 12-core processor with 24 threads and 48 GB RAM with Ubuntu Linux 21.04, and Python 3.7.5 code. Code and data can be found in the following GitHub repository https://github.com/mariosky/fuzzy-control.

We compared the algorithms using the mean, median and standard deviation of the RMSE of 30 algorithm executions. Moreover, we performed a Wilcoxon rank-sum test between algorithms for performance comparison.

3. Results

This section shows the results of running the optimization of parameters for the controller described in the previous sections, with the average RMSE of three paths to establish the fitness. The descriptive statistics of these results are shown in Table 7. As expected, and because the optimization is non-deterministic, there are a few outliers at both extremes of the performance. In particular, outliers are presented on the GA algorithm with RMSE = 0.18205, and the AOA algorithm with RMSE = 0.7978. The PSO algorithm obtained the lower RMSE average (0.00546) and the best controller (0.00158) and AOA with median of (0.00523).

Table 7. Descriptive statistics (n=30) results for algorithms with an evaluation with three paths.

Algorithm	**Average RMSE**	**Standard Deviation**	**Median**	**Min**	**Max**
GA	0.01564	0.03163	0.00918	0.00574	0.18205
PSO	**0.00546**	0.00202	0.00536	**0.00158**	0.00102
AO	0.00695	0.00210	0.00696	0.00355	0.01407
GWO	0.00617	**0.00170**	0.00643	0.00315	0.00849
AOA	0.03413	0.14403	**0.00198**	0.00198	**0.79780**
HS	0.00774	0.00273	0.00740	0.00320	0.01454

In Figure 3, we can see that the GA algorithm has the worst median and overall results, while the rest algorithms obtained competitive results.

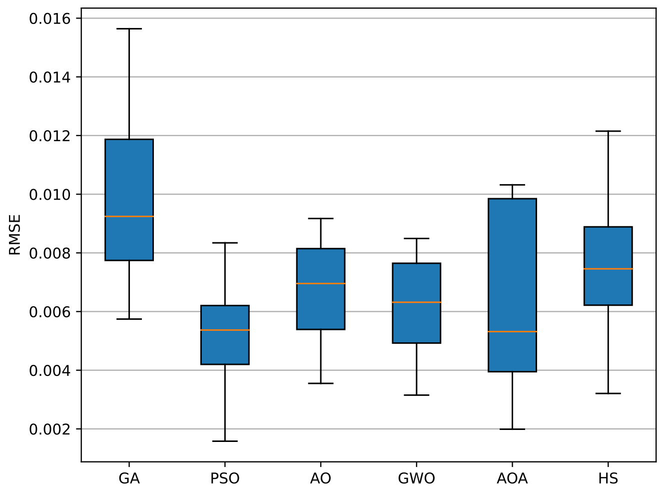


Figure 3. Boxplot showing the results of 30 experiments, with outliers filtered in order to show the plot at a larger scale.

Table 8 gives the results of the Wilcoxon rank-sum test with an alternative hypothesis H_1 comparing if the algorithm in each row has a lower median than the algorithm on each column with a significance level at $\alpha = 0.05$. Results with a p-value lower than 0.05 are shown in bold. Among the six algorithms, the PSO algorithm outperformed the GA, AO, GWO and HS algorithms. The GA algorithm was outperformed by all the algorithms. The AOA algorithm was not outperformed by any algorithm with enough statistical significance. The GWO algorithm beat the HS algorithm, making it the second best, in mumber of wins. Nevertheless, in general, most algorithms (PSO, AO, AOA and GWO) achieved good results.

Table 8. Wilcoxon rank-sum test between algorithms, showing p-values for $H_1 : A < B$

	GA	PSO	AO	GWO	AOA	HS
GA		1.00E+00	1.00E+00	1.00E+00	9.99E-01	9.99E-01
PSO	**6.44E-09**		**2.23E-03**	**3.80E-02**	1.73E-01	**9.54E-05**
AO	**1.20E-05**	9.98E-01		9.22E-01	7.76E-01	1.33E-01
GWO	**1.19E-07**	9.63E-01	7.96E-02		4.80E-01	**1.31E-02**
AOA	**1.02E-03**	8.31E-01	2.28E-01	5.25E-01		1.10E-01
HS	**1.24E-03**	1.00E+00	8.70E-01	9.87E-01	8.92E-01	

To present a qualitative comparasion, we selected two of the best controllers found in the 30 experiments, one from the best algorithm (PSO) and the other from the worst (GA), and then compare them against the baseline, a controller using the control law in Equation 16. This results are presented in Table 9. We can see that even the fuzzy controler optimized with the GA outperformed the control law in two paths. The results of the PSO are highly competitive against the baseline.

Table 9. RMSE of the best controllers in all three paths.

Path	Control law	PSO	GA
Path M	2.003	**0.00217**	0.00396
Path A	0.014	**0.00168**	0.00575
Path S	0.521	**0.00195**	0.00845

When we observe the optimized MFs for both controllers, GA in Figure 4(**a**) and PSO in Figure 4(**b**), we can see that they are similar on the `low` and `high` MFS, for both the θ_r and e. There is a noticeable difference however in the high negative and low negative MFs. in each of them both variables are very similar. We can also see that the MFs of ω remain fixed on both, this is because we kept those parameters fixed. The best parameters found from all algorithms are shown in Table 10.

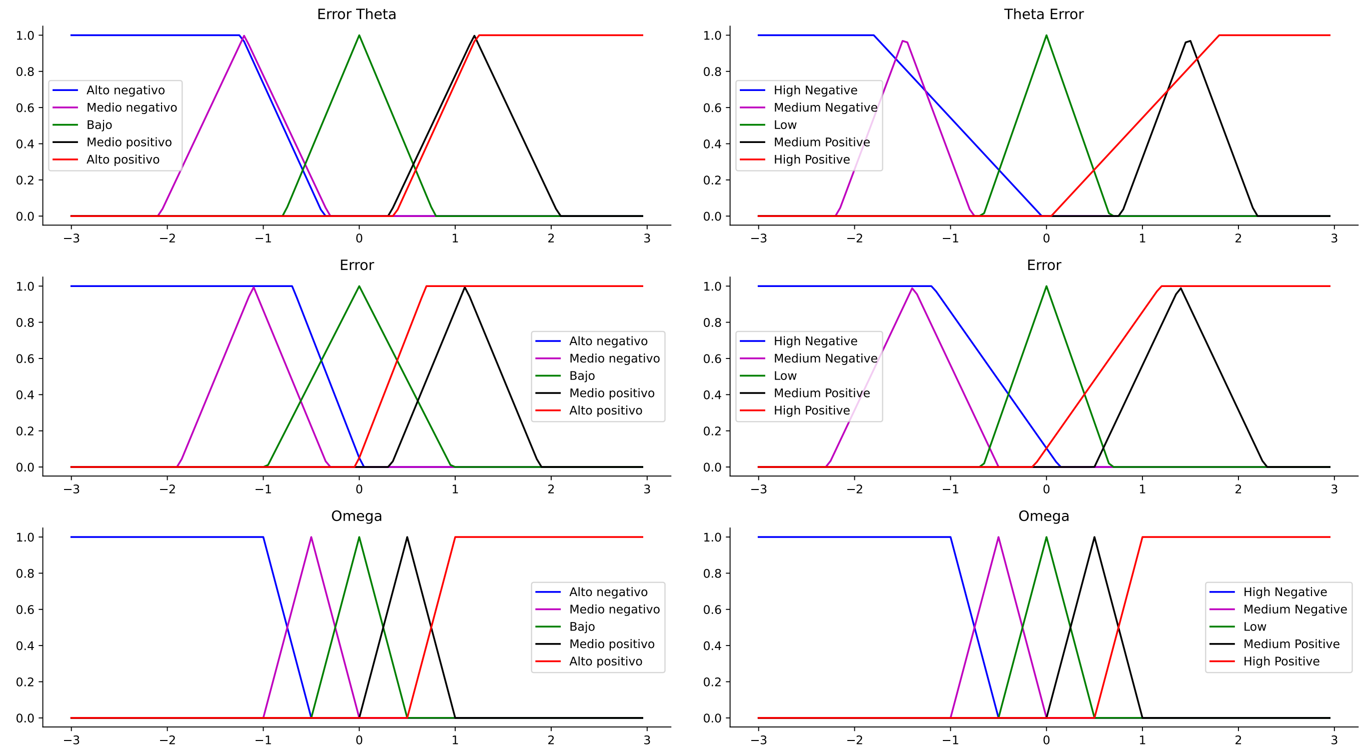
Table 10. RMSE of the best controllers in all three paths.

Algorithm	Parameter Vector
GA	
PSO	[0.78, 0.48, 0.43, 0.69, 0.88, 0.96, -0.13, 0.36, 0.60, 0.77]
AO	[0.71, 0.41, 0.44, 0.22, 0.52, 0.61, 0.12, 0.36, 0.50, 0.18]
GWO	[0.74, 0.46, 0.49, 0.59, 0.40, 0.40, 0.11, 0.36, 0.30, 0.53]
AOA	[1.0, -0.36, 0.37, 0.83, -0.47, 1.0, 0.00, 0.27, 0.27, 0.92]
HS	[0.96, 0.65, 0.55, 0.13, 0.62, 0.56, 0.12, 0.36, 0.96, 0.96]

We now show a plot of the paths and the robot's movement following the path. First, we have the path "M" (see Figure 5), in which all controllers have problems at the beginning with a curve that is very sharp to the left and then a sharp U-turn to go back to the start. We can see that the Control law follows the path with less zig-zag than the GA and PSO controllers, that keep the tracking closer to the path but with noticeable zig-zagging.

The plots in Figure 7 show the "A" path. This time all controllers closely follow the path. Again controller GA has a noticeable zig-zagging but manages to have a lower RMSE than the Control law.

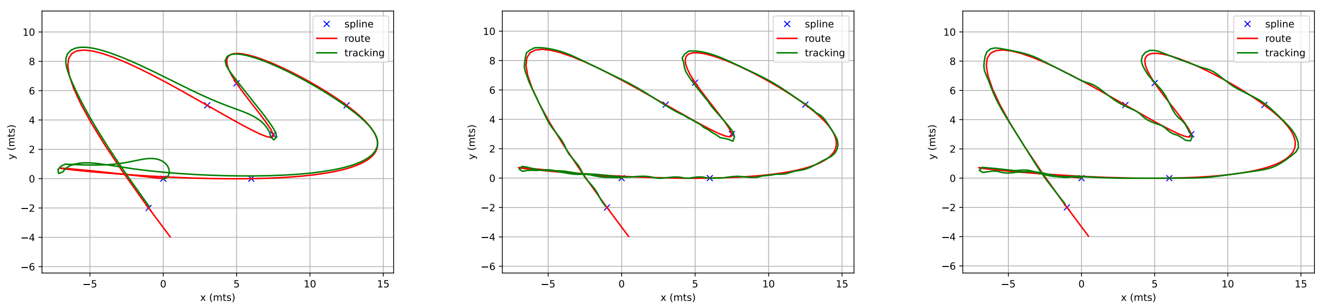
The results of Figure **??** highlight the overall behavior of the three controllers on path "S". The control law takes more time to reach the path when it deviates from the reference because it has smoother steering, but once it is over the path, e does not increase. That is by design because one of the conditions is that a small initial tracking error will remain small. On the other hand controller PSO, does not deviate much from the reference, and has a smoother control than the GA.



(**a**) Optimized membership functions of the best controller optimized with the PSO algorithm.

(**b**) Optimized membership functions of the best controller optimized with the GA algorithm.

Figure 4. Optimized membership functions with PSO and GA algorithms, parameters are shown in Table 10.



(**a**) Control law

(**b**) PSO

(**c**) GA

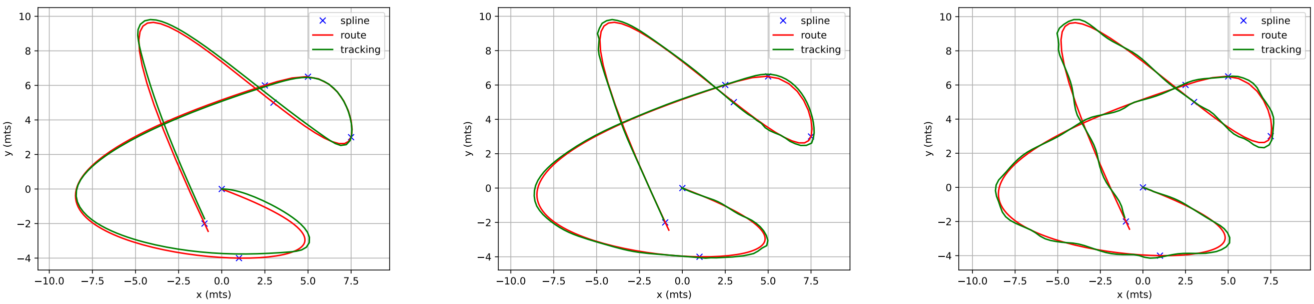
Figure 5. Plot for the best simulations on Track M.

4. Discussion

In this paper, we have proposed a method using a GA metaheuristic for evolving fuzzy controllers by optimizing its membership function's parameters. The fuzzy controller, aimed at autonomous path tracking, using rear-wheel feedback, was optimized by running simulations and measuring the RMSE of the distance between the robot's position and a reference path. We compared two controllers with distinct fuzzy inference systems, using three MFs for each variable and a second with five MFs. Adding more

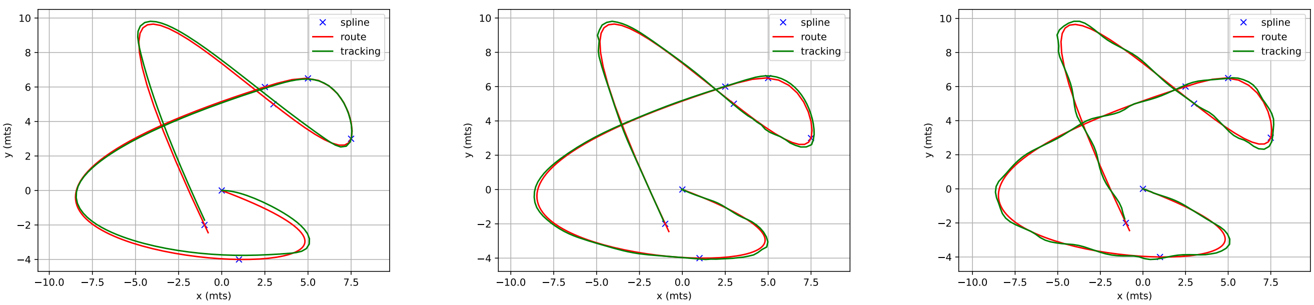
MFs and, consequently, more fuzzy rules added more complexity to the knowledge base, adding more granularity to the representation. We have shown that this change improved the control in terms of RMSE, although needing more computational resources on the evolutionary phase. With additional experiments, we have shown that by adding additional simulations to establish the quality of candidate solutions, during the evolution, the generated controllers have better performance than those evolved with just one simulation.

When comparing the controllers, we found that some solutions had undesired yaw oscillation while keeping a low RMSE; this is a crucial aspect to be considered in future work. We can consider including a damping module or an evaluation metric that negatively weights this kind of oscillation. We can even include a fuzzy variable to the controller; in comparison, the control law also uses the curvature of the path $\mathcal{K}(s)$ as a variable for determining ω. Perhaps we need to treat the evolutionary optimization of fuzzy controllers as a supervised learning task. The assessment of the quality of solutions needs to consider other metrics as part of the desired control properties, such as oscillation, overshoot, and generalization capabilities.



(**a**) Control law (**b**) PSO (**c**) GA

Figure 6. Plot for the best simulations on Track A.



(**a**) Control law (**b**) PSO (**c**) GA

Figure 7. Plot for the best simulations on Track A.

Furthermore, because of the high computational cost, as future work, we will propose a technique to execute these experiments in a distributed way, adding multiple populations such as the island-model, or a pool-based evolutionary approach, that could enhance the search, giving supralinear execution times. Later we could try other bio-inspired methods and compare the results. We could also add more membership functions and test other functions to improve the inference system.

Author Contributions: Conceptualization, A.M., M.G, and O.C.; methodology, A.M.; software, M.G.; validation, O.C. and J.M.; data curation, A.M.; writing—original draft preparation, A.M.; writing—review and editing, M.G. and J.M ; visualization, A.M.; supervision, O.C.; All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by projects TecNM-5654.19-P and DemocratAI PID2020-115570GB-C22.

Institutional Review Board Statement: Not applicable

Informed Consent Statement: Not applicable

Data Availability Statement: All data and code is available with an open source license from https://github.com/mariosky/fuzzy-control

Conflicts of Interest: "The authors declare no conflict of interest."

References

1. Goguen, J. LA Zadeh. Fuzzy sets. Information and control, vol. 8 (1965), pp. 338–353.-LA Zadeh. Similarity relations and fuzzy orderings. Information sciences, vol. 3 (1971), pp. 177–200. *The Journal of Symbolic Logic* **1973**, *38*, 656–657. Publisher: Cambridge University Press.
2. Zadeh, L.A. Fuzzy sets. In *Fuzzy sets, fuzzy logic, and fuzzy systems: selected papers by Lotfi A Zadeh*; World Scientific, 1996; pp. 394–432.
3. Driankov, D.; Hellendoorn, H.; Reinfrank, M. *An introduction to fuzzy control*; Springer Science & Business Media, 2013.
4. Mamdani, E.H. Application of fuzzy algorithms for control of simple dynamic plant. Proceedings of the institution of electrical engineers. IET, 1974, Vol. 121, pp. 1585–1588.
5. King, P.J.; Mamdani, E.H. The application of fuzzy control systems to industrial processes. *Automatica* **1977**, *13*, 235–242.
6. Passino, K.M.; Yurkovich, S.; Reinfrank, M. *Fuzzy control*; Vol. 42, Citeseer, 1998.
7. Yang, X.; Moallem, M.; Patel, R.V. An improved fuzzy logic based navigation system for mobile robots. Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)(Cat. No. 03CH37453). IEEE, 2003, Vol. 2, pp. 1709–1714.
8. Driankov, D.; Saffiotti, A. *Fuzzy logic techniques for autonomous vehicle navigation*; Vol. 61, Physica, 2013.
9. Lee, T.; Lam, H.; Leung, F.H.; Tam, P.K. A practical fuzzy logic controller for the path tracking of wheeled mobile robots. *IEEE Control Systems Magazine* **2003**, *23*, 60–65. Publisher: IEEE.
10. Sanchez, O.; Ollero, A.; Heredia, G. Adaptive fuzzy control for automatic path tracking of outdoor mobile robots. Application to Romeo 3r. Proceedings of 6th International Fuzzy Systems Conference. IEEE, 1997, Vol. 1, pp. 593–599.
11. Bi, M. Control of Robot Arm Motion Using Trapezoid Fuzzy Two-Degree-of-Freedom PID Algorithm. *Symmetry* **2020**, *12*, 665. doi:10.3390/sym12040665.
12. Antonelli, G.; Chiaverini, S.; Fusco, G. A Fuzzy-Logic-Based Approach for Mobile Robot Path Tracking. *IEEE Transactions on Fuzzy Systems* **2007**, *15*, 211–221. doi:10.1109/TFUZZ.2006.879998.
13. Laumond, J.P., Ed. *Robot motion planning and control*; Number 229 in Lecture notes in control and information sciences, Springer: London ; New York, 1998.
14. Beleño, R.D.H.; Vítor, G.B.; Ferreira, J.V.; Meirelles, P.S. Planeación y Seguimiento de Trayectorias de un Vehículo Terrestre con Base en el Control de Dirección en un Ambiente Real. *Scientia et technica* **2014**, *19*, 407–412. Publisher: Universidad Tecnológica de Pereira.
15. Guerrero-Castellanos, J.; Villarreal-Cervantes, M.; Sánchez-Santana, J.; Ramírez-Martínez, S. Trajectory tracking of a mobile robot (3, 0) by means of bounded control. *RIAI-Revista Iberoamericana de Automatica e Informatica Industrial* **2014**, pp. 426–434. Publisher: Elsevier Doyma.
16. Xia, J.; Zhang, J.; Feng, J.; Wang, Z.; Zhuang, G. Command filter-based adaptive fuzzy control for nonlinear systems with unknown control directions. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* **2019**.
17. Isaka, S.; Sebald, A.; Karimi, A.; Smith, N.; Quinn, M. On the design and performance evaluation of adaptive fuzzy controllers. Proceedings of the 27th IEEE Conference on Decision and Control. IEEE, 1988, pp. 1068–1069.
18. Martinez-Soto, R.; Castillo, O.; Aguilar, L.T.; Baruch, I.S. Bio-inspired optimization of fuzzy logic controllers for autonomous mobile robots. 2012 Annual Meeting of the North American Fuzzy Information Processing Society (NAFIPS). IEEE, 2012, pp. 1–6.
19. Salem, M.; Mora, A.M.; Guervós, J.J.M.; García-Sánchez, P. Evolving a TORCS Modular Fuzzy Driver Using Genetic Algorithms. Applications of Evolutionary Computation - 21st International Conference, EvoApplications 2018, Parma, Italy, April 4-6, 2018, Proceedings; Sim, K.; Kaufmann, P., Eds. Springer, 2018, Vol. 10784, *Lecture Notes in Computer Science*, pp. 342–357. doi:10.1007/978-3-319-77538-8_24.
20. Holland, J.H. Genetic algorithms. *Scientific american* **1992**, *267*, 66–73.
21. Muelas, S.; Pena, J.; LaTorre, A.; Robles, V. Algoritmos distribuidos heterogéneos para problemas de optimización continua. VI Congreso Español sobre Metaheurísticas, Algoritmos Evolutivos y Bioinspirados, MAEB, 2009, pp. 425–432.

22. Perez, R.l.; Behdinan, K. Particle swarm approach for structural design optimization. *Computers & Structures* **2007**, *85*, 1579–1588.
23. Durgun, İ.; Yildiz, A.R. Structural design optimization of vehicle components using cuckoo search algorithm. *Materials Testing* **2012**, *54*, 185–188.
24. Yildiz, A.R. A new hybrid particle swarm optimization approach for structural design optimization in the automotive industry. *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering* **2012**, *226*, 1340–1351.
25. Geem, Z.W.; Lee, K.S.; Tseng, C.L. Harmony search for structural design. Proceedings of the 7th annual conference on Genetic and evolutionary computation, 2005, pp. 651–652.
26. Back, T. *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*; Oxford university press, 1996.
27. Kennedy, J. Swarm intelligence. In *Handbook of nature-inspired and innovative computing*; Springer, 2006; pp. 187–219.
28. Holland, J.H.; others. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*; MIT press, 1992.
29. Eiben, A.E.; Smith, J.E. Genetic algorithms. In *Introduction to evolutionary computing*; Springer, 2003; pp. 37–69.
30. Karaboğa, D.; Ökdem, S. A simple and global optimization algorithm for engineering problems: differential evolution algorithm. *Turkish Journal of Electrical Engineering & Computer Sciences* **2004**, *12*, 53–60.
31. Clerc, M. *Particle swarm optimization*; Vol. 93, John Wiley & Sons, 2010.
32. Mirjalili, S.; Mirjalili, S.M.; Lewis, A. Grey wolf optimizer. *Advances in engineering software* **2014**, *69*, 46–61.
33. Abualigah, L.; Yousri, D.; Abd Elaziz, M.; Ewees, A.A.; Al-qaness, M.A.; Gandomi, A.H. Aquila Optimizer: A novel meta-heuristic optimization Algorithm. *Computers & Industrial Engineering* **2021**, *157*, 107250.
34. Geem, Z.W.; Kim, J.H.; Loganathan, G.V. A new heuristic optimization algorithm: harmony search. *simulation* **2001**, *76*, 60–68.
35. Abualigah, L.; Diabat, A.; Mirjalili, S.; Abd Elaziz, M.; Gandomi, A.H. The arithmetic optimization algorithm. *Computer methods in applied mechanics and engineering* **2021**, *376*, 113609.
36. Hernandez, E.; Castillo, O.; Soria, J. Optimization of fuzzy controllers for autonomous mobile robots using the grey wolf optimizer. 2019 IEEE international conference on fuzzy systems (FUZZ-IEEE). IEEE, 2019, pp. 1–6.
37. Lagunes, M.L.; Castillo, O.; Soria, J. Methodology for the optimization of a fuzzy controller using a bio-inspired algorithm. North American Fuzzy Information Processing Society Annual Conference. Springer, 2017, pp. 131–137.
38. Wagner, C.; Hagras, H. A genetic algorithm based architecture for evolving type-2 fuzzy logic controllers for real world autonomous mobile robots. 2007 IEEE International Fuzzy Systems Conference. IEEE, 2007, pp. 1–6.
39. Wu, D.; Tan, W.W. Genetic learning and performance evaluation of interval type-2 fuzzy logic controllers. *Engineering Applications of Artificial Intelligence* **2006**, *19*, 829–841.
40. Astudillo, L.; Melin, P.; Castillo, O. Optimization of a fuzzy tracking controller for an autonomous mobile robot under perturbed torques by means of a chemical optimization paradigm. In *Recent advances on hybrid intelligent systems*; Springer, 2013; pp. 3–20.
41. Castillo, O.; Melin, P. A review on the design and optimization of interval type-2 fuzzy controllers. *Applied Soft Computing* **2012**, *12*, 1267–1278. doi:10.1016/j.asoc.2011.12.010.
42. Mancilla, A.; Castillo, O.; Valdez, M.G. Evolutionary Approach to the Optimal Design of Fuzzy Controllers for Trajectory Tracking. Intelligent and Fuzzy Techniques for Emerging Conditions and Digital Transformation; Kahraman, C.; Cebi, S.; Cevik Onar, S.; Oztaysi, B.; Tolga, A.C.; Sari, I.U., Eds.; Springer International Publishing: Cham, 2022; pp. 461–468.
43. Mancilla, A.; Castillo, O.; Valdez, M.G., Optimization of Fuzzy Logic Controllers with Distributed Bio-Inspired Algorithms. In *Recent Advances of Hybrid Intelligent Systems Based on Soft Computing*; Melin, P.; Castillo, O.; Kacprzyk, J., Eds.; Springer International Publishing: Chm, 2021; pp. 1–11. doi:10.1007/978-3-030-58728-4_1.
44. Pamucar, D.; Ćirović, G. Vehicle route selection with an adaptive neuro fuzzy inference system in uncertainty conditions. *Decision Making: Applications in Management and Engineering* **2018**, *1*, 13–37.
45. De Luca, A.; Oriolo, G.; Samson, C. Feedback control of a nonholonomic car-like robot. In *Robot motion planning and control*; Springer, 1998; pp. 171–253.
46. Paden, B.; Čáp, M.; Yong, S.Z.; Yershov, D.; Frazzoli, E. A survey of motion planning and control techniques for self-driving urban vehicles. *IEEE Transactions on intelligent vehicles* **2016**, *1*, 33–55. Publisher: IEEE.
47. Samson, C. Path following and time-varying feedback stabilization of a wheeled mobile robot. International Conference on Control, Automation, Robotics and Vision, 1992, 1992.
48. Tan, P.N.; Steinbach, M.; Kumar, V. *Introduction to data mining*; Pearson Education India, 2016.
49. Zhang, K.; Guo, J.X.; Gao, X.S. Cubic spline trajectory generation with axis jerk and tracking error constraints. *International Journal of Precision Engineering and Manufacturing* **2013**, *14*, 1141–1146.
50. Sakai, A.; Ingram, D.; Dinius, J.; Chawla, K.; Raffin, A.; Paques, A. PythonRobotics: a Python code collection of robotics algorithms. *CoRR* **2018**, *abs/1808.10703*. _eprint: 1808.10703.